Fairness, Diversity, Explainability, and Robustness for Algorithmic Decision-Making

Madhusudhan Reddy Pittu[1]

CMU-CS-25-135

August 2025

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
David Woodruff, Co-chair
Anupam Gupta, Co-chair
Prasad Tetali
Mohit Singh, Georgia Institute of Technology

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

The work in this dissertation was supported in part by NSF awards CCF-1955785, CCF-2006953, and CCF-2224718.

Keywords: Algorithmic Decision-Making, Fairness, Diversity, Explainability, Robustness, Determinant Maximization, Nash Social Welfare, Permanents, Subspace Approximation, Explainable Clustering, Comparison Oracles, Combinatorial Optimization, Submodular functions.

To my family, and my younger self.

Abstract

This thesis investigates foundational algorithmic challenges that arise when embedding fairness, diversity, explainability, and robustness into computational decision-making. As machine learning systems, resource allocation mechanisms, and data analysis pipelines increasingly influence critical decisions, it is essential that these systems uphold not only efficiency and accuracy but also ethical and structural guarantees. However, enforcing these principles introduces complex trade-offs and computational difficulties.

We address five core problems that capture different facets of algorithmic decision-making under structural and informational constraints: (1) determinant maximization under matroid constraints, modeling the selection of diverse and representative subsets; (2) approximation of the weighted Nash Social Welfare objective, a fairness-centric formulation in indivisible resource allocation; (3) constrained subspace approximation, which enforces group-level representation in data summarization; (4) explainable clustering, which trades off interpretability and clustering quality using decision trees with axis-aligned threshold cuts; and (5) combinatorial optimization using comparison oracles, which enables robust decision-making in uncertain or preference-driven environments.

Each of these problems introduces structural constraints that challenge conventional algorithmic techniques. We develop new frameworks that combine combinatorial methods, convex and non-convex relaxations, convex geometry, and probabilistic methods. The resulting algorithms offer improved approximation guarantees, shed light on key trade-offs between fairness, interpretability, and performance, and support the development of more equitable, interpretable, diverse, and reliable algorithmic systems. These contributions have broad implications in machine learning, economics, data summarization, and human-in-the-loop decision-making.

Acknowledgments

I have been extremely fortunate to receive mentorship from many generous people in theoretical computer science (TCS).

First, I would like to thank my advisor, Anupam Gupta. He is an exemplar of clarity, composure, knowledge, and elegance. Conversations with him invariably bring focus. From his advice, I have learned invaluable skills: how to communicate efficiently—both with others and, most importantly, with myself—which is to say, to think clearly—and to always try the most obvious ideas first. He went out of his way on several occasions to find career opportunities for me and remained patient and supportive through every challenge and setback. I am excited to continue working with him for one more year at NYU. For all of this, I am deeply grateful.

Next, I would like to thank Mohit and Prasad, who took me on as an intern when I was a junior in undergrad and still an amateur researcher. I have worked with Mohit for almost as long as with Anupam; he has profoundly shaped my intuition, my taste in research, and the tools I use to approach problems. Prasad is one of the kindest people I have ever met, a quality that is surely one of the reasons he is so respected and admired in TCS and Mathematics—a treasure trove of knowledge whose insights are consistently illuminating. His recommendation was pivotal in my admission to CMU, and I am lucky to have been mentored by him.

I would also like to thank David Woodruff, who began co-advising me this past year. Working with him is an education in intellectual velocity. He generates a diverse set of ideas to crack a problem, and his problem-solving arsenal is enormous. Working with him is inspiring—and keeping up with his ideas is a happy challenge.

Working with Ola Svensson has been delightful and full of laughter. He has a profound intuition for probabilistic thinking, and witnessing him find the elegant path through a thicket of complexity has been a constant source of inspiration. The project with Anupam and Ola was instrumental in making probabilistic thinking one of my key tools.

But none of this would have been possible without returning to the very beginning of my journey. I am indebted to Arindam Khan for introducing me to TCS and laying the foundation for my career. He saw potential in me when I was an undergrad who had never read a paper. I will never forget the summer internship with my co-interns Arnab Maiti and Amatya Sharma. Arindam taught me the foundations of research and collaboration: how to read papers, write mathematics in LaTeX, prepare academic presentations, and think through problems together during daily brainstorming sessions. He catalyzed my career by recommending me to Mohit and Prasad. I owe my research career to him.

I am also grateful for other formative mentorship opportunities. I thank Yury Makarychev and Ali Vakilian for the internship at TTIC—I learned a great deal about various ways to round semi-definite programming solutions from Yury, and Ali was a pleasure to work with. I thank Sreenivas Gollapudi, Aaron Schild, Kostas Kollias, and Ali Kemal Sinop for the Student Researcher opportunity at Google, where applying theoretical ideas to real-world problems was a deeply inspiring experience.

My research was a fundamentally collaborative endeavor, and I am indebted to my co-authors. From each of you, I learned something distinct about the art of collaboration and discovery: Aditya Bhaskara, Sepideh Mahabadi, Guru Guruganesh, Jon Schneider, Renato Paes Leme, Debmalya Panigrahi, Vincent Cohen-Addad, Euiwoong Lee, Tommaso d'Orsi, Andreas Wiese, Tobias Mömke, Mathieu Mari, and Waldo Gálvez.

I have been privileged to interact with and learn from many professors and legends in the field who have influenced my thinking, including Jack Edmonds, Gerard Cornuéjols, Ravindran Kannan, Chandra Chekuri, Jan Vondrák, Nikhil Bansal, R. Ravi, Santosh Vempala, Venkatesan Guruswami, Ryan O'Donnell, and Daniel Sleator. I am also grateful to Karthik Chandrasekharan, Jugal Garg, Siddharth Barman, Pravesh Kothari, and Sahil Singla; talking to them has always been a pleasure. Finally, sincere thanks to Mor Harchol-Balter and Fatma Kılınç-Karzan for their guidance and help with my thesis requirements.

To my colleagues at CMU, thank you for the daily conversations about research and everything else. It was always a pleasure talking with you: Bernardo Subercaseaux, Sherry Sarkar, Daniel Hathcock, Tolson Bell, Olha Silina, Siyue Liu, Noah Singer, William He, Jeff Xu, Tim Hsieh, Yash Savani, Victor Akinwande, and Keerthana Gurushankar. To my external colleagues, thank you for the camaraderie at workshops and conferences: Omar Alrabiah, Janani Sunderesan, Debajyoti Kar, Kishan Gowda, Pooja Kulkarni, Subhang Kulkarni, David Zheng, Prasanth Amireddy, Ekalavya Sharma, Ishan Bansal, Rameesh Paul, Koustav Bhanja, Jatin Yadav, Harmender Gahlawat, Vaishali Surianarayanan, Neel Patel, and Anish Hebbar.

My journey was sustained by the encouragement of friends, both academic and personal. To my academic friends—Aditi Laddha, Adam Brown, Mik Zlatin, Ioannis Anagnostides, Rhea Jain, Arnab Maiti, Amatya Sharma, Aditya Anand, Chaitanya Nalam, Milind Prabhu, Gary Hoppenworth, Neel Karia, Tushant Mittal, and Shashank Srivastava—and my personal friends—Anup Kumar, Parth Malpathak, Mohan Kumar Srirama, Varsha Reddy Redla, Shreya Terupally, AVS Nikhil, Medha Kaushika, Ramya Narayanasamy, Jnana Sai, Jeevana Reddy, Sai Krishna Meka, Nikhil Reddy Ramolla, Vignesh Veeramakali, Kaushik Arcot, and Saurav Musunuru—thank you for your unwavering friendship. I am especially grateful to Mohan, Shreya, and Varsha for making Pittsburgh feel like home. Special thanks also to my seniors and friends Sabir Shaik, Sai Sandeep Reddy Pallerla, and Praneeth Kacham for their guidance and support at various steps along this journey.

Finally, I thank my parents, Srinivasa Reddy and Siva Parvathi, whose boundless love and countless sacrifices created the foundation upon which this entire endeavor was built. I dedicate this thesis to some of my heroes in TCS: Jack Edmonds, Alexander Schrijver, László Lovász, Avi Wigderson, and Manuel Blum. I would like to think like them one day.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

As algorithmic decision-making becomes increasingly influential in machine learning, resource allocation, and data analysis, it is essential to ensure that these methods promote fairness, diversity, explainability, and robustness while maintaining computational efficiency. Striking this balance introduces significant challenges, as optimizing for one of these properties can often come at the expense of another.

Fairness ensures that algorithmic outcomes do not disproportionately disadvantage individuals or groups, with resource allocation being a prominent example. Diversity plays a key role in producing representative and inclusive outcomes, especially in selection and summarization tasks. Explainability is crucial for building interpretable models that allow stakeholders to understand and trust algorithmic decisions, particularly in clustering. Robustness is critical in settings where information is incomplete, uncertain, or noisy, such as preference elicitation or human-in-the-loop decision-making.

Despite the growing importance of these principles, many traditional optimization techniques focus primarily on efficiency and accuracy, often neglecting the structural and ethical considerations required for equitable and interpretable decision-making. Addressing these concerns requires developing algorithms that explicitly incorporate fairness constraints, promote diversity, impose explainability, and ensure robustness to structural or informational uncertainty—all without excessively compromising performance.

This thesis investigates five key problems that encapsulate these challenges:

1. Determinant maximization under matroid constraints [35, 36],
2. Approximating weighted Nash Social Welfare [34],
3. Price of explainability for clustering [92],
4. Low-rank approximation with fair representation [25],

5. Combinatorial optimization using comparison oracles.

Each of these problems highlights a different aspect of fairness, diversity, explainability, and robustness:

1. **Diversity**: Determinant maximization aims to select a diverse and representative subset from a larger collection of elements—for example, choosing k illustrative images from search results. These elements may represent resources, features, or data points, modeled as vectors in $\mathbb{R}^d$. The diversity of a subset is measured by the *volume* of the parallelepiped spanned by its corresponding vectors, ensuring a well-conditioned and diverse representation of the space. In many applications, the selected elements must also satisfy combinatorial constraints, among which matroid constraints form a broad and well-studied class. **Determinant maximization under matroid constraints** has numerous applications, including statistics, resource allocation, network design, and convex geometry.

   We present a combinatorial algorithm that provides improved approximation guarantees for this problem. Leveraging matroid intersection, our approach overcomes the limitations of convex relaxations and achieves the best-known approximation factors for the problem. See §1.2 for a more elaborate introduction and theorem statements.

2. **Fairness:**

   (a) The Nash Social Welfare (NSW) objective provides a balance between fairness and efficiency in resource allocation. Given a set of indivisible goods to be allocated among agents with individual valuations, NSW is defined as the geometric mean of the agents' utilities. It captures the trade-off between maximizing average utility and ensuring a high minimum utility across agents. This measure is central to fair division, as maximizing NSW guarantees both envy-freeness up to one good (EF1) and Pareto efficiency (PO). However, computing an exact NSW-maximizing allocation is NP-hard, motivating research into efficient approximation algorithms. Prior work has explored connections to Fisher market equilibria, convex programming, and stable polynomials to approximate the optimal NSW within constant factors.

   Extending this line of work, our contribution focuses on the more general **weighted Nash Social Welfare** problem, where agents have different weights reflecting varying levels of priority or entitlement. We develop novel convex and non-convex relaxations and round them to approximate the optimal weighted NSW allocation, achieving improved approximation factors. See §1.3 for a more elaborate introduction and theorem statements.

   (b) High-dimensional datasets often have low intrinsic dimensionality, making subspace approximation essential for data analysis. The *Constrained Subspace Approximation* (CSA) problem extends traditional subspace approximation by enforcing constraints on the projection matrix, capturing problems like *partition-constrained low-rank approximation, k-means clustering, and projected non-negative matrix factorization (PNMF)*, among others. A key motivation for CSA arises in fairness-sensitive applications, where partition constraints can enforce group-level representation—for

example, by requiring the subspace to be spanned by one representative from each class—thus ensuring fair summaries that reflect the diverse subpopulations in the dataset. However, due to its combinatorial and non-convex nature, CSA is computationally challenging, often requiring exponential time in the worst case. To address this, we develop a *coreset-guess-solve framework* that provides efficient $(1+\varepsilon)$-multiplicative or ε-additive approximations across various CSA settings. This framework establishes theoretical guarantees, making CSA more practical for large-scale data analysis. See §1.4 for a more elaborate introduction and theorem statements.

3. **Explainability:** Clustering is a fundamental problem in optimization, machine learning, and algorithm design, with k-medians and k-means as two of the most prominent methods. However, traditional clustering often lacks interpretability. *Explainable clustering* enhances transparency by structuring cluster assignments as decision trees with axis-aligned threshold cuts, enabling intuitive explanations while introducing a trade-off—the **"price of explainability"**—between interpretability and clustering quality.

   Prior work developed greedy and randomized thresholding algorithms to make clusterings explainable, improving approximation guarantees for k-medians and k-means. However, fundamental gaps remained in understanding the limits of explainability and the optimality of existing approaches. We provide a tight analysis of the Random Thresholds algorithm for k-medians, proving its price of explainability is at most $\ln k(1+o(1))$ and establishing a matching lower bound of $\ln k(1-o(1))$ for any algorithm. For k-means, we improve the upper bound from $O(k \ln k)$ to $O(k \ln \ln k)$, significantly narrowing the gap with the $\Omega(k)$ lower bound. Finally, we prove that the explainable k-medians problem cannot be approximated better than $O(\ln k)$ unless $\mathsf{P} = \mathsf{NP}$, resolving key questions on its approximability. See §1.5 for a more elaborate introduction and theorem statements.

4. **Robustness:** Algorithmic robustness is critical in scenarios where input data is noisy, incomplete, or imprecisely specified. This challenge is particularly pronounced in real-world applications such as preference aggregation, crowd-sourcing, and human-in-the-loop systems, where algorithms may not have access to exact numerical values but only to pairwise comparisons between alternatives. Motivated by this, we study combinatorial optimization under a comparison oracle model, where the goal is to find optimal or near-optimal solutions using only comparison-based access to the objective function. While this model significantly restricts the information available to the algorithm, we show that, for several classic problems—such as minimum cut, spanning trees, matchings, and shortest paths—comparison access suffices to recover optimal solutions efficiently. Our results provide the first general framework for solving a broad class of combinatorial problems in this robust comparison-based setting, closing the gap between comparison and value-based optimization for these problems. See §1.6 for a more elaborate introduction and theorem statements.

These five problems capture key aspects of fairness, diversity, explainability, and robustness in algorithmic decision-making—ensuring diverse selection, fair allocation, fair representation, interpretable clustering, and robust optimization under limited information. This work develops improved approximation algorithms using tools from discrete and continuous optimization, con-

vex geometry, and probabilistic methods. The results have broad implications across machine learning, economics, large-scale data analysis, and human-in-the-loop systems, reinforcing the need for equitable, interpretable, and reliable algorithmic frameworks in modern computational settings.

1.2 Diversity: Determinant Maximization

In an instance of a determinant maximization problem, we are given a collection of vectors $U = \{v_1, \ldots, v_n\} \subset \mathbb{R}^d$, and the goal is to pick a subset $S \subseteq U$ of the given vectors to maximize the determinant of the matrix $\sum_{i \in S} v_i v_i^\top$. Additionally, we may require that the set S of picked vectors must satisfy some combinatorial constraints such as cardinality constraint ($|S| \leq k$) or matroid constraint (S is a basis of a matroid defined on the vectors).

Determinant maximization problem gives a general framework that models problems arising in diverse fields such as statistics [155], convex geometry [116], fair allocations [6], combinatorics [8], spectral graph theory [149], network design and random processes [119]. Apart from its modeling strength, from a technical perspective, determinant maximization has brought interesting connections between areas such as combinatorial optimization, convex analysis, geometry of polynomials, graph sparsification and complexity of permanent and other counting problems [3, 5, 6, 116].

What follows is a summary of our contributions. The results outlined below offer a high-level overview of the algorithmic techniques and approximation guarantees we obtain. The full technical development, including formal statements and detailed proofs, is provided in Chapter 2.

1.2.1 Our Results and Contributions

In this work, we introduce new combinatorial methods for determinant maximization under a matroid constraint and give a $d^{O(d)}$-deterministic approximation algorithm. While previous works have used a convex programming approach and the theory of stable polynomials, our approach builds on the classical matroid intersection algorithm. Our first result focuses on the case when the rank of the matroid is exactly d, i.e., the output solution will contain precisely d vectors.

Theorem 1.2.1 (Rank equals dimension)**.** *There is a polynomial-time algorithm which, given a collection of vectors $v_1, \ldots, v_n \in \mathbb{R}^d$ and a matroid $\mathcal{M} = ([n], \mathcal{I})$ of rank d, returns a set $S \in \mathcal{I}$ such that*

$$\det\left(\sum_{i \in S} v_i v_i^\top\right) = \Omega\left(\frac{1}{d^{O(d)}}\right) \max_{S^* \in \mathcal{I}} \det\left(\sum_{i \in S^*} v_i v_i^\top\right).$$

Our results improve the $e^{O(d^2)}$-approximation algorithm that relies on the $e^{O(d)}$-estimation algorithm [5, 8, 131]. Our algorithm builds on the matroid intersection algorithm and is an iterative algorithm that starts at any feasible solution and improves the objective in each step. To maintain feasibility in the matroid constraint, each step of the algorithm is an exchange of multiple elements as found by an alternating cycle of an appropriately defined exchange graph.

Result for $r \leq d$. We also generalize the result when the rank r of the matroid is at most d. Observe that the solution matrix $\sum_{i \in S} v_i v_i^\top$ is a $d \times d$ matrix of rank at most r and, therefore, the appropriate objective to consider is the product of its largest r eigenvalues, or equivalently, the elementary symmetric function of order r of its eigenvalues. Let $\mathrm{sym}_r(M)$ be the r^{th} elementary symmetric function of the eigenvalues of the $d \times d$ matrix M. Thus, our objective is to maximize $\mathrm{sym}_r\left(\sum_{i \in S} v_i v_i^\top\right)$.

Theorem 1.2.2 (Rank less than dimension)**.** *There is a polynomial-time algorithm which, given a collection of vectors $v_1, \ldots, v_n \in \mathbb{R}^d$ and a matroid $\mathcal{M} = ([n], \mathcal{I})$ of rank $r \leq d$, returns a set $S \in \mathcal{I}$ such that*

$$\mathrm{sym}_r\left(\sum_{i \in S} v_i v_i^\top\right) = \Omega\left(\frac{1}{r^{O(r)}}\right) \max_{S^* \in \mathcal{I}} \mathrm{sym}_r\left(\sum_{i \in S^*} v_i v_i^\top\right).$$

This again improves the best bound of $e^{O(r^2)}$-approximation algorithm based on $e^{O(r)}$-approximate estimation algorithms.

Result for $r \geq d$ Finally, we give an analogous result when the number r of vectors selected is larger than d. Here we require some sparsity properties of the convex programming relaxation, but the key algorithmic ideas are the same as the previous cases.

Theorem 1.2.3 (Rank more than dimension)**.** *There is a polynomial-time algorithm which, given a collection of vectors $v_1, \ldots, v_n \in \mathbb{R}^d$ and a matroid $\mathcal{M} = ([n], \mathcal{I})$ of rank $r \geq d$, returns a set $S \in \mathcal{I}$ such that*

$$\det\left(\sum_{i \in S} v_i v_i^\top\right) = \Omega\left(\frac{1}{d^{O(d)}}\right) \max_{S^* \in \mathcal{I}} \det\left(\sum_{i \in S^*} v_i v_i^\top\right).$$

This matches the $d^{O(d)}$-approximate estimation algorithm [131], which only gives an estimate of the optimum value and results in $d^{O(d^2)}$-approximation algorithm.

1.3 Fairness: Nash Social Welfare Maximization

In an instance of the weighted Nash Social Welfare problem, we are given a set of m indivisible items $\mathcal{G}$, and a set of n agents, $\mathcal{A}$. Every agent $i \in \mathcal{A}$ has a weight $w_i \geq 0$ and an additive valuation function $\mathbf{v}_i : 2^{\mathcal{G}} \to \mathbb{R}_{\geq 0}$. Let $v_{ij} := \mathbf{v}_i(\{j\})$. The goal is to find an assignment of items, $\sigma : \mathcal{G} \to \mathcal{A}$ so that the following welfare function is maximized:

$$\prod_{i \in \mathcal{A}} \left(\sum_{j \in \sigma^{-1}(i)} v_{ij}\right)^{w_i}.$$

For ease of notation, we will work with the log objective and denote

$$\mathrm{NSW}(\sigma) = \sum_{i \in \mathcal{A}} w_i \log\left(\sum_{j \in \sigma^{-1}(i)} v_{ij}\right) \tag{1.1}$$

and $\mathrm{OPT} = \max_{\sigma:\mathcal{G}\to\mathcal{A}} \mathrm{NSW}(\sigma)$ denote the optimal log objective. The much studied case is the symmetric Nash social welfare problem in which $\mathbf{w} = \mathbf{u}$, where $u_i = \frac{1}{n}$ for each $i \in \mathcal{A}$ and the objective is the geometric mean of agents' valuations.

Fair and efficient division of resources among agents is a fundamental problem arising in various fields [19, 31, 32, 160, 161, 190]. While there are many social welfare functions which can be used to evaluate the efficacy of an assignment of goods to the agents, the Nash Social Welfare function is well-known to interpolate between fairness and overall utility. The unweighted Nash Social Welfare function first appeared as the solution of an arbitration scheme proposed by Nash for two-person bargaining games that was generalized to multiple players [115, 144]. Since then, it has been widely used in numerous fields to model resource allocation problems. An attractive feature of the objective is that it is invariant under scaling by any of the agent's valuations and therefore each of the agents can specify their utility in their own units (see [43] for a detailed treatment). While the theory of Nash Social Welfare objective was initially developed for divisible items, more recently it has been applied in the context of indivisible items. We refer the reader to [42] for a comprehensive overview of the problem in the latter setting. Indeed, optimizing the Nash Social Welfare objective also implies notions of fairness such as *envy free* allocation in an approximate sense [20, 42].

The Nash Social Welfare function with weights (also referred to as asymmetric or non-symmetric Nash Social Welfare) was first studied in the seventies [100, 112] in the context of two person bargaining games. For example, in the bargaining context, it allows different agents to have different weights. This flexibility has made the problem arise in many different domains, including bargaining theory [43, 123], water resource allocation [82, 106], and climate agreements [191]. In the context of indivisible goods, the study of this problem has been much more recent [85, 86, 87]. In this work, we aim to shed light on this problem, especially, with a focus on mathematical programming relaxations for the problem.

What follows is a summary of our contributions. The results outlined below offer a high-level overview of the algorithmic techniques and approximation guarantees we obtain. The full technical development, including formal statements and detailed proofs, is provided in Chapter 4.

1.3.1 Our Results and Contributions

We present a

$$\exp\left(2\log 2 + \frac{1}{2e} + \mathrm{KL}(\boldsymbol{w} \,\|\, \boldsymbol{u})\right) \approx 4.81 \cdot \exp\left(\log n - \sum_{i=1}^{n} w_i \log \frac{1}{w_i}\right)$$

approximation algorithm for the weighted Nash Social Welfare problem with additive valuations, which improves upon the previous approximation factor of $O(n \cdot w_{\max})$. When all the weights are the same, this gives a constant factor approximation. Our algorithm builds on and extends a convex programming relaxation for the symmetric variant of Nash Social Welfare presented in [9, 60, 61]. In the theorem, we state the guarantee in log objective and therefore, the guarantee becomes an additive one.

Theorem 1.3.1. *Let $(\mathcal{A}, \mathcal{G}, \mathbf{v}, \mathbf{w})$ be an instance of the weighted Nash Social Welfare problem with $\sum_{i\in\mathcal{A}} w_i = 1$ and $|\mathcal{A}| = n$ agents. There exists a polynomial time algorithm (Algorithm 5) that, given $(\mathcal{A}, \mathcal{G}, \mathbf{v}, \mathbf{w})$, returns an assignment $\sigma : \mathcal{G} \to \mathcal{A}$ such that*

$$\mathrm{NSW}(\sigma) \geq \mathrm{OPT} - 2\log 2 - \frac{1}{2e} - 2 \cdot D_{\mathrm{KL}}(\mathbf{w} \,\|\, \mathbf{u}),$$

where OPT *is the optimal log-objective for the instance and $D_{\mathrm{KL}}(\mathbf{w}\|\mathbf{u}) = \log n - \sum_{i\in\mathcal{A}} w_i \log \frac{1}{w_i}$.*

Observe that the KL-divergence term $D_{\mathrm{KL}}(\mathbf{w} \,\|\, \mathbf{u}) = \left(\log n - \sum_{i\in\mathcal{A}} w_i \log \frac{1}{w_i}\right)$ is always upper bounded by $nw_{\max}$ which is exactly the guarantee of previous work [87]. In many settings, the term $D_{\mathrm{KL}}(\mathbf{w} \,\|\, \mathbf{u})$ can be significantly smaller than $nw_{\max}$; for example, consider the setting where $w_1 = \frac{1}{\log n}$ and $w_i = \frac{1}{n-1}(1 - \frac{1}{\log n})$ for $i = 2, \ldots, n$, i.e., one agent has significantly higher weight than the others. In this case, our results imply $O(1)$-approximation while previous results imply $O(\frac{n}{\log n})$-approximation.

Our algorithm relies on two mathematical programming relaxations (CVX-Weighted) and (NCVX-Weighted) both of which generalize the convex relaxation for the unweighted version [7, 60, 61]. One of the relaxation is non-convex but retains a lot of structural insights obtained for the convex relaxation in the unweighted version. We show that the same rounding algorithm as in the unweighted version [60] gives a $O(1)$-approximation for the asymmetric version applied to a fractional solution of the non-convex program. While the rounding algorithm is the same, the analysis requires new ideas as many of the interpretations via market equilibrium in the unweighted case are no longer present in the weighted version. Unfortunately, due to its non-convex nature, we cannot solve this relaxation to optimality even though it can be rounded efficiently. This is where the second mathematical programming relaxation comes to the rescue. This relaxation is convex and thus can be solved efficiently. We solve the convex relaxation, use it as an initial point to obtain a first order stationary solution to the non-convex relaxation that we round to an integral solution. The approximation factor of $D_{\mathrm{KL}}(\mathbf{w} \,\|\, \mathbf{u})$ arrives due to the difference in objective values of these two programs.

1.4 Fairness: Constrained Subspace Approximation

In subspace approximation, given a set of n points $\{a_1, \ldots, a_n\} \subset \mathbb{R}^d$ and a rank parameter k, the goal is to find a rank-k projection matrix $\boldsymbol{P}$ that minimizes the projection costs $\|a_i - \boldsymbol{P}a_i\|_2$, aggregated over all $i \in [n]$. The choice of aggregation function leads to different well-studied formulations. In the ℓ_p-subspace approximation problem, the objective is to minimize $(\sum_{i=1}^n \|a_i - \boldsymbol{P}a_i\|_2^p)$. Formally, letting $\boldsymbol{A}$ be the $d \times n$ matrix with columns $a_1, \ldots, a_n$, the ℓ_p-subspace approximation problem seeks a rank-k projection matrix $\boldsymbol{P} \in \mathbb{R}^{d\times d}$ that minimizes

$$\|\boldsymbol{A} - \boldsymbol{P}\boldsymbol{A}\|_{2,p}^p := \sum_{i=1}^n \|a_i - \boldsymbol{P}a_i\|_2^p.$$

The ℓ_p-subspace approximation captures several classical problems for different values of p: the

median hyperplane problem ($p = 1$), *principal component analysis (PCA)* ($p = 2$), and the *center hyperplane problem* ($p = \infty$).

In the most general setting of the *constrained ℓ_p-subspace approximation* problem, we are additionally given a collection $\mathcal{S}$ of rank-k projection matrices (specified either explicitly or implicitly) and the goal is to find a projection matrix $\boldsymbol{P} \in \mathcal{S}$ that minimizes the objective:

$$\min_{\boldsymbol{P} \in \mathcal{S}} : \|\boldsymbol{A} - \boldsymbol{P}\boldsymbol{A}\|_{2,p}^p. \tag{CSA}$$

We next present an overview of the applications motivating CSA and our core algorithmic contributions. The results summarized here provide a high-level perspective on the techniques and guarantees we obtain. A full technical treatment—including precise definitions, algorithmic frameworks, and detailed proofs—can be found in Chapter 5.

1.4.1 Applications and Our results

This section contains a small subset of the applications and results from [25].

Projective Non-negative Matrix Factorization and k-means clustering

In projective non-negative matrix factorization, the basis matrix $\boldsymbol{U} \in \mathbb{R}^{d \times k}$ is constrained to have non-negative entries. More formally, the mathematical program formulation for Projective Non-negative Matrix Factorization (NMF) is

$$\min : \|\boldsymbol{A} - \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{A}\|_F^2 \tag{NMF}$$

$$\boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}_k,\ \boldsymbol{U} \in \mathbb{R}_{\geq 0}^{d \times k}. \tag{1.2}$$

Theorem 1.4.1 (Additive approximation for NMF)**.** *Given an instance $\boldsymbol{A} \in \mathbb{R}^{d \times n}$ of Non-negative matrix factorization, there is an algorithm that computes a $\boldsymbol{U} \in \mathbb{R}_{\geq 0}^{d \times k}$, $\boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}_k$ such that*

$$\|\boldsymbol{A} - \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{A}\|_F^2 \leq (1 + \varepsilon) \cdot \mathrm{OPT} + O(\delta \cdot \|\boldsymbol{A}\|_F^2)$$

in time $O(dk^2/\varepsilon) \cdot (1/\delta)^{O(k^2/\varepsilon)}$. For any $0 < \delta < 1$.

Theorem 1.4.2 (Multiplicative approximation for NMF)**.** *Given an instance $\boldsymbol{A} \in \mathbb{R}^{d \times n}$ of Non-negative matrix factorization with integer entries of absolute value at most γ in $\boldsymbol{A}$, there is an algorithm that computes a $\boldsymbol{U} \in \mathbb{R}_{\geq 0}^{d \times k}$, $\boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}_k$ such that*

$$\|\boldsymbol{A} - \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{A}\|_F^2 \leq (1 + \varepsilon) \cdot \mathrm{OPT}$$

in time $(nd\gamma/\varepsilon)^{O(k^3/\varepsilon)}$.

In the k-means problem, we are given a collection of data points $a_1, \ldots, a_n \in \mathbb{R}^d$. The objective is to find k-centers $c_1, \ldots, c_k \in \mathbb{R}^d$ and an assignment $\pi : [n] \to [k]$ that minimizes:

$$\sum_{i=1}^{n} \|a_i - c_{\pi(i)}\|_2^2. \tag{k-means}$$

The original formulation can be re-cast as the following special case of NMF with additional constraints in the following way :

$$\begin{aligned} &\min : \|\boldsymbol{A} - \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{A}\|_F^2 && (k\text{-means-CSA}) \\ &\boldsymbol{U}_{i,j} = 1/\sqrt{\|U_{\cdot,j}\|_0} \quad \forall i \in [n],\, j \in [k]. \end{aligned}$$

Theorem 1.4.3. *Given an instance $\boldsymbol{A} \in \mathbb{R}^{n\times d}$ of k-means, there is an algorithm that computes a $(1+\varepsilon)$-approximate solution to k-means in $O(nnz(\boldsymbol{A}) + 2^{\widetilde{O}(k/\varepsilon)} + n^{o(1)})$ time.*

Partition Constrained ℓ_p-Subspace Approximation

Given a set of subspaces $S_1, \ldots, S_k \subseteq \mathbb{R}^d$, select a vector $v_i \in S_i$ for $i \in [k]$ in order to minimize $\sum_{i\in[n]} \| \operatorname{proj}^{\perp}_{\text{span}(v_1,\ldots,v_k)}(a_i)\|_2^p$, where $p \geq 1$ is a given parameter. A more compact formulation is:

$$\begin{aligned} &\min : \|\boldsymbol{A} - \boldsymbol{V}\boldsymbol{C}\|_{2,p}^p && (\text{PC-}\ell_p\text{-SA-fac}) \\ &\quad \boldsymbol{V} = [v_1, \ldots, v_k] \\ &\quad v_i \in S_i \quad \forall i \in [k]. \end{aligned}$$

The PC-ℓ_p-SA-fac can be formulated equivalently as

$$\begin{aligned} &\min : \|\boldsymbol{A} - \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{A}\|_{2,p}^p && (\text{PC-}\ell_p\text{-SA}) \\ &\quad \boldsymbol{U} \text{ is an orthogonal basis for Span}(v_1, v_2, \ldots, v_k) \\ &\quad v_i \in S_i \quad \forall i \in [k]. \end{aligned}$$

Assume each input entry of $\boldsymbol{A}$ has bit complexity at most H.

Theorem 1.4.4 (Additive approximation)**.** *There exists an algorithm for PC-ℓ_p-subspace approximation with runtime $(\kappa/\varepsilon)^{O(kr)} \cdot \operatorname{poly}(n, d, k/\varepsilon) \cdot H$ which returns a solution with additive error at most $O(\varepsilon p) \cdot \|A\|_{p,2}^p$, where κ is the condition number of an optimal solution $\boldsymbol{V}^* = [v_1^*, v_2^*, \ldots, v_k^*]$ for the PC-subspace approximation problem PC-ℓ_p-SA-fac.*

For the special case of $p = 2$, it turns out that we can obtain a $(1+\varepsilon)$-multiplicative approximation, using additional tools like polynomial system solvers.

Theorem 1.4.5 (Multiplicative approximation)**.** *Let $(\boldsymbol{A}, \mathcal{S})$ be an instance of PC-ℓ_2-subspace approximation. If there exists an (approximately) optimal solution with bit complexity poly(n, H), there exists an algorithm that runs in time $n^{O(k^2/\varepsilon)} \cdot poly(H)$ and outputs a solution whose objective value is within a $(1+\varepsilon)$ factor of the optimum objective value. We denote $s = \sum_{j=1}^k s_j$ and $s_j = dim(S_j)$; n for this result can be set to $\max(s, d, k/\varepsilon)$.*

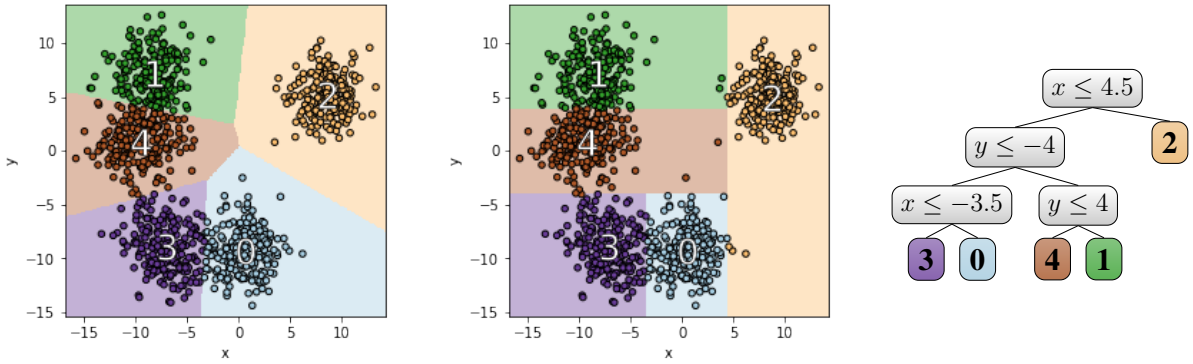
1.5 Explainability: Explainable Clustering

An explainable clustering is one where the clusters are defined by a decision tree with axis-aligned threshold cuts (see, e.g., Figure 1.1). Each internal node in the tree splits the data along a

single feature using a threshold, and each leaf represents a distinct cluster. This structure ensures that a point's cluster assignment follows a sequence of simple, interpretable decisions. This notion of explainability for clustering was introduced by Dasgupta et al. [62]

Clustering is a central topic in optimization, machine learning, and algorithm design, with k-medians and k-means being two of the most prominent examples. In recent years, mainly motivated by the impressive but still mysterious advances in machine learning, there has been an increased interest in the transparency and in the explainability of solutions.

To motivate the concept of explainability, consider the task of clustering n points in $\mathbb{R}^d$ into k clusters. If we solve k-means, the clusters are in general given by a Voronoi diagram where each cluster/cell is defined by the intersection of hyperplanes. Each cluster may be defined using up to $k-1$ hyperplanes, each one of them possibly depending on all d dimensions with arbitrary coefficients. Since the dimensions typically correspond to features (e.g., "age", "weight", and "height" are natural features in a dataset of people), arbitrary linear combinations of these features may be difficult to interpret. To achieve more explainable solutions, we may need to restrict our algorithms to find clusters with simpler descriptions.



(a) Optimal 5-means clusters *(b) Tree based 5-means clusters* *(c) Threshold tree*

Figure 1.1: Example from [62]. The optimal 5-means clustering (left) uses combinations of both features. The explainable clustering (middle) uses axis-aligned rectangles summarized by the threshold tree (right).

Explainability is thus a very desirable and appealing property, but the best explainable clustering may have cost much higher than the cost of the best unrestricted clusterings. This trade-off is captured by the *price of explainability*: the loss in cost/quality if we restrict ourselves to explainable clusterings.

What follows is a summary of our contributions to explainable clustering. The results outlined below provide a high-level overview of our techniques and bounds. The full technical development is provided in Chapter 6.

1.5.1 Our Results and Contributions

Our main results on the price of explainability are (a) to settle this conjecture in the affirmative (i.e., to give a *tight* analysis of the Random Thresholds algorithm), and (b) to show that its price

of explainability of $1 + H_{k-1} = (1 + o(1)) \ln k$ is not only asymptotically correct, but also *tight up to lower order terms*: we cannot do much better regardless of the algorithm.

Theorem 1.5.1 (Upper bound for k-medians)**.** *The price of explainability for k-medians is at most $1 + H_{k-1}$. Specifically, given any reference k-medians clustering, the* Random Thresholds *algorithm outputs an explainable clustering with expected cost at most $1 + H_{k-1}$ times the cost of the reference clustering.*

Theorem 1.5.2 (Lower Bound for k-medians)**.** *There exist instances of k-medians for which any explainable clustering has cost at least $(1 - o(1)) \ln k$ times the cost of the optimal k-medians clustering.*

These results resolve the performance of the Random Thresholds algorithm and the price of explainability for k-medians.

For k-means, we are unable to settle the price of explainability completely, but we make significant progress in closing the gap between known upper and lower bounds. Here, the best upper bound before our work was $O(k \ln k)$ [73] (see also [45] for better guarantees when the input is low-dimensional). Moreover, we know instances where any single threshold cut increases the cost of the clustering by a factor $\Omega(k)$ (see, e.g., [83]), and hence the price of explainability of k-means is at least $\Omega(k)$.

It is tempting to guess that the $O(k \ln k)$ guarantee in [73] is tight, for the following reason. The first lower bound $\Omega(\ln k)$ for k-means in [62] is obtained by arguing that (i) a single threshold cut increases the cost by at least that of the reference clustering and (ii) a threshold tree has height $\Omega(\ln k)$, and so the total cost increases by a constant $\Omega(\ln k)$ times. Since we have examples where any single cut increases the cost by $\Omega(k)$, it is reasonable to hope for more complex instances to combine the two sources of hardness, and lose a $\Omega(k) \cdot \Omega(\ln k)$ factor. However, we prove that this is *not* the case and give an improved upper bound:

Theorem 1.5.3 (Upper bound for k-means)**.** *The price of explainability for k-means is at most $O(k \ln \ln k)$. Specifically, given any reference k-means clustering, there exists an algorithm that outputs an explainable clustering with expected cost at most $O(k \ln \ln k)$ times the reference cost.*

Hence the price of explainability for k-means lies between $\Omega(k)$ and $O(k \ln \ln k)$. We leave the tight answer as an intriguing open problem. In particular, we conjecture that the lower bound is tight and that it is achieved by the k-means variant of the Random Thresholds algorithm.

Our final contribution is to *study the approximability of explainable clustering*. So far, the literature has mostly focused on settling the price of explainability [45, 62, 73, 83, 121, 133] and its behavior in a bi-criteria setting [134] where the explainable clustering is allowed to form more than k clusters. These algorithms give upper bounds on the approximability of explainable clustering since they are all efficient, and the cost of an optimal unconstrained clustering is a valid lower bound on the best explainable one. Recent work of [18, 120] asked the question: *how well can we approximate the best explainable clustering?* They showed that the problem is APX-hard, but left open the question of whether the problem can be approximated better. Resolving this natural question positively would have the advantage of finding good explainable clusterings for those instances that do admit such clusterings, which is often the experience for

more practical instances. Our result shows a surprising hardness for the k-medians and k-means problem.

Theorem 1.5.4 (Approximability)**.** *The explainable k-medians and k-means problems are hard to approximate better than $(1/2 - o(1)) \ln k$, unless $\mathsf{P} = \mathsf{NP}$.*

These results show that we cannot approximate k-medians much better than its price of explainability (unless $\mathsf{P} = \mathsf{NP}$); the approximability for k-means remains tantalizingly open.

1.6 Robustness: Combinatorial Optimization using Comparison Oracles

Consider the following general optimization problem: we are given a ground set U of n elements, a family $\mathcal{F} \subseteq 2^U$ of feasible subsets, and an *unknown* objective function $f : 2^U \to \mathbb{R}_+$. At each step, we may query a *comparison oracle* which, given any two feasible sets $S, T \in \mathcal{F}$, reveals only the sign of $f(S) - f(T)$. That is, the algorithm learns whether $f(S) > f(T)$, $f(S) < f(T)$, or $f(S) = f(T)$, but not the actual values. The goal is to identify an (approximately) optimal set $S^* \in \mathcal{F}$ using as few comparison queries as possible.

In the worst case, one could always find the optimum by performing $|\mathcal{F}| - 1$ comparisons via brute-force search. However, this is often computationally infeasible since $\mathcal{F}$ may be exponentially large. This raises the central question of this chapter:

> *Can we find (approximately) optimal solutions using only a polynomial number of comparison queries—especially for linear objective functions and combinatorially structured families $\mathcal{F}$?*

To make this concrete, consider the classical minimum cut problem. We are given a simple undirected graph $G = (V, E, w)$ with nonnegative edge weights, but the algorithm has no access to the edge set or weights. It only knows the vertex set V. At each step, it may submit two non-trivial subsets $A, B \subset V$ to a cut-comparison oracle, which reports whether the cut induced by A is smaller, larger, or equal in weight to the one induced by B. That is, the oracle returns the sign of $f(A) - f(B)$, where $f(S) = \sum_{e \in \partial S} w_e$ and ∂S denotes the cut edges of S.

This setting is a natural instance of the general comparison model, with $U = V$, $\mathcal{F}$ being the set of non-trivial cuts, and f being a modular function over the cut edges.

There are several compelling motivations for studying this model:

- **Real-world uncertainty:** In many real-world applications—such as recommendation systems, fair allocation, or crowdsourced decision-making—agents may not assign precise numerical utilities to outcomes, but can often express reliable *ordinal* preferences between alternatives. Comparison oracles capture this natural mode of feedback.
- **Robustness:** Since only the order of values matters, comparison-based algorithms are invariant under monotone transformations of the objective function, making them robust to scaling, normalization, or reparameterization of utilities.

- **Minimal information:** Comparison queries reveal strictly less information than value queries. Studying what can still be computed efficiently under such constraints sheds light on the *query complexity* of combinatorial optimization problems.

These considerations motivate a fundamental question: even when exact utilities are inaccessible, can we still solve classical optimization problems efficiently using only pairwise comparisons? As a case in point, consider the minimum cut problem. Prior work shows that it can be solved using a linear number of *value* queries—each returning the cost of a cut. Since a comparison query can be simulated using two value queries, this highlights the comparison model as a strictly weaker alternative. Understanding its power and limitations—particularly for problems like min-cut—offers insight into the minimal information needed for efficient optimization. What follows is a summary of our contributions to combinatorial optimization using comparison oracles. The results outlined below offer a high-level overview of the algorithmic techniques and bounds we establish across several classical problems. A complete technical treatment can be found in Chapter 7.

1.6.1 Our Results and Techniques

We study the comparison-based model for linear objectives of the form $f(S) = \sum_{e \in S} w_e$, where the weights w_e are unknown. The feasible sets $\mathcal{F} \subseteq 2^U$ are problem-specific, and the algorithm may only compare $f(S)$ and $f(T)$ for any $S, T \in \mathcal{F}$.

Graph Cuts. We first consider the classical minimum cut problem in unweighted undirected graphs, where feasible solutions correspond to non-trivial vertex cuts.

Theorem 1.6.1 (Minimum cut)**.** *There is a randomized algorithm that computes the exact minimum cut of a simple graph G with high probability using $\widetilde{O}(n)$ cut comparison queries and $\widetilde{O}(n^2)$ time.*

Theorem 1.6.2 (Graph recovery)**.** *A simple unweighted graph $G \notin \{K_2, \bar{K}_2, K_3, \bar{K}_3\}$ can be recovered using $O(\min\{(m+n)\log n, n^2\})$ cut comparison queries and in $\widetilde{O}(n^2)$ time.*

Matroid Bases, Matroid Intersections, and Paths. We extend our techniques to other classical structures: matroid bases, matroid intersections, and s-t paths.

Theorem 1.6.3 (Matroid Bases)**.** *There is an algorithm outputs the minimum-weight basis of a matroid on n elements using $O(n \log n)$ comparison queries in $\widetilde{O}(n^2)$ time.*

Theorem 1.6.4 (Matroid Intersection)**.** *Let $\mathcal{M}_1 = (U, \mathcal{I}_1)$ and $\mathcal{M}_2 = (U, \mathcal{I}_2)$ be two matroids defined on a ground set U containing n elements. Then, there is an algorithm that outputs the minimum-weight set that is in both $\mathcal{I}_1, \mathcal{I}_2$ using $O(n^4)$ comparison queries in $O(n^4)$ time.*

Theorem 1.6.5 (s-t walks)**.** *There is an algorithm that finds the minimum-length s-t path in a graph G (or a negative cycle, if one exists) using $O(n^3)$ s-t walk comparisons and $O(n^3)$ time.*

General Linear Optimization. We also give general results for arbitrary families $\mathcal{F} \subseteq 2^U$ and geometric optimization problems:

Theorem 1.6.6 (Boolean Linear Optimization)**.** *For any family $\mathcal{F} \subseteq 2^U$ and unknown weight function $w : U \to \mathbb{R}$, we can solve $\arg\min_{S\in\mathcal{F}} \sum_{e\in S} w_e$ using $O(n \log n \cdot \log |\mathcal{F}|) = \widetilde{O}(n^2)$ comparison queries, where $n = |U|$.*

Theorem 1.6.7 (General Linear Optimization)**.** *There is an algorithm that, for any point set $\mathcal{P} \subseteq \mathbb{R}^d$ with conic dimension k and unknown weights $w \in \mathbb{R}^d$, returns the minimizer $x^* = \arg\min_{x\in\mathcal{P}} \langle w, x\rangle$ using at most $O(k \log k \log |\mathcal{P}|)$ comparisons.*

These results establish that a wide class of combinatorial and geometric optimization problems admit efficient algorithms even in this restrictive comparison-based model. See Chapter 7 for full proofs, technical insights, and limitations of this framework.

1.7 Technical Unification

The five problems investigated in this thesis, while spanning the distinct goals of promoting diversity, fairness, explainability, and robustness, share a deep technical connection: they each require optimizing a continuous objective function—such as volume, projection error, or welfare—under constraints that are fundamentally discrete, combinatorial, or informational in nature. This interplay between the continuous and the discrete is the central challenge that this work addresses. The solutions presented, though seemingly distinct, can be understood as different facets of a single algorithmic philosophy: leveraging the structure of one domain to overcome computational barriers in the other. This philosophy manifests in two primary, complementary strategies, along with a third meta-strategy for quantifying their interaction.

The first strategy is to use continuous relaxations to guide combinatorial search. In this approach, a computationally hard, discrete problem is mapped into a more tractable continuous landscape. This is the core of the work on fair allocation, where novel convex and non-convex programs are designed to approximate the Nash Social Welfare objective. The resulting fractional solution, which lies in a continuous space, is then carefully rounded back into a high-quality discrete assignment of indivisible goods. Similarly, for constrained subspace approximation, a discrete coreset is used to create a simplified continuous problem. Its complexity is then managed by guessing a small number of key continuous coefficients, transforming an intractable non-convex problem into a series of efficiently solvable convex ones.

The second, converse strategy is to use combinatorial structures to navigate a complex continuous objective. Here, instead of relaxing the problem, a discrete scaffold is built to optimize the continuous function directly, often when its global structure is inaccessible or non-convex. The work on determinant maximization exemplifies this, where a combinatorial "exchange graph" derived from matroid theory guides a local search algorithm toward a globally-near-optimal solution for the continuous volume objective, all while staying within the discrete confines of a matroid constraint. This strategy is taken to its extreme in the work on combinatorial optimization with comparison oracles, where the continuous objective function is entirely hidden. It is shown that discrete, ordinal queries (comparisons) are sufficient to reconstruct the essential combinatorial structure of the problem allowing for the discovery of an optimal solution without ever learning its value.

Finally, this thesis also directly confronts the cost of this interplay by quantifying the fundamental trade-off of imposing discrete structure onto a continuous problem. This is the focus of the work on explainable clustering, which analyzes the "price of explainability". Here, one domain is not used to solve the other; rather, a precise probabilistic analysis of a randomized combinatorial algorithm is employed to understand the inherent cost of forcing a discrete, tree-based structure onto a continuous geometric clustering problem, thereby making its solutions interpretable.

Together, these strategies form a powerful and unified approach for tackling modern algorithmic challenges. They demonstrate a robust synthesis of continuous and discrete techniques, providing a principled framework for algorithm design at the frontier where societal values like fairness and explainability impose structures that defy classical optimization.

1.8 Organization and Credits

The remainder of this thesis is divided into four parts, each corresponding to one of the four facets of my research: Diversity (Part I), Fairness (Part II), Explainability (Part III), and Robustness (Part IV).

Part I focuses on diversity and contains two chapters. *Chapter 2* on determinant maximization is based on joint work with Adam Brown, Aditi Laddha, Mohit Singh, and Prasad Tetali, published in [36] and [35].

Chapter 3 addresses the computationally challenging, #P-complete problem of computing the permanent of a non-negative matrix. Due to the permanent's computational complexity, a significant research effort has been focused on finding efficient upper bounds. In this chapter, we introduce a new pathway for establishing such bounds by systematically adapting powerful tools from determinant theory. We present a novel permanental analogue of the Schur's formula for determinants, which is built upon a newly defined permanental inverse. Building on this, we introduce the **permanent process**, an iterative, deterministic procedure analogous to Gaussian elimination that yields constructive and algorithmically computable upper bounds on the permanent. The inequalities we develop were derived by generalizing Lemma 2.5.1 that was used in the analysis of Chapter 2. We chose not to detail this work in the main introduction to avoid disrupting the narrative arc of diversity and determinant maximization. Beyond this application, our developments are of independent interest and contribute to the large body of work on permanent upper bounds. This chapter is based on recent unpublished work with Aditi Laddha.

Part II addresses fairness through two chapters. *Chapter 4* on weighted Nash Social Welfare maximization is based on a publication and a forthcoming journal article, which are joint work with Adam Brown, Aditi Laddha, and Mohit Singh [34]. *Chapter 5* on constrained subspace approximation is based on joint work with Aditya Bhaskara, Sepideh Mahabadi, Ali Vakilian, and David P. Woodruff, published in [25].

Part III contains one chapter on the price of explainability in clustering. *Chapter 6* is based on the publication [92], which is joint work with Anupam Gupta, Ola Svensson, and Rachel Yuan.

The final part, Part IV, presents one chapter on combinatorial optimization using comparison

oracles. *Chapter 7* is based on recent work (currently under submission) with Vincent Cohen-Addad, Tommaso d'Orsi, Anupam Gupta, Guru Guruganesh, Euiwoong Lee, Renato Paes Leme, Debmalya Panigrahi, Jon Schneider, and David P. Woodruff.

All co-authors are listed in alphabetical order by last name.

Part I

Diversity

Chapter 2

Determinant Maximization

2.1 Introduction

The determinant maximization problem provides a powerful and general framework for modeling a wide variety of problems across several fields, including statistics [155], convex geometry [116], fair allocations [6], combinatorics [8], spectral graph theory [149], network design, and random processes [119].

In a typical instance of determinant maximization, we are given a collection of vectors $U = \{v_1, \ldots, v_n\} \subset \mathbb{R}^d$. The goal is to select a subset $S \subseteq [n]$ that maximizes the determinant of the matrix $\sum_{i \in S} v_i v_i^\top$. The selection of S may also be subject to combinatorial constraints, such as a cardinality constraint ($|S| \leq k$), or more generally, a matroid constraint (e.g., S must be a basis in a matroid over the vector indices.). Maximizing the determinant naturally promotes **diversity**: the selected vectors tend to be well spread out and linearly independent, leading to informative and representative summaries of the dataset.

Beyond its modeling flexibility, determinant maximization has revealed deep connections across fields such as combinatorial optimization, convex analysis, the geometry of polynomials, graph sparsification, and the complexity of the permanent and other counting problems [3, 5, 6, 116].

Applications

1. (Convex geometry) When exactly d vectors are selected, the objective becomes the square of the volume of the parallelepiped spanned by the selected vectors. The problem of finding the largest volume parallelepiped from a set of vectors has been studied extensively for over three decades [116, 147, 181].

2. (Random processes) Another notable application is in determinantal point processes (DPPs) [119]. In DPPs, a probability distribution is defined over subsets of vectors, where the probability of a subset is proportional to the squared volume of the parallelepiped they span. These distributions exhibit appealing properties, such as negative correlation, and

the task of finding the most probable subset is equivalent to solving a determinant maximization problem.

3. (Experimental design) A classical example arises in statistics through the experimental design problem [155]. The objective is to estimate an unknown parameter vector $\theta^\star \in \mathbb{R}^d$ using linear observations of the form $y_i = v_i^\top \theta^\star + \eta_i$, where $v_i \in \mathbb{R}^d$ and η_i is Gaussian noise.

   Given a candidate set of vectors $\{v_1, \ldots, v_n\}$, we aim to choose a small subset S of size $r \ll n$ to maximize the accuracy of estimating $\theta^\star$. Under Gaussian noise, the maximum likelihood estimate is given by the ordinary least squares solution:

$$\widehat{\theta} = \arg\min_{\theta} \sum_{i \in S} |v_i^\top \theta - y_i|^2.$$

   The estimation error $\widehat{\theta} - \theta^\star$ follows a d-dimensional Gaussian distribution with covariance matrix proportional to $\left(\sum_{i \in S} v_i v_i^\top\right)^{-1}$. Minimizing the determinant of this covariance matrix—equivalent to minimizing the volume of the confidence ellipsoid—leads directly to a determinant maximization problem under a uniform matroid constraint of rank r.

4. (Nash Social Welfare) Given a collection of m indivisible items and n players, along with a valuation function for each player that specifies how much the player values a particular bundle of items, the goal is to find an allocation of the items to the players that maximizes the geometric mean of the players' valuations [60].

   The geometric mean objective captures both fairness—ensuring that each player receives a bundle of significant value—and efficiency—ensuring that items are allocated to those who value them most. When the valuation functions of the players are additive over items, this problem can be modeled as a special case of determinant maximization under a partition matroid constraint [6].

Computational Aspects

The computational complexity of determinant maximization depends heavily on the structure of the combinatorial constraint that defines the feasible subsets of vectors. The most extensively studied case is when the constraint is simply cardinality-based—that is, when the number of selected vectors is fixed. This setting has been well-explored using a range of algorithmic techniques, including convex programming methods [3, 147, 174, 181], combinatorial methods such as local search and greedy selection [116, 124, 130], and connections to graph sparsification [3]. These approaches have led to efficient approximation algorithms with strong performance guarantees, offering a solid understanding of the problem's complexity in this regime.

The more general setting, where the feasible sets are defined by a matroid constraint, has received increasing attention in recent years [5, 6, 8, 131, 148]. This is especially interesting, as several applications are naturally modeled using matroid constraints — particularly partition matroids.

However, the matroid-constrained case presents a significant gap between what can be achieved through estimation and what can be computed algorithmically. In particular, while it is possible

to approximately *estimate* the optimal determinant value with good guarantees, actually *finding* a subset that achieves such value is much more challenging. This gap can be substantial: for example, even in the case of partition matroids, there exists an e^d-approximate estimation algorithm, but the best known approximation algorithms achieve only an $e^{O(d^2)}$ factor—an exponential blow-up.[1]

This gap largely stems from the fact that current estimation methods rely on non-constructive tools such as convex relaxations and the theory of stable polynomials and its generalization to strongly log-concave polynomials. Unfortunately, these methods are inherently non-algorithmic and do not give a simple way to obtain efficient algorithms with the same guarantees that match the estimation bounds.

2.1.1 Preliminaries and Notation

Let $U = \{v_1, \ldots, v_n\} \subseteq \mathbb{R}^d$ be a collection of vectors. For any subset of indices $S \subseteq [n]$, we define:

- $V_S := \{v_i : i \in S\}$ — the set of vectors indexed by S. By slight abuse of notation, we also use V_S to denote the $d \times |S|$ matrix whose columns are these vectors, i.e., $V_S = [v_i]_{i \in S}$.
- $\mathrm{vol}(S)$ — the volume of the parallelepiped spanned by the vectors in V_S, defined as

  $$\mathrm{vol}(S) := |\det(V_S)|,$$

  when $|S| = d$. This notion can be extended to $|S| \leq d$ as

  $$\mathrm{vol}(S) := \sqrt{\det(V_S^\top V_S)}.$$

  Although volume is not defined for $|S| > d$, the relevant algebraic surrogate we use is

  $$\mathrm{vol}(S) := \sqrt{\det(V_S V_S^\top)} = \sqrt{\det\left(\sum_{i \in S} v_i v_i^\top\right)}.$$

We use $\mathcal{M} = ([n], \mathcal{I})$ to denote a matroid over the ground set $[n]$ with independent sets $\mathcal{I}$. Whenever we refer to a set $S \in \mathcal{I}$ as a basis, we mean the index set, and use V_S to refer to the corresponding vectors or matrix.

We adopt the following index conventions throughout: the symbol j is used for vectors $u_j \notin V_S$ (i.e., outside the current solution), and i for vectors $v_i \in V_S$ (i.e., inside the solution). For a matrix $A \in \mathbb{R}^{S \times T}$, and subsets $Y \subseteq S$, $X \subseteq T$, we use $A_{Y,X}$ to denote the submatrix of A consisting of rows indexed by Y and columns indexed by X. We use both a_{ij} and a_{vu} to refer to matrix entries, depending on whether we are indexing by position or by the corresponding vectors.

[1] Since the determinant is taken over $d \times d$ matrices, approximation factors are often reported in terms of the d^{th} root of the determinant, making the exponential dependence on d appropriate.

For notational convenience, we may write expressions like $S \Delta C$, $S \cap C$, or $C \setminus S$ even when $C \subseteq U$ is a set of vectors, interpreting these as operations on the corresponding index sets. Similarly, we may use notation such as $\mathrm{vol}(S + u - v)$, where S is an index set and u, v are vectors, with the understanding that this refers to modifying the associated vector set. Let $\mathrm{span}(S) := \mathrm{span}(V_S)$. Finally, we denote by $\mathrm{sym}_r(M)$ the r^{th} elementary symmetric polynomial of the eigenvalues of a $d \times d$ positive semidefinite matrix M.

2.1.2 Our Results and Contributions

In this work, we introduce new combinatorial techniques for determinant maximization under matroid constraints. We give a deterministic approximation algorithm with approximation factor $O(d^{O(d)})$, improving significantly over the previously best-known bounds. Unlike earlier approaches that rely heavily on convex programming and the theory of stable polynomials, our algorithm is purely combinatorial and builds on the classical matroid intersection algorithm.

Our main results handle three different regimes, depending on the relationship between the matroid rank r and the dimension d.

Case 1 ($r = d$)

Our first result addresses the case where the rank of the matroid is exactly d, meaning the selected set must contain exactly d vectors. In this setting, the determinant of the matrix $\sum_{i \in S} v_i v_i^\top$ corresponds to the squared volume of the parallelepiped spanned by the selected vectors.

Theorem 1.2.1 (Rank equals dimension)**.** *There is a polynomial-time algorithm which, given a collection of vectors $v_1, \ldots, v_n \in \mathbb{R}^d$ and a matroid $\mathcal{M} = ([n], \mathcal{I})$ of rank d, returns a set $S \in \mathcal{I}$ such that*

$$\det\left(\sum_{i \in S} v_i v_i^\top\right) = \Omega\left(\frac{1}{d^{O(d)}}\right) \max_{S^* \in \mathcal{I}} \det\left(\sum_{i \in S^*} v_i v_i^\top\right).$$

Our results improve the $e^{O(d^2)}$-approximation algorithm which relies on the $e^{O(d)}$-estimation algorithm [5, 8, 131]. Our algorithm iteratively improves the objective while maintaining matroid feasibility. This is achieved by performing exchanges guided by alternating cycles in a carefully constructed *exchange graph*, building on ideas from the matroid intersection framework.

Case 2 ($r < d$)

We extend our techniques to the case where the matroid rank r is less than d. In this setting, the solution matrix $\sum_{i \in S} v_i v_i^\top$ is a $d \times d$ positive semidefinite matrix of rank at most r. Since its determinant is zero, a more meaningful objective is the product of its top r eigenvalues, which corresponds to the r^{th} elementary symmetric polynomial of its eigenvalues.

Let $\mathrm{sym}_r(M)$ denote the r^{th} elementary symmetric function of the eigenvalues of a $d \times d$ matrix M. Our objective, then, is to maximize $\mathrm{sym}_r\left(\sum_{i \in S} v_i v_i^\top\right)$. When $|S| \leq d$, this is equivalent to the squared volume $\mathrm{vol}(S)^2$ as defined in the Preliminaries § 2.1.1.

Theorem 1.2.2 (Rank less than dimension)**.** *There is a polynomial-time algorithm which, given a collection of vectors $v_1, \ldots, v_n \in \mathbb{R}^d$ and a matroid $\mathcal{M} = ([n], \mathcal{I})$ of rank $r \leq d$, returns a set $S \in \mathcal{I}$ such that*

$$\operatorname{sym}_r \left(\sum_{i \in S} v_i v_i^\top \right) = \Omega \left(\frac{1}{r^{O(r)}} \right) \max_{S^* \in \mathcal{I}} \operatorname{sym}_r \left(\sum_{i \in S^*} v_i v_i^\top \right).$$

This result improves over the best known $e^{O(r^2)}$-approximation algorithms derived from $e^{O(r)}$-approximate estimation methods.

Case 3 ($r > d$)

Finally, we consider the case where the number of selected vectors exceeds the dimension d. Our approach in this overdetermined setting uses sparsity properties of a convex programming relaxation along with the algorithmic ideas developed for the earlier cases.

Theorem 1.2.3 (Rank more than dimension)**.** *There is a polynomial-time algorithm which, given a collection of vectors $v_1, \ldots, v_n \in \mathbb{R}^d$ and a matroid $\mathcal{M} = ([n], \mathcal{I})$ of rank $r \geq d$, returns a set $S \in \mathcal{I}$ such that*

$$\det \left(\sum_{i \in S} v_i v_i^\top \right) = \Omega \left(\frac{1}{d^{O(d)}} \right) \max_{S^* \in \mathcal{I}} \det \left(\sum_{i \in S^*} v_i v_i^\top \right).$$

This matches the $d^{O(d)}$-approximate estimation algorithm [131], which only gives an estimate of the optimum value and results in $d^{O(d^2)}$-approximation algorithm.

Remark. Notice that the theorem statements for the $r < d$ and $r > d$ cases each subsume the result for $r = d$. We present the $r = d$ case separately for the sake of conceptual clarity, as it serves as a natural starting point for understanding the algorithm and analysis.

2.1.3 Technical Overview

To build intuition, recall that when $|S| = d$, the squared volume of the parallelepiped spanned by the vectors indexed by S is given by $\operatorname{vol}(S)^2 = \det(V_S V_S^\top)$. Therefore, maximizing volume is equivalent to maximizing the determinant of the sum of outer products.

We begin by considering the feasibility problem: is there a set $S \in \mathcal{I}$ such that $\operatorname{vol}(S) > 0$? This reduces to matroid intersection. Specifically, this question is equivalent to asking whether there exists a common basis between the given matroid $\mathcal{M}$ and the linear matroid defined by the vectors $\{v_1, \ldots, v_n\}$ (i.e., the column independence structure).

Since our goal is to maximize $\operatorname{vol}(S)$ over all $S \in \mathcal{I}$, a natural idea is to use the weighted matroid intersection algorithm. However, the key challenge is that the volume objective is not linear: we cannot write $\operatorname{vol}(S)$ as a sum $\sum_{i \in S} w_i$ or even as a product $\prod_{i \in S} w_i$ for some weights w. Nevertheless, our algorithm takes inspiration from the matroid intersection framework.

Overview of Matroid Intersection

Let us briefly recall the classical matroid intersection algorithm. Given two matroids $\mathcal{M}_1 = (U, \mathcal{I}_1)$ and $\mathcal{M}_2 = (U, \mathcal{I}_2)$ on the same ground set $U = \{1, \ldots, n\}$, and a weight function $w : U \to \mathbb{R}$, the goal is to find a common basis S maximizing $w(S) := \sum_{i \in S} w_i$, assuming one exists.

The algorithm starts from a common basis S and either certifies optimality or finds a strictly better common basis S'. It constructs a directed bipartite exchange graph $G(S)$ with bipartition $(U \setminus S, S)$. It adds:

- an arc from $j \in U \setminus S$ to $i \in S$ if replacing i with j in S yields a basis in $\mathcal{M}_2$;
- an arc from $i \in S$ to $j \in U \setminus S$ if the same operation yields a basis in $\mathcal{M}_1$.

Each $j \in U \setminus S$ gets a weight of $-w_j$, and each $i \in S$ gets weight w_i. A foundational result from matroid theory (Theorem 41.5 in [172]) shows that S is a maximum weight common basis if and only if $G(S)$ contains no negative weight cycle. Moreover, if C is a directed negative weight cycle with minimum hops[2], then $S \Delta C$ is a common basis of the two matroids whose weight is strictly larger than that of S.

Our Algorithm

Our algorithm adapts this framework to the determinant maximization setting. The two matroids involved are the constraint matroid $\mathcal{M}$ and the linear matroid defined by the columns $\{v_1, \ldots, v_n\}$. A key difficulty is that the objective function $\det(V_S V_S^\top) = \mathrm{vol}(S)^2$ is not linear, so vertex weights cannot be used as in the classical case.

Instead, we work with the function $\log \mathrm{vol}(S)$, which is known to be submodular. While we do not directly use submodularity, we linearize $\log \mathrm{vol}(S)$ locally and search for improvement directions using an exchange graph. In the case that $r \leq d$, we use the geometric relationship between vol and $\det$ closely, while we take a more algebraic approach when $r \geq d$.

The first new ingredient is the use of arc weights (rather than vertex weights) in the exchange graph. For each forward arc (j, i), where $j \notin S$ and $i \in S$, corresponding to the linear matroid, we assign a weight of $-\log\left(\frac{\mathrm{vol}(S-i+j)}{\mathrm{vol}(S)}\right)$. Backward arcs (corresponding to the constraint matroid $\mathcal{M}$) are given zero weight.

A key observation is that this volume ratio has a geometric interpretation. Let u_j be written in the basis V_S as $u_j = \sum_{i \in S} a_{ij} v_i$. Then:

$$\frac{\mathrm{vol}(S - v_i + u_j)}{\mathrm{vol}(S)} = |a_{ij}|.$$

This relationship (formalized in Lemma 2.2.4) plays a central role in our analysis.

[2]Hops here refers to the number of arcs in the cycle.

From Determinant to Cycle

We show that if the current solution is far from optimal in terms of volume, then the exchange graph must contain a cycle that violates an appropriate inequality:

Lemma 2.1.1 (Determinant to Cycle)**.** *Let S be a basis in $\mathcal{M}$, and let* OPT *be an optimal basis maximizing* $\mathrm{vol}(\mathrm{OPT})$*. If* $\mathrm{vol}(\mathrm{OPT}) \geq e^{12d \log d} \cdot \mathrm{vol}(S)$*, then there exists a directed cycle C of 2ℓ hops in $G(S)$ such that*

$$\prod_{(j,i) \in C,\ j \notin S,\ i \in S} |a_{ij}| \geq \max\{2, (\ell!)^{11}\} =: f(\ell).$$

Such a cycle is called an *f-violating cycle*. To detect it, we define new arc weights for forward arcs as

$$w_\ell(j,i) = \frac{1}{\ell} \log f(\ell) - \log |a_{ij}|,$$

and search for negative weight cycles of length 2ℓ.

Let V_S and V_{OPT} be the matrices with columns indexed by S and OPT, respectively. Writing each vector in OPT in the basis V_S, we get $V_{\mathrm{OPT}} = V_S A$ for some matrix A. The condition of the lemma implies that $|\det(A)| \geq e^{12d \log d}$. Furthermore, the arc weights from $u_j \in \mathrm{OPT}$ to $v_i \in S$ are $-\log |a_{ij}|$, where a_{ij} is the corresponding entry in A. These facts together guarantee the existence of an f-violating cycle.

From Cycle to Determinant

After finding an f-violating cycle C, the next step is to update the solution to $T := S \Delta C$. The key is to relate the volume of the new solution $\mathrm{vol}(T)$ to the coefficients a_{ij}. While the $|a_{ij}|$ values along the cycle are large, the determinant $\det(V_T V_T^\top)$ depends on more than just those values—it depends on all coefficients between vectors in $C \setminus S$ and $C \cap S$.

Let B be the matrix with rows indexed by $C \cap S$ and columns by $C \setminus S$, where $B_{ij} = a_{ij}$ is the coefficient of v_i in the expansion of u_j with respect to the basis V_S. Then:

$$\mathrm{vol}(T) = |\det(B)| \cdot \mathrm{vol}(S) \quad \text{(Lemma 2.2.11)}.$$

The diagonal entries of B correspond to the weights of the forward arcs on C, which are guaranteed to be large by Lemma 2.1.1. We show that if C is a *minimum hop f*-violating cycle, then the off-diagonal entries of B (i.e., chords of C) are small enough to ensure a strong lower bound on $\det(B)$.

Lemma 2.1.2 (Cycle to Determinant)**.** *If C is a minimum hop f-violating cycle in $G(S)$, then* $\mathrm{vol}(S \Delta C) \geq 2 \cdot \mathrm{vol}(S)$*. Moreover, $S \Delta C$ is also a basis of $\mathcal{M}$.*

This lemma is crucial: minimality of C controls the size of off-diagonal entries in B, allowing us to lower bound its determinant. A careful calculation then shows that each update at least doubles the volume, ensuring fast convergence.

2.1.4 Related Work

Determinant Maximization under Cardinality Constraints. Determinant maximization problems under a cardinality constraint have been studied widely [3, 116, 130, 147, 174, 181]. Currently, the best approximation algorithm for the case $r \leq d$ is an e^r-approximation due to Nikolov [147] and for $r \geq d$, there is an e^d-approximation [174]. It turns out that the problem gets significantly easier when $r >> d$, and there is a $(1+\varepsilon)^d$-approximation when $r \geq d + \frac{d}{\varepsilon}$ [3, 124, 130]. These results use local search methods and are closely related to the algorithm discussed in this chapter, as the cycle improving algorithm will always find a 2-cycle when the matroid is defined by the cardinality constraint.

Determinant Maximization under Matroid Constraints. As mentioned earlier, determinant maximization under a matroid constraint is considerably challenging and the bounds also depend on the rank r of the constraint matroid. There are $e^{O(r)}$-estimation algorithms when $r \leq d$ [5, 10, 148] and a $\min\{e^{O(r)}, O\left(d^{O(d)}\right)\}$-estimation algorithm when $r \geq d$ [131]. The output of these algorithms is a random feasible set whose objective is at least $\min\{e^{O(r)}, O\left(d^{O(d)}\right)\}$ of the objective of a convex programming relaxation, in expectation. Since the approximation guarantees are exponential, it can happen that the output set has objective zero almost always. To convert them into deterministic algorithms (or randomized algorithms that work with high probability), additional loss in approximation factor is incurred. These results imply an $e^{O(d^2)}$-approximation algorithm when $r \leq d$, and a $O\left(d^{O(d^3)}\right)$-approximation algorithm [131] for $r \geq d$. Approximation algorithms are also known where the approximation factor is exponential in the size of the ground set for special classes of matroids [71].

Nash Social Welfare and its generalizations. A special case of the determinant maximization problem is the Nash Social Welfare problem [41]. In the Nash Social Welfare problem, we are given m items and d players and there is a valuation function $v_i : 2^{[m]} \to \mathbb{R}_+$ for each player $i \in [d]$ that specifies value obtained by a player when given a bundle of items. The goal is to find an assignment of items to players to maximize the *geometric mean* of the valuations of each of the players. When the valuation functions are additive, the problem becomes a special case of the determinant maximization and this connection can be utilized to give an e-approximation algorithm [6]. Other methods including rounding algorithms [60, 61] as well as primal-dual methods [20] have been utilized to obtain improved bounds. The problem has been studied when the valuation function is more general [9, 21, 84, 86] and a constant-factor approximation is known when the valuation function is submodular [127].

Other Spectral Objectives. While we focus on the determinant objective, the problem is also interesting when considering other spectral objectives including minimizing the trace or the maximum eigenvalue of the $\left(\sum_{i \in S} \left(v_i v_i^\top\right)\right)^{-1}$. These problems have been studied for the cardinality constraint [3, 149]. For the case of partition matroid, the problem of maximizing the minimum eigenvalue is closely related to the Kadison-Singer problem [136].

2.1.5 Organization

In the next section we will provide a description of the algorithm in the special case when $r = d$, and the constraint matroid is a partition matroid. This case already contains the key technical ideas and will allow for a full description of the algorithm. The analysis is divided into two lemmas: Lemma 2.1.1, which shows that if there is an optimality gap we can find a very negative cycle, and Lemma 2.1.2 which shows that when we update along a minimal such very negative cycle, it leads to an improvement in the objective value. At the end of that section we briefly explain how the result extends to general matroid constraints.

In § 2.3 we address what modifications are needed when we are selecting fewer vectors than the dimension. This requires new versions of the determinant-to-cycle and cycle-to-determinant lemmas, and a slight modification to the algorithm.

In § 2.4 we complete the final case where we select more vectors than the dimension. In this case we use additional sparsity properties of the convex programming relaxation, but the general plan remains the same.

2.2 The case that rank equals dimension

In this section, we present the algorithm that proves the following theorem:

Theorem 1.2.1 (Rank equals dimension)**.** *There is a polynomial-time algorithm which, given a collection of vectors $v_1, \ldots, v_n \in \mathbb{R}^d$ and a matroid $\mathcal{M} = ([n], \mathcal{I})$ of rank d, returns a set $S \in \mathcal{I}$ such that*

$$\det\left(\sum_{i\in S} v_i v_i^\top\right) = \Omega\left(\frac{1}{d^{O(d)}}\right) \max_{S^*\in\mathcal{I}} \det\left(\sum_{i\in S^*} v_i v_i^\top\right).$$

We begin by analyzing the case of a partition matroid with rank d. This allows us to introduce the main ideas without invoking the full generality of matroid theory. The extension to general matroids is standard and is deferred to § 2.2.3.

Let $\mathcal{M}$ be a partition matroid with d parts, where each part $\mathcal{P}_i$ has capacity 1. Our goal is to find an index set S that selects one vector from each part and maximizes the determinant objective:

$$\max\left\{\det\left(\sum_{i\in S} v_i v_i^\top\right) : |S| = d,\ |S \cap \mathcal{P}_i| = 1 \text{ for all } i \in [d]\right\}.$$

Let OPT denote the optimal solution. The following theorem is a specialization of Theorem 1.2.1 to the partition matroid setting.

Theorem 2.2.1. *Given a partition matroid $\mathcal{M}$ with d parts, let OPT be the optimal solution to the determinant maximization problem under $\mathcal{M}$. Then, there is a polynomial-time deterministic algorithm that returns a feasible set $S \in \mathcal{M}$ such that*

$$\det\left(\sum_{i\in S} v_i v_i^\top\right) \geq e^{-24d\log(d)} \cdot \det\left(\sum_{i\in \mathrm{OPT}} v_i v_i^\top\right).$$

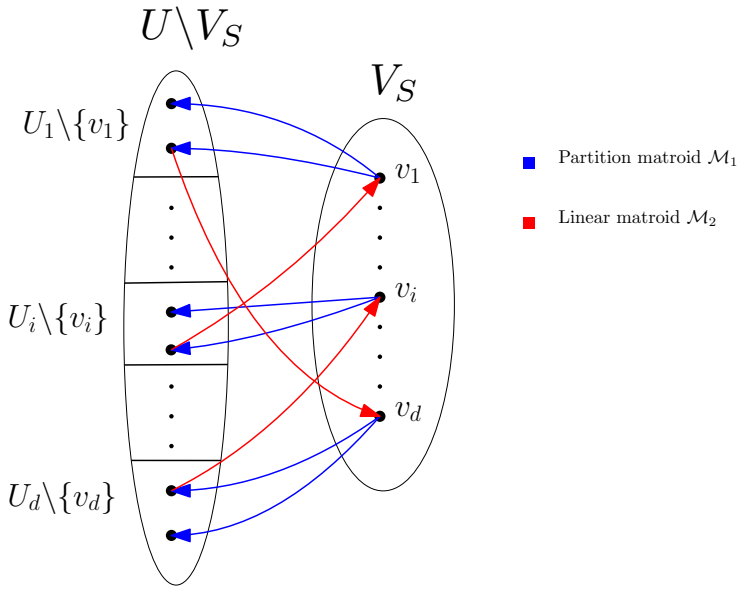


Figure 2.1: The exchange graph $G(S)$.

2.2.1 Algorithm

We now formally define the exchange graph, the relevant weight functions, and present the algorithm used to prove Theorem 2.2.1.

Let $U_i := \{v_j : j \in \mathcal{P}_i\}$ be the set of vectors in the i-th part of the partition matroid.

Definition 2.2.2 (Exchange Graph)**.** Let $S \subseteq [n]$ be an index set such that $|S \cap \mathcal{P}_i| = 1$ for all $i \in [d]$. For each i, let s_i be the unique element in $S \cap \mathcal{P}_i$, and define (or re-index such that) $v_i := v_{s_i}$. The exchange graph $G(S)$(see Figure 2.1) is a bipartite directed graph with:

- Right side: the selected vectors $R := V_S$,
- Left side: the remaining vectors $L := \bigcup_{i=1}^{d} (U_i \setminus \{v_i\})$.

The arcs are defined as follows:

- For each $v_i \in R$, there is a backward arc to every $u \in U_i \setminus \{v_i\}$,
- For each $u \in L$, there is a forward arc to every $v_i \in V_S$ such that $V_S - v_i + u$ is linearly independent.

More generally, the arcs from R to L correspond to swaps that preserve feasibility in the constraint matroid $\mathcal{M}$, while arcs from L to R correspond to arcs that preserve linear independence.

We define a family of weight functions on the arcs of $G(S)$. The base weight function is denoted by w_0, and for each $1 \leq \ell \leq d$, we define weight functions w_ℓ. We use the function $f : [d] \to \mathbb{Z}_+$ defined as $f(1) = 2$ and $f(i) = (i!)^{11}$ for $i \geq 2$.

Definition 2.2.3 (Weight Functions on the Exchange Graph)**.** For $u_j \in L$, let $v_i = V_S[i]$ be the representative from partition i. Suppose u_j can be written in the basis V_S as $u_j = \sum_{i=1}^{d} a_{ij} v_i$.

Then:

- The base weight function w_0 assigns weight $w_0(u_j, v_i) = -\log |a_{ij}|$ to each forward arc (u_j, v_i), and weight 0 to each backward arc (v_i, u_j).
- The modified weight function w_ℓ for $\ell \in [d]$ assigns:

$$w_\ell(u_j, v_i) := \frac{\log(f(\ell))}{\ell} + w_0(u_j, v_i),$$

  and still gives weight 0 to all backward arcs.

The weight $w_0(u_j, v_i)$ reflects the relative change in volume when replacing v_i with u_j. The following lemma, proved in the appendix, formalizes this:

Lemma 2.2.4. *Let $S \subseteq [n]$ with $\mathrm{vol}(S) > 0$, and let $u_j \notin S$, $v_i \in S$. Then*

$$w_0(u_j, v_i) = -\log\left(\frac{\mathrm{vol}(S - v_i + u_j)}{\mathrm{vol}(S)}\right).$$

From here on, when not specified, we assume the default weight function is w_0.

Definition 2.2.5 (Cycle Weight)**.** The weight of a cycle C in $G(S)$ is defined as $w_0(C) = \sum_{e \in C} w_0(e)$. Note that the weight only depends on the forward arcs, as all backward arcs have weight 0.

We now describe how to iteratively improve the current solution set S using updates guided by cycles in the exchange graph $G(S)$. However, not all cycles are useful for making progress. We define a special class of cycles called f-violating cycles and Minimal f-violating cycles. The algorithm always exchanges on a Minimal f-violating cycle.

Definition 2.2.6 (f-Violating Cycle)**.** A cycle C in $G(S)$ is called an f-violating cycle if

$$w_0(C) < -\log f(|C|/2),$$

where $|C|$ denotes the number of arcs in the cycle.

The next observation relates the weight of a cycle to the product of coefficients appearing in the vector representation and follows directly from the definition of arc weights.

Observation 2.2.7. If C is an f-violating cycle, then

$$\prod_{(u,v) \in C : u \in L,\, v \in R} |a_{vu}| > f(|C|/2).$$

The algorithm will always update S using minimal f-violating cycles, defined below.

Definition 2.2.8 (Minimal f-Violating Cycle)**.** A cycle C in $G(S)$ is a minimal f-violating cycle if:

- C is an f-violating cycle, and

- for every cycle C' such that $V(C') \subset V(C)$, C' is not an f-violating cycle.

To find such a cycle, we iterate over increasing values of ℓ and look for a cycle with 2ℓ arcs and negative weight under w_ℓ:

Algorithm 1 Finding a minimal f-violating cycle

```
for ℓ = 1 to d do
    if there is a negative-weight cycle C with exactly 2ℓ arcs in G(S) under w_ℓ then
        return C
    end if
end for
```

The correctness of the procedure is immediate from the definitions.

Lemma 2.2.9. *Algorithm 1 returns a minimal f-violating cycle in $G(S)$, if one exists.*

After finding a minimal f-violating cycle C, we update the current solution set via $S \leftarrow S\Delta C$, and repeat. By construction, $S\Delta C$ is a feasible set, as it continues to pick exactly one element from each part.

The core idea is as follows: if the volume of the current solution is small relative to the optimum—specifically, if $\mathrm{vol}(S) < \mathrm{vol}(\mathrm{OPT}) \cdot e^{-\Omega(d \log d)}$—then the exchange graph $G(S)$ must contain an f-violating cycle (see Lemma 2.2.10). Moreover, exchanging along a minimal f-violating cycle leads to a significant improvement: it at least doubles the volume, i.e., $\mathrm{vol}(S\Delta C) \geq 2 \cdot \mathrm{vol}(S)$ (see Lemma 2.2.12).

To initialize the process, we start with any feasible set S satisfying $\mathrm{vol}(S) > 0$. Such a set can be computed using the matroid intersection algorithm for the partition matroid and the linear matroid induced by the vectors. Since the ratio $\mathrm{vol}(\mathrm{OPT})/\mathrm{vol}(S)$ is at most $2^{4\sigma}$, where σ denotes the encoding length of the input (Chapter 3, Theorem 3.2 [170]), only polynomially many such exchanges are needed before reaching a solution that satisfies the bound in Theorem 2.2.1.

Algorithm 2 Exchange Algorithm for Partition Matroids

```
Initialize S such that |S| = d, |S ∩ P_i| = 1 for all i ∈ [d], and vol(S) > 0
while there exists an f-violating cycle in G(S) do
    C ← minimal f-violating cycle in G(S)
    S ← SΔC
end while
return S
```

Lemma 2.2.10. *For any set S with $|S| = d$ and $\mathrm{vol}(S) > 0$, if $\mathrm{vol}(S) < e^{-12d \log d} \cdot \mathrm{vol}(\mathrm{OPT})$, then $G(S)$ contains an f-violating cycle.*

Proof. Let $\mathrm{OPT} = \{u_1, u_2, \ldots, u_d\}$ and $S = \{v_1, v_2, \ldots, v_d\}$, where $u_i, v_i \in \mathcal{P}_i$ for all $i \in [d]$. Observe that (v_i, u_i) is a backward arc in $G(S)$ for every i where $u_i \neq v_i$.

Let V_T and V_S be the matrices whose columns are the vectors in OPT and S, respectively. Let A be the coefficient matrix such that $V_T = V_S A$. Then,

$$\mathrm{vol}(\mathrm{OPT})^2 = \det(V_T V_T^\top) = \det(V_S A A^\top V_S^\top) = \mathrm{vol}(S)^2 \cdot |\det(A)|^2.$$

Define $X = \mathrm{OPT} \setminus S$ and $Y = S \setminus \mathrm{OPT}$, and suppose $|X| = |Y| = k$. Without loss of generality, write $Y = \{v_1, \ldots, v_k\}$ and $X = \{u_1, \ldots, u_k\}$. Then A takes the block form:

$$A = \begin{bmatrix} A_k & 0 \\ A'_k & I_{d-k} \end{bmatrix},$$

where A_k is a $k \times k$ matrix. Hence $\det(A) = \det(A_k)$.

Using the hypothesis $\mathrm{vol}(S)^2 < \mathrm{vol}(\mathrm{OPT})^2 \cdot e^{-24d \log d}$, we get

$$|\det(A_k)| > e^{12d \log d} \geq e^{12k \log k}.$$

By the Leibniz formula and bounding the number of permutations,

$$|\det(A_k)| \leq \sum_{\sigma \in \mathcal{S}_k} \prod_{i=1}^{k} |a_{i\sigma(i)}| \leq k! \cdot \max_{\sigma \in \mathcal{S}_k} \prod_{i=1}^{k} |a_{i\sigma(i)}|.$$

Hence, for some $\sigma \in \mathcal{S}_k$, we must have:

$$\prod_{i=1}^{k} |a_{i\sigma(i)}| \geq \frac{|\det(A_k)|}{k!} > e^{11k \log k}.$$

Let $\sigma = \{C_1, \ldots, C_\ell\}$ be the decomposition of σ into disjoint cycles. Each C_j corresponds to a cycle in $G(S)$ with $2|C_j|$ arcs (alternating forward and backward). If all of these cycles were non-violating, then

$$\prod_{i=1}^{k} |a_{i\sigma(i)}| = \prod_{j=1}^{\ell} \prod_{i \in C_j} |a_{i\sigma(i)}| \leq \prod_{j=1}^{\ell} f(|C_j|) \leq e^{11k \log k},$$

contradicting the bound above. Therefore, at least one f-violating cycle exists. □

Tightness of the bound The volume threshold in Lemma 2.2.10 is essentially tight. For example, let $S = \{e_1, \ldots, e_d\}$ be the standard basis of $\mathbb{R}^d$, and let $\mathrm{OPT} = \{h_1, \ldots, h_d\}$ be the columns of a $d \times d$ Hadamard matrix. Then $\mathrm{vol}(S) = 1$ and $\mathrm{vol}(\mathrm{OPT}) = \prod_{i=1}^{d} \|h_i\| = d^{d/2} = e^{\frac{d}{2} \log d} \cdot \mathrm{vol}(S)$. However, since every entry of the exchange matrix $A = H$ is ± 1, the product of coefficients along any cycle has absolute value 1. Thus, no f-violating cycle exists, despite the large gap in volume.

2.2.2 Cycle Exchange and Determinant

We now show that exchanging along a minimal f-violating cycle C increases the objective value—specifically, the squared volume—by at least a factor of two. The proof relies on two technical lemmas.

First, observe that the arc weights $w_0(j, i)$ quantify the change in the objective when we switch from the current solution S to $S + j - i$. However, exchanging along a cycle involves replacing multiple elements simultaneously. Since our function $\mathrm{vol}(\cdot)$ (or more precisely, $\log \mathrm{vol}(\cdot)$) is not additive, it is not immediately clear how the objective changes. The following lemma characterizes the exact change when we replace a large subset.

Let S be the current solution. Let C be a minimal f-violating cycle, and define $\ell = |C|/2$. Let $X = C \setminus S$ and $Y = C \cap S$. Then the updated set is $T = (S \cup X) \setminus Y$. We denote by V_X, V_Y, and V_S the matrices whose columns correspond to the elements of X, Y, and S, respectively. Note that V_S is a $d \times d$ matrix, while both V_X and V_Y are $d \times \ell$. Observe:

$$\mathrm{vol}(S)^2 = \det(V_S V_S^\top), \quad \mathrm{vol}(T)^2 = \det(V_T V_T^\top) = \det(V_S V_S^\top + V_X V_X^\top - V_Y V_Y^\top).$$

The matrix of coefficients a_{ij}, which defines the arc weights for $j \in X$ and $i \in Y$ in the exchange graph, also governs this change in volume.

Lemma 2.2.11. *Let S be a basis, and let X, Y be sets with $|X| = |Y| = \ell$ and $Y \subseteq S, X \subseteq U \setminus S$. Let A be the $d \times \ell$ matrix such that $V_X = V_S A$, and let A_C be the $\ell \times \ell$ submatrix of A restricted to the rows indexed by Y. Then, for $T = (S \cup X) \setminus Y$, we have:*

$$\mathrm{vol}(T)^2 = \mathrm{vol}(S)^2 \cdot \det(A_C A_C^\top).$$

Without loss of generality, write the cycle as

$$C = (v_0 \to u_1 \to v_1 \to u_2 \to v_2 \to \cdots \to u_\ell \to v_0),$$

with $X = \{u_1, \ldots, u_\ell\}$ and $Y = \{v_1, \ldots, v_{\ell-1}, v_0\}$. Order the rows of A_C so that the last row corresponds to v_0. The diagonal entries a_{ii} of A_C represent the coefficients of v_i when expressing u_i in the basis S. Since C is f-violating, we have:

$$\prod_{i=1}^{\ell} |a_{ii}| > f(\ell).$$

To show that $\mathrm{vol}(T)$ is large, it suffices to lower-bound $|\det(A_C)|$. Crucially, since C is a minimal f-violating cycle, any chord creates a strictly smaller cycle that must not be f-violating. This allows us to upper-bound the off-diagonal entries a_{ij}.

Lemma 2.2.12. *If C is a minimal f-violating cycle in $G(S)$, then* $\mathrm{vol}(S \Delta C) \geq 2 \cdot \mathrm{vol}(S)$.

Proof. Let $C = (v_0 \to u_1 \to v_1 \to \cdots \to u_\ell \to v_0)$ where v_i, u_{i+1} belong to the same part and $v_i \in S$ (see Figure 2.2).

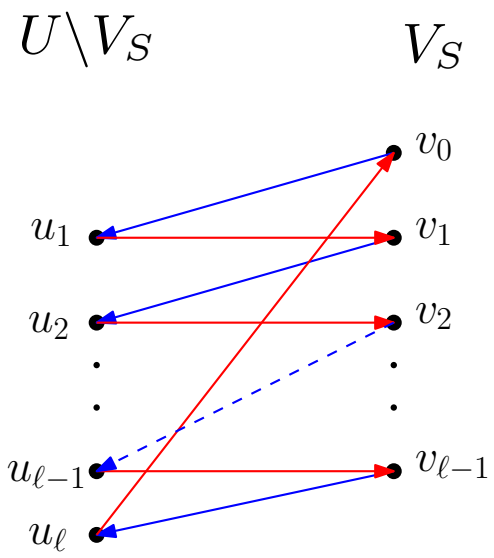


Figure 2.2: The cycle C

From Lemma 2.2.11, we know:

$$\mathrm{vol}(S \Delta C) = |\det(A_C)| \cdot \mathrm{vol}(S).$$

Using notation from Lemma 2.2.11 with $X = C \backslash S$ and $Y = C \cap S$. Index the entries of A_C using $\{v_i\}_{0 \le i \le \ell-1}$ and $\{u_j\}_{1 \le j \le \ell}$, with the last row corresponding to v_0 (we use v_ℓ and v_0 interchangeably). Since C has 2ℓ edges, A_C is an $\ell \times \ell$ matrix.

We now upper-bound the off-diagonal entries of A_C in terms of its diagonal entries. For $i < j$, define the cycle:

$$C_{i,j} := (u_j \to v_i \to u_{i+1} \to \cdots \to v_{j-1} \to u_j).$$

This cycle has $2(j-i)$ edges and is a proper subgraph of C. Since C is minimal, $C_{i,j}$ is not f-violating. Thus:

$$|a_{i,j}| \cdot \prod_{s=i+1}^{j-1} |a_{s,s}| < f(j-i) \quad \Rightarrow \quad |a_{i,j}| < \frac{f(j-i)}{\prod_{s=i+1}^{j-1} |a_{s,s}|}.$$

Note that $i = 0$ is part of the $i < j$ case. For $j < i < \ell$, define:

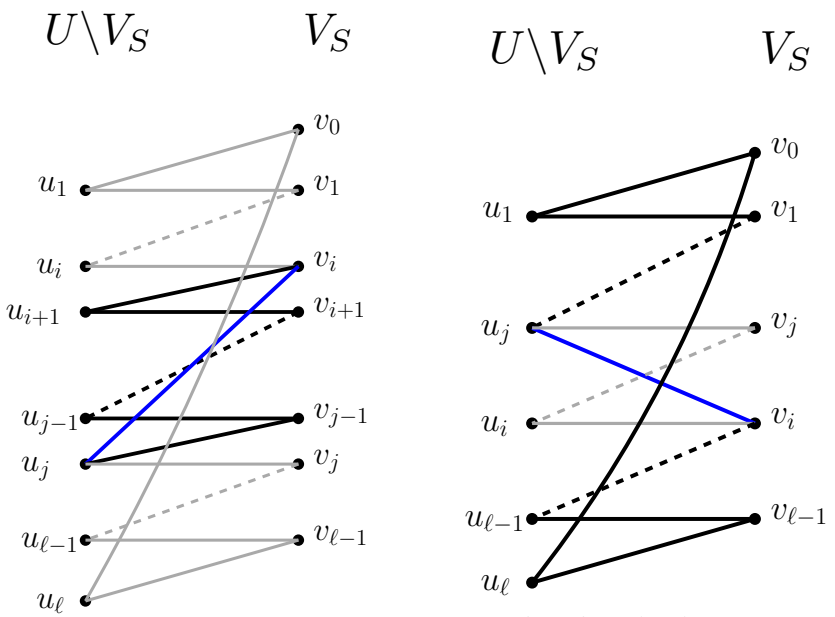
$$C'_{i,j} := (v_0 \to u_1 \to \cdots \to u_j \to v_i \to \cdots \to u_\ell \to v_0).$$

This cycle has $2(\ell - i + j)$ edges and is not f-violating. Hence:

$$|a_{i,j}| \cdot \prod_{s=1}^{j-1} |a_{s,s}| \cdot \prod_{s=i+1}^{\ell} |a_{s,s}| < f(\ell - i + j).$$

Since C is f-violating, we also have:

$$\prod_{s=1}^{\ell} |a_{s,s}| > f(\ell),$$



(a) Sub-cycle when $i < j$

(b) Sub-cycle when $i > j$

Figure 2.3: Structure when edge $u_j \to v_i$ (in blue) is added

which leads to:

$$|a_{i,j}| < \frac{f(\ell - i + j)}{f(\ell)} \cdot \prod_{s=j}^{i} |a_{s,s}|.$$

Now define matrix B_ℓ by applying the following transformations to A_C:

- Multiply the columns $1 < j \leq \ell$ with $\prod_{s=1}^{j-1} a_{s,s}$.
- Divide the rows $1 \leq i \leq \ell - 1$ by $\prod_{s=1}^{i} a_{s,s}$.
- Divide last row by $f(\ell)$.

Then $|\det(A_C)| = f(\ell) \cdot |\det(B_\ell)|$, and B_ℓ satisfies:

- $b_{i,i} = 1$ for all $i < \ell$ and $b_{\ell,\ell} \geq 1$,
- $|b_{i,j}| \leq f(j - i)$ for $i < j$,
- $|b_{i,j}| \leq f(\ell - i + j)/f(\ell)$ for $j < i$.

By Corollary 2.5.4, we have:

$$\det(B_\ell) \geq 1 - \frac{0.1}{\ell},$$

and therefore:

$$|\det(A_C)| \geq f(\ell) \cdot \left(1 - \frac{0.1}{\ell}\right) \geq 2 \quad \text{for } \ell \geq 2.$$

□

2.2.3 Update Step for General Matroids

Consider the case where $\mathcal{M} = ([n], \mathcal{I})$ is a general matroid of rank d. When we exchange along a cycle C and update $S \leftarrow S\Delta C$, the resulting set is guaranteed to remain independent in the linear matroid, due to the determinant bound in Lemma 2.2.12. However, it is not immediately clear that $S\Delta C$ will remain independent in a general matroid $\mathcal{M}$, especially when $\mathcal{M}$ is not a partition matroid.

In this section, we show that by exchanging along a minimal f-violating cycle, we can preserve independence in $\mathcal{M}$. Specifically, we prove the existence of an f-violating cycle whenever the current basis S has volume significantly smaller than that of the optimal solution OPT. We also show that exchanging along a minimal f-violating cycle preserves matroid independence.

Theorem 2.2.13. *Let S be a basis with $|S| = d$ and $\mathrm{vol}(S) > 0$. If $\mathrm{vol}(S) < \mathrm{vol}(\mathrm{OPT}) \cdot e^{-12d\log(d)}$, then there exists an f-violating cycle in $G(S)$.*

Proof. Since S and OPT are both independent sets of size d, there exists a perfect matching between $\mathrm{OPT} \setminus S$ and $S \setminus \mathrm{OPT}$ using the backward arcs in $G(S)$ (Chapter 39, Corollary 39.12a, [172]). Let $X = \mathrm{OPT} \setminus S$ and $Y = S \setminus \mathrm{OPT}$, with $|X| = |Y| = k$. Without loss of generality, let $Y = \{v_1, \ldots, v_k\}$ and $X = \{u_1, \ldots, u_k\}$ such that each $(v_i \to u_i)$ is an arc in $G(S)$ for $i \in [k]$.

Let V_T and V_S be the matrices whose columns are the vectors in OPT and S, respectively. Define A as the matrix of coefficients such that $V_T = V_S A$. Then A can be expressed in block form as

$$A = \begin{bmatrix} A_k & 0 \\ A' & I_{d-k} \end{bmatrix},$$

where A_k is the submatrix corresponding to rows indexed by X and columns indexed by Y.

Applying the same argument as in Lemma 2.2.10, there exists a permutation $\sigma \in \mathcal{S}_k$ such that

$$\prod_{i=1}^{k} |a_{i\sigma(i)}| > |\det(A)| \cdot e^{-k\log(k)} \geq e^{11k\log(k)}. \tag{2.1}$$

Let $\sigma = \{C_1, \ldots, C_\ell\}$ be the decomposition of σ into disjoint cycles. Each C_j corresponds to a cycle in $G(S)$ with $2|C_j|$ arcs (alternating forward and backward). If all of these cycles were non-violating, then

$$\prod_{i=1}^{k} |a_{i\sigma(i)}| = \prod_{j=1}^{\ell} \prod_{i \in C_j} |a_{i\sigma(i)}| \leq \prod_{j=1}^{\ell} f(|C_j|) \leq e^{11k\log k},$$

which contradicts (2.1). Thus, $G(S)$ must contain an f-violating cycle. □

Lemma 2.2.14. *If C is a minimal f-violating cycle in $G(S)$, then $S\Delta C$ is independent in $\mathcal{M}$.*

Proof. Let $V(C)$ denote the vertex set of the cycle C, and define $T := S \Delta C$. Suppose $|C| = 2\ell$. Consider the graph $G(S)$ with arc weights w_ℓ, and define the total weight of a cycle D as $w_\ell(D) := \sum_{e \in D} w_\ell(e)$. Since C is f-violating, we have

$$w_\ell(C) = w_0(C) + \log(f(\ell)) < 0.$$

Let N_1 be the set of backward arcs and N_2 the set of forward arcs in C. Suppose, for contradiction, that $T \notin \mathcal{I}$. Then, by Theorem 39.13 of [172], there exists a distinct matching N_1' on $V(C)$ consisting solely of backward arcs, with $N_1' \neq N_1$.

Construct a multiset A of arcs as follows: include each arc in N_2 twice, and include all arcs in $N_1 \cup N_1'$, with arcs in $N_1 \cap N_1'$ counted twice. Let $D = (V(C), A)$ be the resulting directed multigraph. Each vertex in $V(C)$ has in-degree and out-degree two, so D is Eulerian and decomposes into directed circuits $C_1, \ldots, C_k$.

Since only arcs in N_2 have nonzero weights, we have:

$$\sum_{i=1}^{k} w_\ell(C_i) = 2w_\ell(C).$$

Now, either:

(i) there exists a C_j with $V(C_j) = V(C)$, or

(ii) all C_j satisfy $V(C_j) \subsetneq V(C)$.

In case (i), $w_\ell(C_j) = w_\ell(C)$, and so the remaining circuits must sum to $w_\ell(C) < 0$, implying the existence of some $C_i \subsetneq C$ with $w_\ell(C_i) < 0$.

In case (ii), $\sum_i w_\ell(C_i) = 2w_\ell(C) < 0$, so again some C_i has $w_\ell(C_i) < 0$ and $V(C_i) \subsetneq V(C)$.

Thus, in either case, we find a subcycle $C' \subsetneq C$ with $w_\ell(C') < 0$. Let $|C'| = 2y$. Then:

$$w_\ell(C') = \frac{y}{\ell} \log(f(\ell)) + w_0(C') < 0.$$

Since $y < \ell$ and $\log(f(y))/y \leq \log(f(\ell))/\ell$, we have:

$$w_0(C') \leq -\frac{y}{\ell} \log(f(\ell)) \leq -\log(f(y)).$$

Thus C' is an f-violating cycle with $V(C') \subsetneq V(C)$, contradicting the minimality of C. □

2.3 Rank less than dimension

In this section, we extend Algorithm 2 to the setting where the number of vectors selected is less than the dimension d. Specifically, we address the case when the matroid rank $r \leq d$ and prove Theorem 1.2.2.

Let $\mathcal{M} = ([n], \mathcal{I})$ be a matroid of rank $r \leq d$. Starting with a basis S with non-zero volume, we apply a slight modification of Algorithm 2 to iteratively find a basis with strictly larger volume. However, since S is not full-dimensional in $\mathbb{R}^d$, the edge weight functions in our exchange graph must be adapted accordingly.

Let $S = \{v_1, v_2, \ldots, v_r\}$ be a basis of $\mathcal{M}$ such that $\mathrm{vol}(S) > 0$. Any vector $u_j \in U$ can be written as

$$u_j = \sum_{i=1}^{r} a_{ij} v_i + u_j^{\perp},$$

where $u_j^{\perp}$ is the component of u_j orthogonal to $\mathrm{span}(V_S)$.

The change in volume incurred by replacing some $v_i \in S$ with $u_j \notin S$ is given by

$$\frac{\mathrm{vol}(S - v_i + u_j)}{\mathrm{vol}(S)} = \sqrt{a_{ij}^2 + \frac{\|u_j^{\perp}\|^2}{\|v_i^{\perp}\|^2}}, \tag{2.2}$$

where $v_i^{\perp}$ is the component of v_i orthogonal to $\mathrm{span}(V_S \setminus \{v_i\})$.

Each of the terms in (2.2) has a geometric interpretation. Let us decompose u_j as $u_j = u_j^{\parallel} + u_j^{\perp}$, where $u_j^{\parallel}$ lies in $\mathrm{span}(S)$. Then:

- $|a_{ij}| = \frac{\mathrm{vol}(S - v_i + u_j^{\parallel})}{\mathrm{vol}(S)}$ captures the change in volume if we replace v_i with the component of u_j parallel to the span of S.
- $\frac{\|u_j^{\perp}\|}{\|v_i^{\perp}\|} = \frac{\mathrm{vol}(S - v_i + u_j^{\perp})}{\mathrm{vol}(S)}$ captures the change in volume if we replace v_i with the component of u_j perpendicular to the span of S.

This motivates an augmentation of the exchange graph to account for both components.

As in Lemma 2.2.10, we show that if $\mathrm{sym}_r(S) < \mathrm{sym}_r(\mathrm{OPT}) \cdot r^{-\Omega(r)}$, then there exists an f-violating cycle in the augmented exchange graph.

However, unlike Lemma 2.2.12, the change in the objective induced by a cycle C in the augmented graph is not a simple function of the arc and chord weights in C. To overcome this, we use geometric properties of the volume function—specifically, its subadditivity—to relate sym_r to the total weight along arcs and chords in the cycle.

Finally, each vector $v_i \in S$ can be decomposed as

$$v_i = \sum_{i' \neq i} \alpha_{i',i} v_{i'} + v_i^{\perp},$$

where $v_i^{\perp}$ is orthogonal to $\mathrm{span}(V_S \setminus \{v_i\})$. We refer to $v_i^{\perp}$ as the orthogonal component of v_i and use it to define edge weights in the augmented exchange graph denoted by $\widetilde{G}(S)$.

Before we present our algorithm and analyze it, we will define some useful terminology:

Definition 2.3.1 (Orthogonal Projection)**.** Given a linearly independent set $S \subseteq [n]$, define the projection matrix onto $\mathrm{span}(V_S)$ as

$$\mathrm{proj}_S := V_S(V_S^\top V_S)^{-1}V_S^\top,$$

and the orthogonal projection matrix as $\mathrm{proj}_S^\perp := I_d - \mathrm{proj}_S$.

We write $\mathrm{proj}_S^\perp(u)$ for the orthogonal projection of a vector $u \in \mathbb{R}^d$ onto the complement of $\mathrm{span}(V_S)$. When S is clear from context, we abbreviate this as $u^\perp$.

Definition 2.3.2 (Coefficient Extraction)**.** Given a linearly independent set $S \subseteq [n]$, define the function $\mathrm{coef}_S : \mathbb{R}^d \times V_S \to \mathbb{R}$ as follows. For $u_j \in \mathbb{R}^d$ and $v_i \in V_S$, the value $\mathrm{coef}_S(u_j, v_i) = a_{i,j}$ denotes the coefficient of v_i in the decomposition of $u_j - u_j^\perp$ with respect to the basis V_S, where $u_j^\perp = \mathrm{proj}_S^\perp(u_j)$. More concretely,

$$\mathrm{coef}_S(u_j, v_i) = e_i^\top (V_S^\top V_S)^{-1}V_S^\top u_j.$$

Definition 2.3.3 (Augmented Exchange Graph for $r < d$)**.** Let $V_S = \{v_1, \ldots, v_r\}$ be a set of vectors such that S is independent in the matroid $\mathcal{M}$ and $\mathrm{vol}(S) > 0$. The *augmented exchange graph*, denoted $\widetilde{G}(S)$, is a bipartite graph with:

- Right side: the selected vectors V_S,
- Left side: The remaining vectors $U \backslash V_S$.

There is an arc from $v_i \in V_S$ to $u_j \notin V_S$ if the set $S - i + j$ is independent in $\mathcal{M}$. Additionally, if $V_S - v_i + u_j$ is linearly independent, then there are two types of arcs from u_j to v_i, labeled I and II.

Definition 2.3.4 (Weight Functions on the Augmented Exchange Graph)**.** Let $V_S = \{v_1, \ldots, v_r\}$ be a basis of matroid $\mathcal{M}$ with $\mathrm{vol}(S) > 0$, and let $U \setminus V_S = \{u_1, \ldots, u_{n-r}\}$. Suppose each u_j can be written as

$$u_j = \sum_{i=1}^{r} a_{i,j}v_i + u_j^\perp,$$

where $u_j^\perp$ is orthogonal to all vectors in S. For each $v_i \in V_S$, let $v_i^\perp$ denote the projection of v_i orthogonal to $\mathrm{span}(S \setminus \{v_i\})$.

The weights of the forward arcs in $\widetilde{G}(S)$ are defined as:

- For an arc of type I, $u_j \xrightarrow{\text{I}} v_i$, the weight is

$$w(u_j \xrightarrow{\text{I}} v_i) := -\log\left(|a_{i,j}|\right).$$

- For an arc of type II, $u_j \xrightarrow{\text{II}} v_i$, the weight is

$$w(u_j \xrightarrow{\text{II}} v_i) := -\log\left(\frac{\|u_j^\perp\|}{\|v_i^\perp\|}\right).$$

2.3.1 Algorithm

As in previous sections, the first step is to argue that if the current solution has a much smaller objective value compared to the optimum, then there exists a cycle in the exchange graph with significantly negative weight.

Algorithm 3 Algorithm to approximate OPT for $r < d$

Require: Vectors $U = \{v_1, \ldots, v_n\} \subseteq \mathbb{R}^d$, matroid $\mathcal{M} = ([n], \mathcal{I})$ of rank $r < d$
  $S \leftarrow$ any basis of $\mathcal{M}$ with $\mathrm{vol}^2(S) = \det(V_S^\top V_S) > 0$
  while there exists an f-violating cycle in $\widetilde{G}(S)$ **do**
    $C \leftarrow$ minimal f-violating cycle in $\widetilde{G}(S)$
    $S \leftarrow S \triangle C$
  end while
Ensure: S

We now state our main lemma showing the existence of such cycles.

Lemma 2.3.5 (Volume-to-Cycle)**.** *Let T, S be two bases of $\mathcal{M}$ such that*

$$\mathrm{vol}^2(T) \geq \mathrm{vol}^2(S) \cdot (2r)! \cdot f(2r).$$

Then there exists an f-violating cycle C in $\widetilde{G}(S)$, i.e.,

$$\sum_{e \in C} w(e) \leq -\log\left(f(|C|/2)\right).$$

After identifying a minimal f-violating cycle C, we update $S \leftarrow S \triangle C$. When the cycle consists only of type I edges, the volume increase can be bounded similar to the full-rank case $r = d$. However, for general $r < d$, we cannot assume that all cycles avoid type II edges.

Fortunately, we show (Lemma 2.3.11) that any minimal f-violating cycle contains at most one type II edge. We then argue that exchanging on this single II edge affects the structure of the remaining cycle only mildly. This lets us control the change in determinant by bounding the error introduced through this update and leveraging the minimality of the cycle.

These bounds culminate in the following lemma:

Lemma 2.3.6. *Let C be a minimal f-violating cycle in $\widetilde{G}(S)$ and let $T = S \triangle C$. Then $T \in \mathcal{I}$ and $\mathrm{vol}^2(T) > 2 \cdot \mathrm{vol}^2(S)$.*

2.3.2 Useful properties of the volume function

We now describe geometric properties of the volume function that will be used in subsequent lemmas.

Lemma 2.3.7 (Gram–Schmidt Volume Formula)**.** *For $V_S = \{v_1, \ldots, v_r\}$, we have:*

$$\mathrm{vol}(S) = \prod_{t=1}^{r-1} \left\| \mathrm{proj}_{S_t}^{\perp}(v_{t+1}) \right\|,$$

where $S_t := \{v_1, \ldots, v_t\}$.

Proof. Write $\mathrm{vol}(S) = \prod_{t=1}^{r-1} \mathrm{vol}(S_{t+1}) / \mathrm{vol}(S_t)$. It suffices to show that

$$\frac{\mathrm{vol}(S_{t+1})}{\mathrm{vol}(S_t)} = \left\| \mathrm{proj}_{S_t}^{\perp}(v_{t+1}) \right\|.$$

From the definition of $\mathrm{vol}(\cdot)$ (see Section 2.1.1), we have:

$$\frac{\mathrm{vol}^2(S_{t+1})}{\mathrm{vol}^2(S_t)} = \frac{\det(V_{S_{t+1}}^\top V_{S_{t+1}})}{\det(V_{S_t}^\top V_{S_t})}.$$

Using Schur's determinant formula, this equals:

$$v_{t+1}^\top v_{t+1} - v_{t+1}^\top V_{S_t} (V_{S_t}^\top V_{S_t})^{-1} V_{S_t}^\top v_{t+1} = \left\| \mathrm{proj}_{S_t}^{\perp}(v_{t+1}) \right\|^2.$$

□

Corollary 2.3.8. *For $V_S = \{v_1, \ldots, v_r\}$, the following hold:*

1. *Linear Independence: If $v_i = v_{i'}$ for some $i \neq i'$, then $\mathrm{vol}(S) = 0$.*
2. *Triangle Inequality: If $v_i = \alpha v_i^{(0)} + \beta v_i^{(1)}$ for some $v_i \in S$, then*

$$\mathrm{vol}(S) \leq |\alpha| \cdot \mathrm{vol}(S + v_i^{(0)} - v_i) + |\beta| \cdot \mathrm{vol}(S + v_i^{(1)} - v_i).$$

3. *Log-Submodularity: For any $S' \subseteq S$,*

$$\prod_{i \in S \setminus S'} \|v_i^{\perp}\| \leq \frac{\mathrm{vol}(S)}{\mathrm{vol}(S')} \leq \prod_{i \in S \setminus S'} \|v_i\|,$$

where $v_i^{\perp} := \mathrm{proj}_{S-i}^{\perp}(v_i)$.

Proof. 1. W.l.o.g., suppose $i' > i$, and define $S_t := \{v_1, \ldots, v_t\}$. Since $v_i = v_{i'}$, we have $\mathrm{proj}_{S_{i'-1}}^{\perp}(v_{i'}) = 0$. Using Lemma 2.3.7, we conclude $\mathrm{vol}(S) = 0$.

2. Divide both sides by $\mathrm{vol}(S - i)$ and apply Lemma 2.3.7. The result follows from the triangle inequality:

$$\| \mathrm{proj}_{S-i}^{\perp}(v_i) \| \leq |\alpha| \cdot \| \mathrm{proj}_{S-i}^{\perp}(v_i^{(0)}) \| + |\beta| \cdot \| \mathrm{proj}_{S-i}^{\perp}(v_i^{(1)}) \|.$$

3. Let $V_{S'} = \{v_1, \ldots, v_k\}$. Then by Lemma 2.3.7,

$$\frac{\mathrm{vol}(S)}{\mathrm{vol}(S')} = \prod_{i=k}^{r-1} \left\| \mathrm{proj}_{S_i}^{\perp}(v_{i+1}) \right\|.$$

Since projection norms decrease with larger subspaces (i.e., $\mathrm{proj}_T^{\perp}$ is monotonic in T), the bounds follow.

□

Lemma 2.3.9. *Let S be a basis with $\mathrm{vol}(S) > 0$ and let $u_j \notin S$ such that*

$$u_j = \sum_{j \in S} a_{ij} v_i + u_j^{\perp}.$$

Then for any $v_i \in S$, we have:

$$\frac{\mathrm{vol}^2(S + u_j)}{\mathrm{vol}^2(S)} = \|u_j^{\perp}\|^2,$$
$$\frac{\mathrm{vol}^2(S - v_i)}{\mathrm{vol}^2(S)} = \frac{1}{\|v_i^{\perp}\|^2},$$
$$\frac{\mathrm{vol}^2(S + u_j - v_i)}{\mathrm{vol}^2(S)} = a_{ij}^2 + \frac{\|u_j^{\perp}\|^2}{\|v_i^{\perp}\|^2}.$$

2.3.3 Volume to Cycle

Lemma 2.3.5 (Volume-to-Cycle)**.** *Let T, S be two bases of $\mathcal{M}$ such that*

$$\mathrm{vol}^2(T) \geq \mathrm{vol}^2(S) \cdot (2r)! \cdot f(2r).$$

Then there exists an f-violating cycle C in $\widetilde{G}(S)$, i.e.,

$$\sum_{e \in C} w(e) \leq -\log\left(f(|C|/2)\right).$$

Proof. Using Lemma 2.3.10, we have

$$\frac{\mathrm{vol}(T)}{\mathrm{vol}(S)} \leq \sum_{\sigma \in \mathcal{S}_k} \prod_{t=1}^{k} \left(|a_{t,\sigma(t)}| + \frac{\|u_t^{\perp}\|}{\|v_{\sigma(t)}^{\perp}\|} \right),$$

where $a_{i,j} = \mathrm{coef}_S(u_j, v_i)$.

By assumption, $\mathrm{vol}(T)/\mathrm{vol}(S) \geq \sqrt{(2r)! \cdot f(2r)}$. This implies that there exists a permutation $\sigma \in \mathcal{S}_k$ and a subset $P \subseteq [k]$ such that

$$\prod_{t \in P} \frac{\|u_t^{\perp}\|}{\|v_{\sigma(t)}^{\perp}\|} \cdot \prod_{t \in [k] \setminus P} |a_{t,\sigma(t)}| \geq \frac{2^{-r}}{r!} \cdot \sqrt{(2r)! \cdot f(2r)} \geq f(r).$$

Now, consider edges in the exchange graph as follows:

1. For each $t \in [k] \setminus P$, include the type I arc $u_t \xrightarrow{\text{I}} v_{\sigma(t)}$;

2. For each $t \in P$, include the type II arc $u_t \xrightarrow{\text{II}} v_{\sigma(t)}$.

The total weight of these edges is at most $-\log f(r)$. Taking the union of these forward arcs with the backward matching arcs from the constrained matroid $\mathcal{M}$ on $T \Delta S$, we obtain a cycle decomposition.

Since $\log f(\cdot)$ is superlinear, one of these cycles must have total weight at most $-\log f(|C|/2)$, completing the proof. □

We now prove the main technical lemma that was used to prove Lemma 2.3.5:

Lemma 2.3.10. *Let T, S be two bases of $\mathcal{M}$ with $S \setminus T = \{v_1, \ldots, v_k\}$ and $T \setminus S = \{u_1, \ldots, u_k\}$. Then:*

$$\frac{\text{vol}(T)}{\text{vol}(S)} \leq \sum_{\sigma \in \mathcal{S}_k} \prod_{t=1}^{k} \left(|a_{t,\sigma(t)}| + \frac{\|u_t^{\perp}\|}{\|v_{\sigma(t)}^{\perp}\|} \right),$$

where $a_{i,j} = \text{coef}_S(u_j, v_i)$.

Proof. Let $R = S \cap T$. We begin by writing:

$$\text{vol}(T) = \text{vol}(R \cup \{u_1, \ldots, u_k\}).$$

Decompose each u_j as $u_j = u_j^{(0)} + u_j^{(1)}$, where $u_j^{(0)} := \text{proj}_S^{\perp}(u_j)$ and $u_j^{(1)} := \sum_{i \in S} a_{i,j} v_i$. Using Part (2) of Corollary 2.3.8 (linearity of volume), we obtain:

$$\text{vol}(T) \leq \sum_{P \subseteq [k]} \text{vol}\left(R \cup \{u_j^{(0)} : j \in P\} \cup \{u_j^{(1)} : j \in [k] \setminus P\} \right).$$

Then, applying Part (3) of Corollary 2.3.8, we get:

$$\text{vol}(T) \leq \sum_{P \subseteq [k]} \prod_{j \in P} \|u_j^{(0)}\| \cdot \text{vol}\left(R \cup \{u_j^{(1)} : j \in [k] \setminus P\} \right).$$

Next, consider the term:

$$\text{vol}\left(R \cup \{u_j^{(1)}\}_{j \in [k] \setminus P} \right) = \text{vol}\left(R \cup \left\{ \sum_{i \in S} a_{i,j} v_i \right\}_{j \in [k] \setminus P} \right).$$

Using Parts (1) and (2) of Corollary 2.3.8, we can upper-bound this as:

$$\leq \sum_{g : [k] \setminus P \to S} \prod_{j \in [k] \setminus P} |a_{g(j),j}| \cdot \text{vol}(R \cup \{v_{g(j)}\}).$$

Now, only injective maps g that avoid overlap with R contribute nonzero volume. We conservatively sum over all bijections $\sigma \in \mathcal{S}_k$, mapping $T \setminus S$ to $S \setminus T$. Then:

$$\operatorname{vol}\left(R \cup \left\{u_j^{(1)}\right\}_{j\in[k]\setminus P}\right) \leq \sum_{\sigma\in\mathcal{S}_k} \prod_{j\in[k]\setminus P} |a_{\sigma(j),j}| \cdot \operatorname{vol}\left(R \cup \{v_{\sigma(j)}\}_{j\in[k]\setminus P}\right).$$

Again applying Part (3) of Corollary 2.3.8:

$$\leq \operatorname{vol}(S) \cdot \sum_{\sigma\in\mathcal{S}_k} \prod_{j\in[k]\setminus P} |a_{\sigma(j),j}| \cdot \prod_{j\in P} \frac{1}{\|v_{\sigma(j)}^{\perp}\|}.$$

Putting everything together:

$$\begin{aligned}
\operatorname{vol}(T) &\leq \operatorname{vol}(S) \cdot \sum_{P\subseteq[k]} \prod_{j\in P} \|u_j^{\perp}\| \cdot \left(\sum_{\sigma\in\mathcal{S}_k} \prod_{j\in[k]\setminus P} |a_{\sigma(j),j}| \cdot \prod_{j\in P} \frac{1}{\|v_{\sigma(j)}^{\perp}\|}\right) \\
&= \operatorname{vol}(S) \cdot \sum_{\sigma\in\mathcal{S}_k} \sum_{P\subseteq[k]} \prod_{j\in P} \frac{\|u_j^{\perp}\|}{\|v_{\sigma(j)}^{\perp}\|} \cdot \prod_{j\in[k]\setminus P} |a_{\sigma(j),j}| \\
&= \operatorname{vol}(S) \cdot \sum_{\sigma\in\mathcal{S}_k} \prod_{j=1}^{k} \left(|a_{\sigma(j),j}| + \frac{\|u_j^{\perp}\|}{\|v_{\sigma(j)}^{\perp}\|}\right).
\end{aligned}$$

□

2.3.4 Cycle to Volume

In this section, we prove Lemma 2.3.6, which states that exchanging along a minimal f-violating cycle in $\widetilde{G}(S)$ yields a basis whose volume increases by a constant factor. Unlike the previous section, where we relied on the geometric interpretation of the volume function $\operatorname{vol}(\cdot)$, our analysis here is primarily algebraic.

We begin with the observation that any minimal f-violating cycle contains at most one edge of type II (see Lemma 2.3.11). This allows us to break the analysis into two cases.

If the cycle C contains only type I edges, then the argument mirrors the $r = d$ case. Let $T = S \triangle C$. By Corollary 2.7.3, we have

$$\operatorname{vol}^2(T) \geq \operatorname{vol}^2(S) \cdot \det(A)^2,$$

where $V_T = V_S A + V_T^{\perp}$ and $V_T^{\perp} V_S^{\top} = 0$. The nonzero entries of $A_{S\cap C, T\cap C}$ correspond to the weight of type I edges within C. By the minimality of C, bounding $\det(A)$ reduces to analyzing a numeric matrix whose entries depend only on the size of C and the function f. In particular,

$$|\det(A)| \gg 1. \tag{2.3}$$

Now suppose C contains exactly one type II edge $e = (u \xrightarrow{\text{II}} v)$. We first analyze the volume change induced by exchanging v with u, i.e., $\mathrm{vol}^2(S-v+u)/\mathrm{vol}^2(S)$, and then the change from swapping the remaining edges $C \setminus \{e\}$. The key step is showing that the second update behaves approximately as it would under the original basis S:

$$\begin{aligned}\frac{\mathrm{vol}^2(S\triangle C)}{\mathrm{vol}^2(S)} &= \frac{\mathrm{vol}^2(S-v+u)}{\mathrm{vol}^2(S)} \cdot \frac{\mathrm{vol}^2(S\triangle C)}{\mathrm{vol}^2(S-v+u)} \\ &= \frac{\mathrm{vol}^2(S-v+u)}{\mathrm{vol}^2(S)} \cdot \frac{\mathrm{vol}^2((S-v+u)\triangle(C\setminus\{e\}))}{\mathrm{vol}^2(S-v+u)} \\ &\approx \frac{\mathrm{vol}^2(S-v+u)}{\mathrm{vol}^2(S)} \cdot \frac{\mathrm{vol}^2(S\triangle(C\setminus\{e\}))}{\mathrm{vol}^2(S)}. \end{aligned} \tag{2.4}$$

This approximation follows from Lemma 2.3.12, which shows that the coefficient matrix after a type II exchange remains close to the original, due to structural properties of minimal f-violating cycles. Since the involved entries are controlled by edge weights, we maintain good bounds throughout.

2.3.5 Type II Edges

We begin by showing that any minimal f-violating cycle can contain at most one type II edge.

Lemma 2.3.11. *Let C be a minimal f-violating cycle in $\widetilde{G}(S)$. Then C contains at most one edge of type* II.

Proof. Since $\widetilde{G}(S)$ is bipartite, $|C|$ is always even. The lemma is trivially true for the base case when $|C| = 2$, as C contains only one forward arc. Assume $|C| = 2\ell$, and suppose, for contradiction, that C contains two type II arcs at positions k apart:

$$C = (u_1 \xrightarrow{\text{II}} v_1 \to u_2 \xrightarrow{\text{I}} v_2 \to \ldots \to u_k \xrightarrow{\text{II}} v_k \to \ldots \to v_\ell \to u_1),$$

where $v_i \in S$ and the set $S - v_i + u_{i \bmod \ell+1}$ is independent in $\mathcal{M}$ for all $i \in [\ell]$.

Now consider the two cycles formed by switching the type II edges:

$$\begin{aligned} C_1 &= (u_1 \xrightarrow{\text{II}} v_k \to u_{k+1} \xrightarrow{\text{I}} v_{k+2} \to \ldots \to v_\ell \to u_1), \\ C_2 &= (u_k \xrightarrow{\text{II}} v_1 \to u_2 \xrightarrow{\text{I}} v_3 \to \ldots \to v_{k-1} \to u_k). \end{aligned}$$

Rewriting the weights of arcs yields:

$$\begin{aligned} w(u_1 \xrightarrow{\text{II}} v_1) + w(u_k \xrightarrow{\text{II}} v_k) &= -\log\left(\frac{\|u_1^\perp\|}{\|v_1^\perp\|} \cdot \frac{\|u_k^\perp\|}{\|v_k^\perp\|}\right) \\ &= -\log\left(\frac{\|u_1^\perp\|}{\|v_k^\perp\|} \cdot \frac{\|u_k^\perp\|}{\|v_1^\perp\|}\right) \\ &= w(u_1 \xrightarrow{\text{II}} v_k) + w(u_k \xrightarrow{\text{II}} v_1). \end{aligned}$$

Since $u_1 \xrightarrow{\text{II}} v_1$ is a valid arc, it follows that u_1 has a component orthogonal to all columns of V_S, implying $u_1 \xrightarrow{\text{II}} v_k$ is also a valid arc. Similarly, $u_k \xrightarrow{\text{II}} v_1$ is a valid arc. Thus, $\widetilde{G}(S)$ contains the cycles C_1 and C_2. Furthermore, since

$$w(u_1 \xrightarrow{\text{II}} v_1) + w(u_k \xrightarrow{\text{II}} v_k) = w(u_1 \xrightarrow{\text{II}} v_k) + w(u_k \xrightarrow{\text{II}} v_1),$$

we get $w(C) = w(C_1) + w(C_2)$. But since the vertices of C_1 and C_2 are proper subsets of the vertices of C, and C is a minimal f-violating cycle, it must be that C_1 and C_2 are not f-violating. That is,

$$w(C_1) \geq -\log(f(|C_1|/2)) \quad \text{and} \quad w(C_2) \geq -\log(f(|C_2|/2)).$$

Thus,

$$\begin{aligned} w(C) &= w(C_1) + w(C_2) \\ &\geq -\log(f(|C_1|/2)) - \log(f(|C_2|/2)) \\ &> -\log(f(|C|/2)), \end{aligned}$$

where the last inequality follows from the supermultiplicativity of f. This contradicts the assumption that C is f-violating. Therefore, C contains at most one type II arc. □

The following lemma shows that the coefficient function of the basis obtained by exchanging one pair of vectors can be completely characterized in terms of coefficient and projection functions of the current basis. This lemma plays a crucial role in bounding the determinant when the minimal f-violating cycle contains a type II edge.

Lemma 2.3.12 (Coefficient change under one-step exchange)**.** *Let $S \in \mathcal{I}$, and suppose $v_1 \in S$ and $u_1 \notin S$ such that $\mathrm{vol}(S - v_1 + u_1) > 0$. Then for any $v_i \in S$, $i \neq 1$ and $u_j \notin S$, we have*

$$\mathrm{coef}_{S-v_1+u_1}(u_j, v_i) = \mathrm{coef}_S(u_j, v_i) + c_1 x_j y_i + c_2 x_j z_i + c_3 w_j y_i + c_4 w_j z_i,$$

where

$$\begin{aligned} x_j &:= \langle u_j^\perp, u_1^\perp \rangle, & w_j &:= \mathrm{coef}_S(u_j, v_1), \\ y_i &:= \mathrm{coef}_S(u_1, v_i), & z_i &:= \frac{\langle v_i^\perp, v_1^\perp \rangle}{\|v_i^\perp\|^2 \|v_1^\perp\|^2}. \end{aligned}$$

. The constants c_1, c_2, c_3, c_4 depend only on u_1 and v_1, and are given by:

$$\begin{aligned} c_1 &= \frac{-\|v_1^\perp\|^{-2}}{a_{1,1}^2 + \|u_1^\perp\|^2/\|v_1^\perp\|^2}, & c_2 &= \frac{a_{1,1}}{a_{1,1}^2 + \|u_1^\perp\|^2/\|v_1^\perp\|^2}, \\ c_3 &= -c_2, & c_4 &= \frac{-\|u_1^\perp\|^2}{a_{1,1}^2 + \|u_1^\perp\|^2/\|v_1^\perp\|^2}. \end{aligned}$$

(When exchanging u_k, v_k instead of u_1, v_1, replace the index 1 with k everywhere.)

Proof. Let $\widehat{S} = S - v_1 + u_1$. By definition, the coefficient is given by

$$\mathrm{coef}_{\widehat{S}}(u_j, v_i) = u_j^\top V_{\widehat{S}} (V_{\widehat{S}}^\top V_{\widehat{S}})^{-1} e_i.$$

Let $X := [v_2\ v_3\ \cdots\ v_\ell]$ be the matrix of remaining vectors in S, so that $V_{\widehat{S}} = [u_1\ X]$. Then

$$V_{\widehat{S}}^\top V_{\widehat{S}} = \begin{bmatrix} u_1^\top u_1 & u_1^\top X \\ X^\top u_1 & X^\top X \end{bmatrix}.$$

Applying the block inversion formula, we obtain:

$$V_{\widehat{S}}(V_{\widehat{S}}^\top V_{\widehat{S}})^{-1} = \begin{bmatrix} \frac{\operatorname{proj}_X^\perp(u_1)}{\|\operatorname{proj}_X^\perp(u_1)\|^2} & X(X^\top X)^{-1} - \frac{\operatorname{proj}_X^\perp(u_1) u_1^\top X (X^\top X)^{-1}}{\|\operatorname{proj}_X^\perp(u_1)\|^2} \end{bmatrix}.$$

For $i \neq 1$, let $a_{i,j} := \operatorname{coef}_S(u_j, v_i)$. Then

$$\operatorname{coef}_{\widehat{S}}(u_j, v_i) = u_j^\top X(X^\top X)^{-1} e_i - \frac{\langle u_j, \operatorname{proj}_X^\perp(u_1)\rangle \cdot \langle u_1, X(X^\top X)^{-1} e_i\rangle}{\|\operatorname{proj}_X^\perp(u_1)\|^2}. \tag{2.5}$$

We use the identities:

$$\operatorname{proj}_X^\perp(u_1) = u_1^\perp + a_{1,1} v_1^\perp, \qquad \|\operatorname{proj}_X^\perp(u_1)\|^2 = \|u_1^\perp\|^2 + a_{1,1}^2 \|v_1^\perp\|^2,$$

and

$$u_j^\top X(X^\top X)^{-1} e_i = \operatorname{coef}_X(u_j, v_i) = a_{i,j} - a_{1,j} \cdot \frac{\langle v_i^\perp, v_1^\perp\rangle}{\|v_i^\perp\|^2}.$$

Plugging these into (2.5), we get:

$$\operatorname{coef}_{\widehat{S}}(u_j, v_i) = a_{i,j} - a_{1,j} \cdot \frac{\langle v_i^\perp, v_1^\perp\rangle}{\|v_i^\perp\|^2} - \frac{\left(\langle u_j, u_1\rangle + a_{1,1} a_{1,j} \|v_1^\perp\|^2\right) \cdot \left(a_{i,1} - a_{1,1} \cdot \frac{\langle v_i^\perp, v_1^\perp\rangle}{\|v_i^\perp\|^2}\right)}{\|u_1^\perp\|^2 + a_{1,1}^2 \|v_1^\perp\|^2}.$$

Grouping terms in terms of inner products and coefficient expressions gives the desired form:

$$\operatorname{coef}_{\widehat{S}}(u_j, v_i) = \operatorname{coef}_S(u_j, v_i) + c_1 x_j y_i + c_2 x_j z_i + c_3 w_j y_i + c_4 w_j z_i.$$

□

The following observation gives useful upper bounds on the magnitudes of the terms x_j, w_j, y_i, z_i, and the coefficients c_1, c_2, c_3, c_4 from Lemma 2.3.12, which we will use later.

Observation 2.3.13 (Error term magnitudes)**.** We have:

$$|x_j| \le \|u_j^\perp\| \cdot \|u_1^\perp\|, \qquad |w_j| = |a_{1,j}|,$$
$$|y_i| = |a_{i,1}|, \qquad |z_i| \le \frac{1}{\|v_i^\perp\| \cdot \|v_1^\perp\|}.$$

For the coefficients:

$$|c_1| \le \frac{1}{\|u_1^\perp\|^2}, \qquad |c_2| \le \frac{\|v_1^\perp\|}{2\|u_1^\perp\|},$$
$$|c_3| \le \frac{\|v_1^\perp\|}{2\|u_1^\perp\|}, \qquad |c_4| \le \|v_1^\perp\|^2.$$

Proof. The bounds for x_j and z_i follow from the Cauchy–Schwarz inequality. For c_1 and c_4, we obtain upper bounds by setting $a_{1,1} = 0$ to minimize the denominator. For c_2 and c_3, we treat $a_{1,1}$ as a variable and maximize the expression over its possible values to upper bound the coefficients. □

Definition 2.3.14 (Weight Matrix)**.** Given a cycle $C = (u_1 \to v_1 \to \cdots \to u_\ell \to v_\ell \to u_1)$ in $\widetilde{G}(S)$, define the weight matrix $B^C \in \mathbb{R}^{\ell \times \ell}$ such that :

$$B^C_{i,j} := \begin{cases} \max\{|a_{i,j}|, \|u_j^\perp\|/\|v_i^\perp\|\} & \text{if } i \neq j, \\ |a_{i,i}| & \text{if } u_i \to v_i \text{ is type I}, \\ \|u_i^\perp\|/\|v_i^\perp\| & \text{if } u_i \to v_i \text{ is type II}. \end{cases}$$

Lemma 2.3.15 (Weight matrix with minimal f-violation)**.** *Let*

$$C = (u_1 \xrightarrow{\text{I}} v_1 \to u_2 \xrightarrow{\text{I}} v_2 \to \cdots \to u_\ell \xrightarrow{*} v_\ell \to u_1)$$

be a minimal f-violating cycle in $\widetilde{G}(S)$, with $|C| = 2\ell$, and associated weight matrix $B := B^C \in \mathbb{R}^{\ell \times \ell}$ as defined in Definition 2.3.14.

Assume that each $u_i \xrightarrow{\text{I}} v_i$ for $i = 1, \ldots, \ell - 1$ is a type I arc, while the final arc $u_\ell \xrightarrow{*} v_\ell$ may be of type I or type II.

Then:

(a) $\displaystyle\prod_{i=1}^{\ell} b_{i,i} \geq f(\ell)$,

(b) $\displaystyle\operatorname{per}(B) \leq \left(1 + \frac{0.1}{\ell}\right) \prod_{i=1}^{\ell} b_{i,i}$,

(c) $\displaystyle\operatorname{per}(B_{\ell-1,\ell-1}) \leq 1.05 \cdot \prod_{i=1}^{\ell-1} b_{i,i}$,

where $b_{i,j} := B_{i,j}$ and $B_{\ell-1,\ell-1} := B_{[\ell-1],[\ell-1]}$ is the principal $(\ell-1) \times (\ell-1)$ submatrix.

The proof mirrors that of Lemma 2.2.12, and is deferred to § 2.7.2.

We are finally ready to prove Lemma 2.3.5. We break the proof into two parts based on the number of edges in the minimal f-violating cycle C: first if C only contains type I edges, then the proof is similar to $r = d$ case, see Lemma 2.3.16. If C contains a type II edge, then Lemma 2.3.17 shows that the determinant still doubles.

Proof of Lemma 2.3.5. By Lemma 2.3.11, any minimal f-violating cycle C contains at most one type II edge.

- If C contains only type I edges, then Lemma 2.3.16 implies $\operatorname{vol}^2(T) > 2 \cdot \operatorname{vol}^2(S)$.

- If C contains exactly one type II edge, then Lemma 2.3.17 shows the same.

Independence of T follows structurally from Lemma 2.2.14, as in the $r = d$ case. □

Lemma 2.3.16. *Let C be a minimal f-violating cycle in $\widetilde{G}(S)$ containing only type* I *edges, and let $T = S \triangle C$. Then:*

$$\mathrm{vol}^2(T) > 2 \cdot \mathrm{vol}^2(S).$$

Proof. Let $X := T \setminus S = \{u_1, \dots, u_\ell\}$ and $Y := S \setminus T = \{v_1, \dots, v_\ell\}$, ordered according to the cycle. Since C consists only of type I edges, every arc $u_i \xrightarrow{\text{I}} v_i$ contributes an entry $a_{i,i}$ in the coefficient matrix $A \in \mathbb{R}^{\ell \times \ell}$, where:

$$A_{i,j} := \mathrm{coef}_S(u_j, v_i).$$

By Corollary 2.7.3:

$$\mathrm{vol}^2(T) \geq \mathrm{vol}^2(S) \cdot \det(A)^2.$$

Now apply Lemma 2.3.15:

1. $\prod_{i=1}^{\ell} |a_{i,i}| \geq f(\ell)$,
2. $\mathrm{per}(A) \leq \left(1 + \frac{0.1}{\ell}\right) \cdot \prod_{i=1}^{\ell} |a_{i,i}|$.

Hence,

$$|\det(A)| \geq 2 \prod_{i=1}^{\ell} |a_{i,i}| - \mathrm{per}(A) \geq 0.95 \cdot \prod_{i=1}^{\ell} |a_{i,i}|,$$

and since $f(\ell) > 2.1$ for $\ell \geq 2$, we get:

$$\det(A)^2 > 4.2 \quad \Rightarrow \quad \mathrm{vol}^2(T) > 2 \cdot \mathrm{vol}^2(S).$$

For $\ell = 1$, this is immediate from $|a_{1,1}| \geq f(1) = 2$. □

Lemma 2.3.17. *Let C be a minimal f-violating cycle in $\widetilde{G}(S)$ containing exactly one type* II *edge, and let $T = S \triangle C$. Then* $\mathrm{vol}^2(T) > 2 \cdot \mathrm{vol}^2(S)$.

Proof. Let $C = (u_1 \xrightarrow{\text{I}} v_1 \to \cdots \to u_\ell \xrightarrow{\text{II}} v_\ell \to u_1)$, and define:

$$X := T \setminus S = \{u_1, \dots, u_\ell\}, \quad Y := S \setminus T = \{v_1, \dots, v_\ell\}.$$

Let $\widehat{S} := S - v_\ell + u_\ell$ denote the intermediate set after exchanging along the single type II arc $(u_\ell \xrightarrow{\text{II}} v_\ell)$. Also let $B := B^C$ be the weight matrix defined in Definition 2.3.14 and Lemma 2.3.15. By Lemma 2.3.9 we have:

$$\mathrm{vol}(\widehat{S}) = \mathrm{vol}(S) \left(a_{\ell,\ell}^2 + \frac{\left\| u_\ell^\perp \right\|^2}{\left\| v_\ell^\perp \right\|^2} \right) \geq \mathrm{vol}(S) \cdot \frac{\left\| u_\ell^\perp \right\|^2}{\left\| v_\ell^\perp \right\|^2} =: \mathrm{vol}(S) \cdot b_{\ell,\ell}^2. \tag{2.6}$$

If $\ell = 1$, this immediately gives $\mathrm{vol}^2(T) = \mathrm{vol}^2(\widehat{S}) > f(1)^2 \,\mathrm{vol}^2(S) > 4\,\mathrm{vol}^2(S)$. Hence assume $\ell \geq 2$ from here on.

We now consider the remaining updates from $\widehat{S}$ to T, which correspond to swapping in $u_1, \ldots, u_{\ell-1}$ for $v_1, \ldots, v_{\ell-1}$. Define $\widehat{X} := X \setminus \{u_\ell\}$ and $\widehat{Y} := Y \setminus \{v_\ell\}$.

By Corollary 2.7.3:

$$\frac{\mathrm{vol}^2(T)}{\mathrm{vol}^2(\widehat{S})} \geq \det(A_{\widehat{S}})^2,$$

where $A_{\widehat{S}}$ is the $\ell - 1 \times \ell - 1$ matrix with entries $\mathrm{coef}_{\widehat{S}}(u_j, v_i)$ for $u_j \in \widehat{X}$ and $v_i \in \widehat{Y}$. Let A_S be the $\ell \times \ell$ matrix with entries $\mathrm{coef}_S(u_j, v_i)$ for $u_j \in X$ and $v_i \in Y$.

Let A be the submatrix of A_S indexed by rows $\widehat{Y}$ and columns $\widehat{X}$. By Lemma 2.3.12, we can write:

$$A_{\widehat{S}} = A + Q, \tag{2.7}$$

where $Q = c_1 y x^\top + c_2 z x^\top + c_3 y w^\top + c_4 z w^\top$ is a structured rank-2 update, and $c_1, \ldots, c_4$ are constants independent of u_j and v_i.

By Lemma 2.3.19, we have :

$$|\det(A_{\widehat{S}})| \geq 0.3 \cdot \prod_{i=1}^{\ell-1} b_{i,i}.$$

Hence,

$$\frac{\mathrm{vol}^2(T)}{\mathrm{vol}^2(\widehat{S})} \geq 0.09 \cdot \prod_{i=1}^{\ell-1} b_{i,i}^2. \quad \text{(square of above)}$$

Combining this with (2.6), we conclude:

$$\frac{\mathrm{vol}^2(T)}{\mathrm{vol}^2(S)} = \frac{\mathrm{vol}^2(T)}{\mathrm{vol}^2(\widehat{S})} \cdot \frac{\mathrm{vol}^2(\widehat{S})}{\mathrm{vol}^2(S)} \geq 0.09 \cdot \prod_{i=1}^{\ell} b_{i,i}^2.$$

Since C is f-violating, by Lemma 2.3.15 we have $\prod_{i=1}^{\ell} b_{i,i} \geq f(\ell)$, and since $\ell \geq 2$, $f(\ell) > 2^6$. Hence:

$$\mathrm{vol}^2(T) > 0.09 \cdot f(\ell)^2 \cdot \mathrm{vol}^2(S) > 2 \cdot \mathrm{vol}^2(S).$$

In order to complete the proof, it suffices to prove the lower bound on $\det(A_{\widehat{S}})$. In order to do this, we need to first understand the structure of the matrices A and Q.

Lemma 2.3.18 (Properties of A and Q)**.** *Let A and Q be as defined in Equation* (2.7)*. Then:*

1. *The entries of A satisfy $|A_{i,j}| = |\mathrm{coef}_S(u_j, v_i)| \leq b_{i,j}$ for all $i, j \in [\ell - 1]$, with equality $|A_{i,i}| = b_{i,i}$.*

2. *The determinant of A is lower bounded as*

$$|\det(A)| \geq 0.95 \prod_{i=1}^{\ell-1} b_{i,i}.$$

3. *The entries of Q satisfy*

$$|c_1 y_i x_j|,\ |c_2 z_i x_j|,\ |c_3 y_i w_j|,\ |c_4 z_i w_j| \leq \frac{b_{i,\ell} b_{\ell,j}}{b_{\ell,\ell}}.$$

In particular,

$$|Q_{i,j}| \leq 4 \cdot \frac{b_{i,\ell} b_{\ell,j}}{b_{\ell,\ell}}.$$

4. *For cross terms, we have*

$$|c_1 c_4 y_i x_j z_{i'} w_{j'}| \leq \frac{b_{i,\ell} b_{i',j} b_{\ell,j'}}{b_{\ell,\ell}}, \quad |c_2 c_3 z_i x_j y_{i'} w_{j'}| \leq \frac{b_{i,j} b_{i',\ell} b_{\ell,j'}}{b_{\ell,\ell}}.$$

Proof of Lemma 2.3.18. 1. This follows directly from the definitions of A and B.

2. We use the bound:

$$|\det(A)| \geq 2 \prod_{s=1}^{\ell-1} a_{s,s} - \operatorname{per}(A) \geq 2 \prod_{s=1}^{\ell-1} b_{s,s} - \operatorname{per}(B_{[\ell-1],[\ell-1]}) \geq 0.95 \prod_{s=1}^{\ell-1} b_{s,s},$$

where the last inequality follows from part (c) of Lemma 2.3.15.

3. From Observation 2.3.13, we have

$$|c_1 y_i x_j| \leq |a_{i,\ell}| \cdot \frac{\|u_j^\perp\|}{\|u_\ell^\perp\|} = |a_{i,\ell}| \cdot \frac{\|u_j^\perp\|/\|v_\ell^\perp\|}{\|u_\ell^\perp\|/\|v_\ell^\perp\|} \leq \frac{b_{i,\ell} b_{\ell,j}}{b_{\ell,\ell}}.$$

The last step uses the definition of B. The same reasoning applies to the other terms.

4. Using similar bounds, we get

$$|c_1 c_4 y_i x_j z_{i'} w_{j'}| \leq |a_{i,\ell}| \cdot |a_{\ell,j'}| \cdot \frac{\|v_\ell^\perp\|}{\|u_\ell^\perp\|} \cdot \frac{\|u_j^\perp\|}{\|v_{i'}^\perp\|} \leq \frac{b_{i,\ell} b_{i',j} b_{\ell,j'}}{b_{\ell,\ell}},$$

and similarly for $|c_2 c_3 z_i x_j y_{i'} w_{j'}|$.

□

Lemma 2.3.19 (Determinant lower bound for $\widehat{A}$)**.** *We have*

$$|\det(\widehat{A})| \geq 0.3 \prod_{i=1}^{\ell-1} b_{i,i}.$$

Proof. We apply Lemma 2.7.5 to expand $\det(\widehat{A}) = \det(A+Q)$ using multilinearity and the triangle inequality:

$$\begin{aligned}|\det(\widehat{A})| \geq \underbrace{|\det(A)|}_{\mathcal{T}1} - \underbrace{\sum_{j=1}^{\ell-1} \left|\det\left(A^{(1)}, \ldots, Q^{(j)}, \ldots, A^{(\ell-1)}\right)\right|}_{\mathcal{T}2} \\ \underbrace{- \sum_{\substack{j,j' \in [\ell-1] \\ j \neq j'}} \left|\det\left(A^{(1)}, \ldots, c_1 y x_j, \ldots, c_4 z w_{j'}, \ldots, A^{(\ell-1)}\right)\right|}_{\mathcal{T}3} \\ \underbrace{- \sum_{\substack{j,j' \in [\ell-1] \\ j \neq j'}} \left|\det\left(A^{(1)}, \ldots, c_2 z x_j, \ldots, c_3 y w_{j'}, \ldots, A^{(\ell-1)}\right)\right|}_{\mathcal{T}4}. \end{aligned} \tag{2.8}$$

By part (2) of Lemma 2.3.18, we have $\mathcal{T}1 \geq 0.95 \prod_{i=1}^{\ell-1} b_{i,i}$.

We now show that:

$$\mathcal{T}2 \leq 0.4 \prod_{i=1}^{\ell-1} b_{i,i}, \qquad \mathcal{T}3, \mathcal{T}4 \leq 0.1 \prod_{i=1}^{\ell-1} b_{i,i}.$$

By expanding each determinant in $\mathcal{T}2$:

$$\mathcal{T}2 \leq \sum_{j=1}^{\ell-1} \sum_{\sigma \in \mathcal{S}_{\ell-1}} \left(\prod_{t \neq j} |a_{\sigma(t),t}| \right) \cdot |Q_{\sigma(j),j}| \tag{2.9}$$

Using the definition of B and the bound on $Q_{i,j}$ in part (3) of Lemma 2.3.18,

$$\leq \sum_{j=1}^{\ell-1} \sum_{\sigma \in \mathcal{S}_{\ell-1}} \left(\prod_{t \neq j} b_{\sigma(t),t} \right) \cdot \left(4 \cdot \frac{b_{\sigma(j),\ell} \cdot b_{\ell,j}}{b_{\ell,\ell}} \right) \tag{2.10}$$

$$= \frac{4}{b_{\ell,\ell}} \sum_{j=1}^{\ell-1} \sum_{\sigma \in \mathcal{S}_{\ell-1}} b_{\sigma(j),\ell} b_{\ell,j} \prod_{t \neq j} b_{\sigma(t),t}. \tag{2.11}$$

We now re-index this sum by defining a permutation $\delta \in \mathcal{S}_\ell$ associated with each pair (σ, j): Set $\delta(j) = \ell$, $\delta(\ell) = \sigma(j)$, and $\delta(t) = \sigma(t)$ for all $t \in [\ell-1] \setminus \{j\}$.

Then:

$$b_{\sigma(j),\ell} b_{\ell,j} \prod_{t \neq j} b_{\sigma(t),t} = \prod_{i=1}^{\ell} b_{\delta(i),i},$$

and every such δ is distinct and satisfies $\delta(\ell) \neq \ell$. Therefore,

$$\sum_{j \in [\ell-1], \sigma \in \mathcal{S}_{\ell-1}} b_{\sigma(j),\ell} b_{\ell,j} \prod_{t \neq j} b_{\sigma(t),t} \leq \operatorname{perm}(B) - \prod_{i=1}^{\ell} b_{i,i}.$$

So:

$$\mathcal{T}2 \leq \frac{4}{b_{\ell,\ell}} \left(\operatorname{perm}(B) - \prod_{i=1}^{\ell} b_{i,i} \right) \leq 0.4 \prod_{i=1}^{\ell-1} b_{i,i}, \tag{2.12}$$

where the last step uses part (b) of Lemma 2.3.15.

In a similar style, consider the term $\mathcal{T}_3$:

$$\mathcal{T}_3 \leq \sum_{\substack{j,j' \in [\ell-1] \\ j \neq j'}} \sum_{\sigma \in \mathcal{S}_{\ell-1}} \left(\prod_{t \in [\ell-1] \setminus \{j,j'\}} |a_{\sigma(t),t}| \right) \cdot |c_1 c_4 y_{\sigma(j)} x_j z_{\sigma(j')} w_{j'}| \tag{2.13}$$

$$\leq \sum_{\substack{j,j' \in [\ell-1] \\ j \neq j'}} \sum_{\sigma \in \mathcal{S}_{\ell-1}} \left(\prod_{t \in [\ell-1] \setminus \{j,j'\}} b_{\sigma(t),t} \right) \cdot \frac{b_{\sigma(j),\ell} b_{\sigma(j'),j} b_{\ell,j'}}{b_{\ell,\ell}}, \tag{2.14}$$

where the second inequality uses part (4) of Lemma 2.3.18.

As before, we associate each triple (σ, j, j') with $j \neq j'$ to a permutation $\delta \in \mathcal{S}_\ell$ by setting:

$$\delta(j') = \ell, \quad \delta(j) = \sigma(j'), \quad \delta(\ell) = \sigma(j), \quad \delta(t) = \sigma(t) \text{ for } t \in [\ell - 1] \setminus \{j, j'\}.$$

Then:

$$b_{\sigma(j),\ell} b_{\sigma(j'),j} b_{\ell,j'} \prod_{t \in [\ell-1] \setminus \{j,j'\}} b_{\sigma(t),t} = \prod_{i \in [\ell]} b_{\delta(i),i}.$$

Each such δ satisfies $\delta(\ell) \neq \ell$. Unlike the case for $\mathcal{T}_2$, this mapping is not injective. However, for any fixed j and δ, the triple (σ, j, j') can be uniquely recovered. Hence, each δ appears at most ℓ times, leading to the bound:

$$\sum_{\substack{j,j' \in [\ell-1] \\ j \neq j'}} \sum_{\sigma \in \mathcal{S}_{\ell-1}} \left(\prod_{t \in [\ell-1] \setminus \{j,j'\}} b_{\sigma(t),t} \right) \cdot b_{\sigma(j),\ell} b_{\sigma(j'),j} b_{\ell,j'} \leq \ell \cdot \left(\operatorname{perm}(B) - \prod_{i \in [\ell]} b_{i,i} \right).$$

Using part (b) of Lemma 2.3.15, we conclude:

$$\mathcal{T}_3 \leq 0.1 \prod_{i \in [\ell-1]} b_{i,i}.$$

The bound for $\mathcal{T}_4$ follows by the same argument, completing the analysis. Thus,

$$|\det(\widehat{A})| \geq (0.95 - 0.4 - 0.1 - 0.1) \prod_{i \in [\ell-1]} b_{i,i} \geq 0.3 \prod_{i \in [\ell-1]} b_{i,i}.$$

□

□

2.4 Rank greater than dimension

In the cases $r = d$ and $r < d$ (see § 2.2 and § 2.3), the edge weight in the exchange graph from a vertex $u_j \in U \setminus V_S$ to a vertex $v_i \in V_S$ is given by $-\log \frac{\mathrm{vol}(S-i+j)}{\mathrm{vol}(S)}$. That is, the edge represents the change in negative log-volume when replacing v_i with u_j in the current solution.

While the change in negative log-volume from a cycle exchange does not exactly match the weight of the cycle, we previously showed (under mild conditions) that the change in log-volume is inversely proportional to the weight of the cycle. Thus, a short cycle implies a significant increase in volume.

Now consider the case $r > d$. We want the arc weight from u_j to v_i to reflect the change in the objective function when v_i is replaced by u_j. Though the objective function differs from the $r < d$ case, the underlying ideas and analysis are similar.

For $r > d$, the change in the objective is:

$$w(u_j \to v_i) = -\frac{1}{2} \log \frac{\det\left(V_S V_S^\top - v_i v_i^\top + u_j u_j^\top\right)}{\det(V_S V_S^\top)}.$$

Using the matrix determinant lemma, we simplify the ratio:

$$\begin{aligned}\frac{\det\left(V_S V_S^\top - v_i v_i^\top + u_j u_j^\top\right)}{\det(V_S V_S^\top)} &= \underbrace{\left(u_j^\top (V_S V_S^\top)^{-1} v_i\right)^2}_{\text{Term I}} \\ &\quad + \underbrace{\left(1 + u_j^\top (V_S V_S^\top)^{-1} u_j\right)\left(1 - v_i^\top (V_S V_S^\top)^{-1} v_i\right)}_{\text{Term II}}. \end{aligned} \tag{2.15}$$

Note:

$$\begin{aligned}\frac{\det(V_S V_S^\top + u_j u_j^\top)}{\det(V_S V_S^\top)} &= 1 + u_j^\top (V_S V_S^\top)^{-1} u_j, \\ \frac{\det(V_S V_S^\top - v_i v_i^\top)}{\det(V_S V_S^\top)} &= 1 - v_i^\top (V_S V_S^\top)^{-1} v_i.\end{aligned}$$

Thus, Term II is the product of the individual marginal changes, while Term I captures the interaction between adding u_j and removing v_i.

2.4.1 Extended Exchange Graph

To capture these two components separately, we define two forward arcs $u_j \xrightarrow{\text{I}} v_i$ and $u_j \xrightarrow{\text{II}} v_i$ in the exchange graph $\widetilde{G}(S)$ for each $u_j \notin V_S$, $v_i \in V_S$.

We assign:

- weight $-\log\left(\left|u_j^\top (V_S V_S^\top)^{-1} v_i\right|\right)$ to arcs of type I,
- weight $-\log\sqrt{(1+u_j^\top (V_S V_S^\top)^{-1} u_j)(1-v_i^\top (V_S V_S^\top)^{-1} v_i)}$ to arcs of type II.

Definition 2.4.1 (Augmented Exchange Graph for $r > d$)**.** Let $V_S = \{v_1, \ldots, v_r\}$ be a basis in $\mathcal{M}$ with $\det(V_S V_S^\top) > 0$. The exchange graph $\widetilde{G}(S)$ is a bipartite graph: the right part consists of vectors in V_S, and the left part consists of all other vectors. There is an arc from $v_i \in V_S$ to $u_j \notin V_S$ if $S - i + j$ is independent in $\mathcal{M}$. If $S - i + j$ spans $\mathbb{R}^d$, then two forward arcs ($\overset{\text{I}}{\to}$ and $\overset{\text{II}}{\to}$) are added from u_j to v_i.

Definition 2.4.2 (Weight Functions)**.** In $\widetilde{G}(S)$:

- Backward arcs $v_i \to u_j$ (i.e., $v_i \in V_S$, $u_j \notin V_S$) have weight 0.
- Type I forward arcs have weight $w(u_j \overset{\text{I}}{\to} v_i) = -\log\left|u_j^\top (V_S V_S^\top)^{-1} v_i\right|$.
- Type II forward arcs have weight

$$w(u_j \overset{\text{II}}{\to} v_i) = -\log\sqrt{\left(1+u_j^\top (V_S V_S^\top)^{-1} u_j\right)\left(1-v_i^\top (V_S V_S^\top)^{-1} v_i\right)}.$$

Definition 2.4.3 (f-Violating Cycle)**.** A cycle C in $\widetilde{G}(S)$ is called f-violating if:

$$w(C) < -\log f(|C|/2).$$

2.4.2 Algorithm

As before, we argue that if the objective for the current solution S is small compared to the optimum, then $\widetilde{G}(S)$ must contain an f-violating cycle.

Algorithm 4 Approximation Algorithm for $r > d$

Require: Vectors $U = \{v_1, \ldots, v_n\} \subset \mathbb{R}^d$, matroid $\mathcal{M} = ([n], \mathcal{I})$ with rank $= r > d$, and optimal solution $\widehat{x}$ to (CP)
  $E \leftarrow \operatorname{supp}(\widehat{x})$
  $\mathcal{M}_E \leftarrow (E, \mathcal{I}_{|E|})$ ▷ Matroid restricted to support
  $S \leftarrow$ a basis of $\mathcal{M}_E$ with $\det(V_S V_S^\top) > 0$
  while an f-violating cycle exists in $\widetilde{G}(S)$ **do**
    $C \leftarrow$ minimal f-violating cycle
    $S \leftarrow S \triangle C$
  end while
  return S

A key technical result supporting the algorithm is the following lemma:

Lemma 2.4.4. *Let T and S be bases of $\mathcal{M}$ with $|T \setminus S| = \ell$, and assume*

$$\det(V_T V_T^\top) \geq \det(V_S V_S^\top) \cdot (2\ell)! \cdot f(2\ell).$$

Then, there exists an f-violating cycle C in $\widetilde{G}(S)$.

While Lemma 2.4.4 appears structurally similar to its counterparts for $r \leq d$, its proof involves significant technical challenges when $r > d$.

In the $r \leq d$ case, arc weights in $\widetilde{G}(S)$ correspond to entries of the transformation matrix expressing vectors in the optimal solution in terms of the current basis. The hypothesis in Lemma corresponding to Lemma 2.4.4 for the $r \leq d$ cases then implies that this transformation has a large determinant, from which the existence of an f-violating cycle follows.

However, for $r > d$, this approach fails because such a transformation matrix is no longer uniquely defined: there are $\binom{r}{d}$ ways to select a basis from a set of r vectors in $\mathbb{R}^d$. Consequently, the arc weights in $\widetilde{G}(S)$ do not directly correspond to any fixed transformation matrix.

To address this, we define a different matrix—derived using the matrix determinant lemma—whose determinant is guaranteed to be large under the lemma's hypothesis. Although the entries of this matrix do not equal the arc weights, they are closely related. We then leverage structural properties of arc weights specific to the $r > d$ case to upper bound this determinant in terms of the cycle weight.

We begin by computing a sparse fractional solution $\widehat{x}$ to (CP), and restrict our matroid to its support. The algorithm then proceeds in a manner analogous to the previous cases.

After identifying a minimal f-violating cycle C, we update the solution by exchanging along C: i.e., we set $S \leftarrow S \triangle C$.

We can show that if all edges in C are of type I, then the determinant increases proportionally to the inverse-exponential of the cycle weight, as in the case $r \leq d$.

One may ask whether we can safely ignore type II edges altogether and still guarantee the existence of such a cycle. Unfortunately, the answer is no. It is possible for the current solution S to be much worse than the optimum, while every negative-weight cycle in $\widetilde{G}(S)$ includes at least one type II edge. We provide such an example in § 2.7.3.

Despite this, we prove that any minimal f-violating cycle contains at most one type II edge (see Lemma 2.3.11). This fact is crucial: it ensures that exchanging along a type II edge causes only a localized change in the determinant and leaves the type I arc weights essentially unaffected elsewhere in the cycle.

This allows us to control the perturbation introduced by the type II edge. By exploiting the minimality of the cycle, we can carefully bound the additional error terms and relate the change in determinant to the weights of the cycle.

This yields the following result:

Theorem 2.4.5. *Let C be a minimal f-violating cycle in $\widetilde{G}(S)$, and let $T = S \triangle C$. Then $T \in \mathcal{I}$ and:*

$$\det(V_T V_T^\top) > 2 \cdot \det(V_S V_S^\top).$$

Applying Lemma 2.4.4 and Theorem 2.4.5 directly yields an approximation factor of $O(r^r)$ when $r > d$.

To improve this to $O(d)^{O(d)}$, we must ensure the existence of an f-violating cycle with only $O(d)$ edges. This is guaranteed if the ground set of $\mathcal{M}$ has size at most $r + \text{poly}(d)$. While this assumption may seem restrictive, we overcome it by employing a convex relaxation that produces a sparse support set, allowing us to restrict attention to a small subset of the ground set.

We now describe this convex program.

2.4.3 Convex Program

Let $\mathcal{M} = ([n], \mathcal{I})$ be the constraint matroid, and denote by $\mathcal{I}_s(\mathcal{M}) := \{S \in \mathcal{I} : |S| = s\}$ the independent sets of size s. Let $\mathcal{P}(\mathcal{M})$ be the matroid base polytope (the convex hull of all bases of $\mathcal{M}$).

Define:

$$\mathcal{Z} = \left\{ z \in \mathbb{R}^n : z(S) := \sum_{i \in S} z_i \geq 0 \quad \forall S \in \mathcal{I}_d(\mathcal{M}) \right\}.$$

Following [131], consider the convex program:

$$\sup_{x \in \mathcal{P}(\mathcal{M})} \inf_{z \in \mathcal{Z}} g(x, z) := \log \det \left(\sum_{i \in [n]} x_i e^{z_i} v_i v_i^\top \right). \tag{CP}$$

This convex program is a relaxation of the determinant maximization problem. Denote by OPT_{CP} the optimum of this relaxation, and by OPT the optimal value of the original (combinatorial) determinant maximization problem.

Theorem 2.4.6 ([131])**.** *For $r > d$, there exists a polynomial-time algorithm that computes an optimal solution $\widehat{x}$ to* (CP) *such that:*

$$|\operatorname{supp}(\widehat{x})| \leq r + 2\left(\binom{d+1}{2} + d \right).$$

Moreover, there exists a basis $T \subseteq \operatorname{supp}(\widehat{x})$ satisfying:

$$\det \left(\sum_{i \in T} v_i v_i^\top \right) \geq (2e^5 d)^{-d} \cdot \text{OPT}.$$

Although the existence of T is not explicitly shown in [131], their randomized rounding procedure implies it in expectation (see their Theorem 2.3). For our algorithm, only the existential guarantee is needed.

Lemma 2.4.7 (Existence of Short f-Violating Cycle)**.** *Assume* $\operatorname{rank}(\mathcal{M}) = r > d$ *and that the ground set of* $\mathcal{M}$ *contains* $r+k$ *elements. Let* S *be any basis with* $\det(V_S V_S^\top) > 0$*. If there exists a basis* S_1 *such that:*

$$\det(V_{S_1} V_{S_1}^\top) \geq \det(V_S V_S^\top) \cdot d \cdot d! \cdot (4k)^d \cdot f(2d),$$

then there exists a basis T *such that* $|T \triangle S| \leq 2d$ *and:*

$$\det(V_T V_T^\top) \geq (2d)! \cdot f(2d) \cdot \det(V_S V_S^\top).$$

Implying the existence of an f*-violating cycle using Lemma 2.4.4.*

Thus, before applying Theorem 2.4.5, we compute a sparse support E using Theorem 2.4.6, and define a restricted matroid $\mathcal{M}_E$ over E:

$$\mathcal{M}_E = (E, \mathcal{I}_E), \quad \text{where } \mathcal{I}_E = \{I \subseteq E \mid I \in \mathcal{I}\}.$$

2.4.4 Main Result

We now have the necessary components to prove our main result.

Proof of Theorem 1.2.3. Let $X = \operatorname{supp}(\widehat{x})$ and T be the basis guaranteed by Theorem 2.4.6. Consider any basis $S \subseteq X$. We aim to show that if:

$$\det(V_S V_S^\top) \leq d^{-c \cdot d} \cdot \det(V_T V_T^\top) = d^{-c' \cdot d} \cdot \mathrm{OPT},$$

for some large enough constants c, c' then the algorithm finds an f-violating cycle in $\widetilde{G}(S)$ and makes progress.

Since $|X| \leq r + 2\left(\binom{d+1}{2} + d\right) \leq r + 3d^2$, if:

$$\det(V_T V_T^\top) \geq d \cdot d! \cdot f(2d) \cdot (12d^2)^d \cdot \det(V_S V_S^\top),$$

then Lemma 2.4.7 implies the existence of an f-violating cycle C of length at most $2d$.

Let C be such a minimal cycle and $\widehat{S} = S \triangle C$. Then by Theorem 2.4.5:

$$\det(V_{\widehat{S}} V_{\widehat{S}}^\top) \geq 2 \cdot \det(V_S V_S^\top).$$

This ensures exponential progress in each iteration, leading to the desired approximation. □

2.4.5 Existence of Short Cycle

Lemma 2.4.8. *Let* $S \subseteq [n]$ *such that* $\det(V_S V_S^\top) > 0$*, and let* $Y \subseteq S$*,* $X \subseteq [n] \backslash S$ *with* $|X| = |Y| = \ell$*. Let* $T = S - Y + X$*. Then*

$$\frac{\det(V_T V_T^\top)}{\det(V_S V_S^\top)} = \det \begin{bmatrix} I_\ell + V_X^\top (V_S V_S^\top)^{-1} V_X & V_X^\top (V_S V_S^\top)^{-1} V_Y \\ -V_Y^\top (V_S V_S^\top)^{-1} V_X & I_\ell - V_Y^\top (V_S V_S^\top)^{-1} V_Y \end{bmatrix}. \tag{2.16}$$

Proof. Let $M = V_S V_S^\top$ and we know $T = S - Y + X$ with $|X| = |Y| = \ell$. Then:

$$V_T V_T^\top = M - V_Y V_Y^\top + V_X V_X^\top.$$

Using the matrix determinant lemma for a symmetric low-rank update:

$$\frac{\det(V_T V_T^\top)}{\det(M)} = \det\left(I + \Sigma V^\top M^{-1} V\right),$$

where $V = [V_X \;\; V_Y]$ and $\Sigma = \begin{bmatrix} I_\ell & 0 \\ 0 & -I_\ell \end{bmatrix}$.

This gives:

$$\det \begin{bmatrix} I_\ell + V_X^\top M^{-1} V_X & V_X^\top M^{-1} V_Y \\ -V_Y^\top M^{-1} V_X & I_\ell - V_Y^\top M^{-1} V_Y \end{bmatrix},$$

as claimed. □

Corollary 2.4.9. *Let $S \subseteq [n]$ such that $\det(V_S V_S^\top) > 0$, and let $Y \subseteq S$, $X \subseteq [n] \backslash S$ with $|X| = |Y| = \ell$. Let $T = S - Y + X$. Then*

$$\det(V_T V_T^\top) \geq \det(V_S V_S^\top) \cdot \det(V_X^\top (V_S V_S^\top)^{-1} V_Y)^2. \tag{2.17}$$

Proof. The proof is identical to that of Corollary 2.7.3. □

Proof of Lemma 2.4.4. Define $Z = (V_S V_S^\top)^{-1}$. Let $X := T \backslash S$ and $Y := S \backslash T$. Using Lemma 2.4.8, we have

$$\frac{\det(V_T V_T^\top)}{\det(V_S V_S^\top)} = \det\left(\begin{bmatrix} I_\ell + V_X^\top Z V_X & V_X^\top Z V_Y \\ -V_X^\top Z V_Y & I_\ell - V_Y^\top Z V_Y \end{bmatrix}\right) := \det(A).$$

Expanding the determinant of A gives

$$|\det(A)| = |\sum_{\sigma \in \mathcal{S}_{2\ell}} \prod_{i=1}^{2\ell} a_{i,\sigma(i)}| \leq \sum_{\sigma \in \mathcal{S}_{2\ell}} \prod_{i=1}^{2\ell} |a_{i,\sigma(i)}|.$$

Since $|\mathcal{S}_{2\ell}| = (2\ell)!$, there exists a permutation $\tau \in \mathcal{S}_{2\ell}$ such that

$$\prod_{i=1}^{2\ell} |a_{i,\tau(i)}| \geq \frac{1}{(2\ell)!} |\det(A)| = \frac{1}{(2\ell)!} \cdot \frac{\det(V_T V_T^\top)}{\det(V_S V_S^\top)} \geq f(2\ell) \tag{2.18}$$

where the last inequality follows from the assumption of the Lemma.

We now relate $|a_{i,\tau(i)}|$ to the weight of an arc in $\widetilde{G}(S)$ for every i and then use those arcs to construct a set of cycles, one of which will give an f-violating cycle.

To bound $|a_{i,\tau(i)}|$, consider the partition of the set of indices, $[2\ell]$, into four sets according to τ as follows. Let $I_1 = \{i \in [\ell] : \tau(i) \leq \ell\}$, $I_2 = \{i \in [\ell] : \tau(i) > \ell\}$, $I_3 = \{i \in [\ell+1, 2\ell] : \tau(i) \leq \ell\}$,

and $I_4 = \{i \in [\ell+1, 2\ell] : \tau(i) > \ell\}$. Note that $|I_1| = |I_4|$ and $|I_2| = |I_3|$; specifically, the number of permutation entries in the top left block is equal to the number of permutation entries in the bottom right block. Let $X = \{u_1, u_2, \ldots, u_\ell\}$ and $Y = \{v_{\ell+1}, v_{\ell+2}, \ldots, v_{2\ell}\}$ so that $S - v_{\ell+i} + u_i \in \mathcal{I}$ for all $i \in [\ell]$.

For any $i \in I_1$, $a_{i,\tau(i)} = 1 + u_i^\top Z u_{\tau(i)}$. As $I_\ell + V_X^\top Z V_X \succeq 0$, we have

$$|a_{i,\tau(i)}| = |1 + u_i^\top Z u_{\tau(i)}| \leq \sqrt{\left(1 + u_i^\top Z u_i\right) \cdot \left(1 + u_{\tau(i)}^\top Z u_{\tau(i)}\right)}.$$

Similarly, for any $i \in I_4$ with $i \neq \tau(i)$, we have

$$|a_{i,\tau(i)}| = |v_i^\top Z v_{\tau(i)}|$$

Since $I - V_Y^\top Z V_Y \succeq 0$, we have

$$|v_i^\top Z v_{\tau(i)}| \leq \sqrt{(1 - v_i^\top Z v_i) \cdot (1 - v_{\tau(i)}^\top Z v_{\tau(i)})}.$$

This inequality is trivially true when $\tau(i) = i$ for all $i \in I_4$.

For $i \in I_2$, $|a_{i,\tau(i)}| = |u_i^\top Z v_{\tau(i)}|$. Similarly, for $i \in I_3$, $|a_{i,\tau(i)}| = |u_{\tau(i)}^\top Z v_i|$.

Putting it all together,

$$\begin{aligned}
\prod_{i=1}^{2\ell} |a_{i,\tau(i)}| &= \left(\prod_{i \in I_1} |a_{i,\tau(i)}|\right) \cdot \left(\prod_{i \in I_2} |a_{i,\tau(i)}|\right) \cdot \left(\prod_{i \in I_3} |a_{i,\tau(i)}|\right) \cdot \left(\prod_{i \in I_4} |a_{i,\tau(i)}|\right) \\
&\leq \left(\prod_{i \in I_1} \sqrt{(1 + u_i^\top Z u_i)(1 + u_{\tau(i)}^\top Z u_{\tau(i)})}\right) \cdot \left(\prod_{i \in I_4} \sqrt{(1 - v_i^\top Z v_i)(1 - v_{\tau(i)}^\top Z v_{\tau(i)})}\right) \\
&\quad \cdot \left(\prod_{i \in I_2} |u_i^\top Z v_{\tau(i)}|\right) \cdot \left(\prod_{i \in I_3} |u_{\tau(i)}^\top Z v_i|\right).
\end{aligned}$$

We will now relate the R.H.S. of the inequality to the weights of cycles in $\widetilde{G}(S)$. For this purpose, we define a new permutation δ as follows. Since $|I_1| = |I_4|$, first define a bijection $h : I_1 \to I_4$. Then the permutation δ is given by $\delta(i) = \tau(h(i))$ for every $i \in I_1$, $\delta(i) = \tau(h^{-1}(i))$ for every $i \in I_4$, and $\delta(i) = \tau(i)$ for any $i \in I_2 \cup I_3$. Then the inequality becomes

$$\begin{aligned}
\prod_{i=1}^{2\ell} |a_{i,\tau(i)}| \leq &\prod_{i \in I_1} \sqrt{(1 + u_i^\top Z u_i) \cdot (1 - v_{\delta(i)}^\top Z v_{\delta(i)})} \cdot \qquad (2.19) \\
&\cdot \prod_{i \in I_4} \sqrt{(1 + u_{\delta(i)}^\top Z u_{\delta(i)}) \cdot (1 - v_i^\top Z v_i)} \cdot \\
&\cdot \prod_{i \in I_2} |u_i^\top Z v_{\delta(i)}| \cdot \prod_{i \in I_3} |u_{\delta(i)}^\top Z v_i|, \qquad (2.20)
\end{aligned}$$

where we have swapped some terms using the new permutation to ensure that each square root has one addition and one subtraction term. We claim that $S+u_i-v_{\delta(i)}$ is a linear spanning subset of $\mathbb{R}^d$ for any $i \in I_1 \cup I_2$. To observe this, note that

$$\frac{\det(S+u_i-v_{\delta(i)})}{\det(S)} = (u_i^\top Z v_{\delta(i)})^2 + (1+u_i^\top Z u_i)\cdot(1-v_{\delta(i)}^\top Z v_{\delta(i)})$$

from the definition of the weights. Since $\prod_{i=1}^{2\ell}|a_{i,\tau(i)}| > 0$, inequality (2.20) implies, either $|u_i^\top Z v_{\delta(i)}| > 0$ or $(1+u_i^\top Z u_i)\cdot(1-v_{\delta(i)}^\top Z v_{\delta(i)}) > 0$ for every $i \in I_1 \cup I_2$. Therefore $\det(S+u_i-v_{\delta(i)}) > 0$ and $S+u_i-v_{\delta(i)}$ is a linearly spanning set of cardinality r. So, $\widetilde{G}(S)$ contains forward arcs $u_i \xrightarrow{\text{I}} v_{\delta(i)}$ and $u_i \xrightarrow{\text{II}} v_{\delta(i)}$ for all $i \in I_1 \cup I_2$. Similarly, $\widetilde{G}(S)$ contains arcs $u_{\delta(i)} \xrightarrow{\text{I}} v_i$ and $u_{\delta(i)} \xrightarrow{\text{II}} v_i$ for every $i \in I_3 \cup I_4$. So, we can rewrite Equation (2.20) in terms of arc weights from $\widetilde{G}(S)$.

$$\prod_{i=1}^{2\ell}|a_{i,\tau(i)}| \le \exp\left(-\sum_{i\in I_1} w(u_i \xrightarrow{\text{II}} v_{\delta(i)}) - \sum_{i\in I_4} w(u_{\delta(i)} \xrightarrow{\text{II}} v_i) - \sum_{i\in I_2} w(u_i \xrightarrow{\text{I}} v_{\delta(i)}) - \prod_{i\in I_3} w(u_{\delta(i)} \xrightarrow{\text{I}} v_i)\right).$$

From (2.18), we have $\prod_{i=1}^{2\ell}|a_{i,\tau(i)}| \ge f(2\ell)$ and therefore

$$f(2\ell) \le \exp\left(-\sum_{i\in I_1} w(u_i \xrightarrow{\text{II}} v_{\delta(i)}) - \sum_{i\in I_4} w(u_{\delta(i)} \xrightarrow{\text{II}} v_i) - \sum_{i\in I_2} w(u_i \xrightarrow{\text{I}} v_{\delta(i)}) - \prod_{i\in I_3} w(u_{\delta(i)} \xrightarrow{\text{I}} v_i)\right). \tag{2.21}$$

Consider a weighted bipartite graph $H(S)$ with bipartitions X and Y, and forward arcs $u_i \xrightarrow{\text{II}} v_{\delta(i)}$ for all $i \in I_1$, $u_i \xrightarrow{\text{I}} v_{\delta(i)}$ for all $i \in I_2$, $u_{\delta(i)} \xrightarrow{\text{I}} v_i$ for all $i \in I_3$, $u_{\delta(i)} \xrightarrow{\text{II}} v_i$ for all $i \in I_4$. Additionally, for each $i \in [\ell]$, we add two backward arcs from $v_{\ell+i} \to u_i$ with weight 0 to $H(S)$. Since $S - v_{\ell+i} + u_i \in \mathcal{I}$, $v_{\ell+i} \to u_i$ is also an arc in $\widetilde{G}(S)$. So the arcs in $H(S)$ are a multiset of arcs in $\widetilde{G}(S)$.

$H(S)$ contains 4ℓ arcs with every vertex incident to exactly two incoming arcs and two outgoing arcs. Therefore $H(S)$ is an Eulerian graph, and we can decompose it into arc disjoint cycles $C_1, \ldots, C_k$. So, summing over the weights of arcs in $H(S)$ gives

$$\sum_{i=1}^{k} w(C_i) = \sum_{i\in I_1} w(u_i \xrightarrow{\text{II}} v_{\delta(i)}) + \sum_{i\in I_4} w(u_{\delta(i)} \xrightarrow{\text{II}} v_i) + \sum_{i\in I_2} w(u_i \xrightarrow{\text{I}} v_{\delta(i)}) + \sum_{i\in I_3} w(u_{\delta(i)} \xrightarrow{\text{I}} v_i).$$

Taking exponential and using (2.21), we get $\prod_{i=1}^{k} \exp(-w(C_i)) \geq f(2\ell)$. Since $C_1, \ldots, C_k$ are arc disjoint and $H(S)$ contains 4ℓ arcs, $\sum_{i=1}^{k} |C_i| = 4\ell$. Now using the fact that $f(a) \cdot f(b) \leq f(a+b)$, we get

$$\prod_{i=1}^{k} \exp(-w(C_i)) \geq f(2\ell) \geq \prod_{i=1}^{k} f(|C_i|/2).$$

So there exists a cycle C in $C_1, \ldots, C_k$ such that $\exp(-w(C)) > f(|C|/2)$. Since every arc in $H(S)$ is also an arc in $\widetilde{G}(S)$, the cycle C is present in $\widetilde{G}(S)$ too. Therefore C is an f-violating cycle in $\widetilde{G}(S)$. □

2.4.6 Sparsity and Existence of a Short Cycle

In this section, we prove if the size of the ground set of $\mathcal{M}$ is $r + k$, and S is a basis with suboptimality ratio $d^{4d} \cdot k^d \cdot f(2d)$, then $\widetilde{G}(S)$ contains an f-violating cycle. Combining this with the sparsity guarantee of Theorem 2.4.6 ensures that an optimality gap of d^{cd} (for some constant c) suffices for the existence of an f-violating cycle.

We start by proving there exists a basis T such that the symmetric difference between T and S is ℓ and the ratio of the determinants of T and S is at least $(2\ell!)^{12}$, then $\widetilde{G}(S)$ contains an f-violating cycle (see Lemma 2.4.4). A crucial ingredient of this proof is to relate the inner product space induced by $(V_S V_S^\top)^{-1}$ to arc weights in $\widetilde{G}(S)$. With this fact in hand, we use the augmentation property of matroids to construct a basis that differs from S in only $2d$ elements to complete the proof of Lemma 2.4.7.

We restate Lemma 2.4.7 for the reader's convenience.

Lemma 2.4.7 (Existence of Short f-Violating Cycle)**.** *Assume* $\operatorname{rank}(\mathcal{M}) = r > d$ *and that the ground set of* $\mathcal{M}$ *contains* $r+k$ *elements. Let* S *be any basis with* $\det(V_S V_S^\top) > 0$*. If there exists a basis* S_1 *such that:*

$$\det(V_{S_1} V_{S_1}^\top) \geq \det(V_S V_S^\top) \cdot d \cdot d! \cdot (4k)^d \cdot f(2d),$$

then there exists a basis T *such that* $|T \triangle S| \leq 2d$ *and:*

$$\det(V_T V_T^\top) \geq (2d)! \cdot f(2d) \cdot \det(V_S V_S^\top).$$

Implying the existence of an f*-violating cycle using Lemma 2.4.4.*

Proof. Let $T = \{u_1, u_2, \ldots, u_r\}$ and $S = \{v_1, v_2, \ldots, v_r\}$ such that $S - v_i + u_i \in \mathcal{I}$ for all $i \in [r]$. By strong basis exchange, such an ordering always exists. Using the Cauchy-Binet formula,

$$\frac{\det(V_T V_T^\top)}{\det(V_S V_S^\top)} = \frac{\sum_{X \subset T, |X|=d} \det(V_X V_X^\top)}{\det(V_S V_S^\top)} = \sum_{X \subset T, |X|=d} \det(V_X^\top (V_S V_S^\top)^{-1} V_X)\,. \tag{2.22}$$

Define $R := T \backslash S$. Since the ground set contains $r+k$ elements, $|R| \leq k$. We partition the set of all d-subsets of T by their intersection with R. For a set $W \subseteq R$, let $S_W = \{X : X \subset T, |X| = d, X \cap R = W\}$. Then

$$\begin{aligned}\frac{\det(V_T V_T^\top)}{\det(V_S V_S^\top)} &= \sum_{X \subset T, |X|=d} \det(V_X^\top (V_S V_S^\top)^{-1} V_X) \\ &= \sum_{W \subseteq R, |W| \leq d} \sum_{X \in S_W} \det(V_X^\top (V_S V_S^\top)^{-1} V_X)\,.\end{aligned}$$

The number of subsets of R of size at most d is $\sum_{i=0}^{d} \binom{k}{i} \leq d \cdot \binom{k}{d} \leq dk^d/d!$. Therefore, there exists a $W \subseteq R$ with $|W| \leq d$ such that

$$\sum_{X \in S_W} \det(V_X^\top (V_S V_S^\top)^{-1} V_x) \geq \frac{d!}{dk^d} \cdot \frac{\det(V_T V_T^\top)}{\det(V_S V_S^\top)} \geq (2d)! \cdot f(2d)\,,$$

where the last inequality follows from the hypothesis of the lemma.

Since $\{S \cap T\} \cup W \subset T$, by the downward closure property of matroids, $\{S \cap T\} \cup W$ is independent in $\mathcal{M}$. So we can extend $\{S \cap T\} \cup W$ to a basis, T_1, of $\mathcal{M}$ in $S \cup W$ such that $\{S \cap T\} \cup W \subseteq T_1$. Again, using the Cauchy-Binet formula on $T_1 T_1^\top$ gives

$$\begin{aligned}\frac{\det(V_{T_1} V_{T_1}^\top)}{\det(V_S V_S^\top)} &= \sum_{X \subset T_1, |X|=d} \det(V_X^\top (V_S V_S^\top)^{-1} V_X) \\ &\geq \sum_{X \subset \{S \cap T\} \cup W, |X|=d} \det(V_X^\top (V_S V_S^\top)^{-1} V_X) \\ &\geq \sum_{X \in S_W, |X|=d} \det(V_X^\top (V_S V_S^\top)^{-1} V_X) \geq (2d)! \cdot f(2d).\end{aligned}$$

Since $|T_1 \backslash S| \leq d$, using Lemma 2.4.4, there exists an f-violating cycle in $\widetilde{G}(S)$. □

2.4.7 Cycle to Determinant

In this Section, we give an outline of the proof of Theorem 2.4.5. The proof is identical to the proof of 2.3.5, as the weight functions satisfy the same properties when $r < d$ and $r > d$. For the sake of completeness, we state the essential lemmas for this case, but forgo the proofs when the proofs repeat from the $r < d$ case.

Since type II are multiplicative in vertices in this case as well, any type minimal f-violating cycle can contain at most one type II edge.

Lemma 2.4.10. *Let C be a minimal f-violating cycle in $\widetilde{G}(S)$. Then C contains at most one edge of type* II.

The proof is identical to that of Lemma 2.3.11.

The following lemma shows that the coefficient function of the basis obtained by exchanging one pair of vectors can be completely characterized in terms of coefficient and projection functions of the current basis. This lemma plays a crucial role in bounding the determinant when the minimal f-violating cycle contains a type II edge.

Lemma 2.4.11. *Let $S \in \mathcal{I}$ and $v_1 \in S$ and $u_1 \notin S$ such that $\det(V_{S-v_1+u_1} V_{S-v_1+u_1}^\top) > 0$. Let $S_1 := S - v_1 + u_1$. Then for any $v_i \in S$ and $u_j \notin S$,*

$$u_j^\top (V_{S_1} V_{S_1}^\top)^{-1} v_i = u_j^\top (V_S V_S^\top)^{-1} v_i + c_1 x_j y_i + c_2 x_j z_i + c_3 w_j y_i + c_4 w_j z_i,$$

with $x_j := u_j^\top X u_1$, $w_j := u_j^\top X v_1$, $y_i = u_1^\top X v_i$, and $z_i := v_i^\top X v_1$, where $X = (V_S V_S^\top)^{-1}$. In addition, let $\delta = (u_1^\top X v_1)^2 + (1 + u_1^\top X u_1) \cdot (1 - v_1^\top X v_1)$. Then $c_1 = -(1 - v_1^\top X v_1)/\delta$, $c_2 = -c_3 = -u_1^\top X v_1/\delta$, and $c_4 = -(1 + u_1^\top X u_1)/\delta$ are constants independent of u_j and v_i.

Analogous to the $r < d$ case, we define a matrix $B \in \mathbb{R}^{\ell \times \ell}$ such that the (i,j)-th entry of B corresponds to the arc with the lower weight, i.e.,

- $b_{i,j} = \max\{|u_j^\top X v_i|, \sqrt{(1 + u_j^\top X u_j) \cdot (1 - v_i^\top X v_i)}\}$ for all $i \neq j$ with $X = (V_S V_S^\top)^{-1}$, and
- $b_{i,i} = \begin{cases} |u_i^\top X v_i| & \text{if } u_i \to v_i \text{ is type I} \\ \sqrt{(1 + u_i^\top X u_i) \cdot (1 - v_i^\top X v_i)} & \text{if } u_i \to v_i \text{ is type II} \end{cases}$.

Then Lemma 2.3.15 still holds for the minimal f-violating cycle in $\widetilde{G}(S)$.

Proof of Lemma 2.4.5. The proof that T is independent is again identical to Lemma 2.2.14 from the case that $r = d$. For bounding $\det(V_T V_T^\top)$, again consider two cases based on the number of type II edges in C: Lemma 2.4.10 establishes that C can contain at most one edge of type II.

Define $X := (V_S V_S^\top)^{-1}$. When C contains no type II edges, by Corollary 2.4.9, $\det(V_T V_T^\top) \geq \det(V_S V_S^\top) \cdot \det(A_S)^2$, where the (i,j)-th entry of A_S is $u_j^\top X v_i$. Identical to the proof of Lemma 2.3.16, we have $b_{i,i} = u_i^\top (V_S V_S^\top)^{-1} v_i$ for all $i \in [\ell]$ and $|u_j^\top (V_S V_S^\top) v_i| \leq w_{i,j}$. As a result,

$$\begin{aligned} |\det(A_S)| &\geq 2 \cdot \prod_{s=1}^{\ell} |a_{s,s}| - \operatorname{per}(|A_S|) \geq 2 \cdot \prod_{s=1}^{\ell} b_{s,s} - \operatorname{per}(B) \\ &\geq 0.95 \prod_{s=1}^{\ell} b_{i,i} > 0.95 f(\ell) > 2. \end{aligned}$$

When C contains exactly one edge of type II, the proof follows exactly as that of Lemma 2.3.17. Consider an intermediate set $\widehat{S} = S - v_\ell + u_\ell$. Then

$$\det(V_{\widehat{S}} V_{\widehat{S}}^\top) \geq \det(V_S V_S^\top) \cdot (1 + u_\ell^\top X u_\ell)(1 - v_\ell^\top X v_\ell) = \det(V_S V_S^\top) \cdot w_{\ell,\ell}^2 > 0$$

where $w_{\ell,\ell} = \exp(-w(u_\ell \xrightarrow{\text{II}} v_\ell)) > 0$ as $u_\ell \xrightarrow{\text{II}} v_\ell$ appears in an f-violating cycle. So,

$$\frac{\det(V_T V_T^\top)}{\det(V_S V_S^\top)} = \frac{\det(V_T V_T^\top)}{\det(V_{\widehat{S}} V_{\widehat{S}}^\top)} \cdot \frac{\det(V_{\widehat{S}} V_{\widehat{S}}^\top)}{\det(V_S V_S^\top)} = \frac{\det(V_T V_T^\top)}{\det(V_{\widehat{S}} V_{\widehat{S}}^\top)} \cdot w_{\ell,\ell}^2.$$

Again, the main idea is to show that $\det(V_T V_T^\top) \geq 0.09 \prod_{i=1}^{\ell-1} b_{i,i}^2 \cdot \det(V_{\widehat{S}} V_{\widehat{S}}^\top)$ and plugging it into the above equation completes the proof.

Let $A_{\widehat{S}}$ be the $(\ell-1) \times (\ell-1)$ matrix with i,j-th entry $|u_j^\top (V_{S_1} V_{S_1}^\top)^{-1} v_i|$. Then by Corollary 2.4.9, $\det(V_T V_T^\top) \geq \det(V_{S_1} V_{S_1}^\top) \cdot \det(A_{\widehat{S}})^2$.

$$[A_{\widehat{S}}]_{i,j} = u_j^\top (V_{\widehat{S}} V_{\widehat{S}}^\top)^{-1} v_i = u_j^\top X v_i + c_1 x_j y_i + c_2 x_j z_i + c_3 w_j y_i + c_4 w_j z_i,$$

with $x_j := u_j^\top X u_\ell$, $w_j := u_j^\top X v_\ell$, $y_i = u_\ell^\top X v_i$, and $z_i := v_i^\top X v_\ell$. In addition, let $\delta = (u_\ell^\top X v_\ell)^2 + (1 + u_\ell^\top X u_\ell) \cdot (1 - v_\ell^\top X v_\ell)$. Then $c_1 = -(1 - v_\ell^\top X v_\ell)/\delta$, $c_2 = -c_3 = -u_\ell^\top X v_\ell / \delta$, and $c_4 = -(1 + u_\ell^\top X u_\ell)/\delta$ are constants independent of u_i and v_j.

We will proceed identically as the proof of Lemma 2.3.17. We only need to establish the following inequalities and then the structure of the proof is identical as that of Lemma 2.3.17.

- $|Q_{i,j}| \leq 4 b_{\ell,j} b_{i,\ell} / b_{\ell,\ell}$ for all $i, j \in [\ell - 1]$
- $|c_2 c_3 z_i x_j y_{i'} w_{j'}| \leq \frac{b_{i,j} b_{i',\ell} b_{\ell,j'}}{b_{\ell,\ell}}$ for $i \neq i'$ and $j \neq i'$
- $|c_1 c_4 y_i x_j z_{i'} w_{j'}| \leq \frac{b_{i,\ell} b_{i',j} b_{\ell,j'}}{b_{\ell,\ell}}$ for $i \neq i'$ and $j \neq j'$

For the first inequality

$$\begin{aligned}
|c_1 x_j y_i| &= \frac{(1 - v_\ell^\top X v_\ell) \cdot |u_j^\top X u_\ell| \cdot |u_\ell^\top X v_i|}{(u_\ell^\top X v_\ell)^2 + (1 - v_\ell^\top X v_\ell)(1 + u_\ell^\top X u_\ell)} \\
&\leq \frac{(1 - v_\ell^\top X v_\ell) \sqrt{u_j^\top X u_j} \sqrt{u_\ell^\top X u_\ell} \cdot b_{i,\ell}}{(u_\ell^\top X v_\ell)^2 + (1 - v_\ell^\top X v_\ell)(1 + u_\ell^\top X u_\ell)} \\
&\leq \frac{\sqrt{(1 - v_\ell^\top X v_\ell)(1 + u_j^\top X u_j)}}{\sqrt{(1 - v_\ell^\top X v_\ell)(1 + u_\ell^\top X u_\ell)}} \cdot b_{i,\ell} = \frac{b_{i,\ell} b_{\ell,j}}{b_{\ell,\ell}}.
\end{aligned}$$

Similarly, $|c_2 x_j z_i|, |c_3 w_j y_i|, |c_4 w_j z_i| \leq b_{i,\ell} b_{\ell,j} / b_{\ell,\ell}$. For the second inequality,

$$\begin{aligned}
|c_2 c_3 x_j z_i w_{j'} y_{i'}| &= \frac{(u_\ell^\top X v_\ell)^2 \cdot |u_j^\top X u_\ell| \cdot |v_i^\top X v_\ell| \cdot |u_{j'}^\top X v_\ell| \cdot |u_\ell^\top X v_{i'}|}{((u_\ell^\top X v_\ell)^2 + (1 - v_\ell^\top X v_\ell)(1 + u_\ell^\top X u_\ell))^2} \\
&\leq \frac{(|u_j^\top X u_\ell| \cdot |v_i^\top X v_\ell| \cdot |u_{j'}^\top X v_\ell| \cdot |u_\ell^\top X v_{i'}|}{(1 - v_\ell^\top X v_\ell)(1 + u_\ell^\top X u_\ell)} \\
&= \frac{|u_j^\top X u_\ell| \cdot |v_i^\top X v_\ell| \cdot b_{\ell,j'} \cdot b_{i',\ell}}{b_{\ell,\ell}^2}
\end{aligned}$$

Note that

$$\begin{aligned}
|u_j^\top X u_\ell| \cdot |v_i^\top X v_\ell| &\leq \sqrt{(1 + u_j^\top X u_j)} \cdot \sqrt{(1 + u_\ell^\top X u_\ell)} \cdot |v_i^\top X v_\ell| \\
&\leq \sqrt{(1 + u_j^\top X u_j)} \cdot \sqrt{(1 + u_\ell^\top X u_\ell)} \cdot \sqrt{(1 - v_i^\top X v_i)} \cdot \sqrt{(1 - v_\ell^\top X v_\ell)} \\
&= \sqrt{(1 + u_j^\top X u_j)(1 - v_i^\top X v_i)} \cdot \sqrt{(1 + u_\ell^\top X u_\ell)(1 - v_\ell^\top X v_\ell)} \leq b_{i,j} b_{\ell,\ell}.
\end{aligned}$$

Therefore, $|c_2 c_3 z_i x_j y_{i'} w_{j'}| \leq \frac{b_{i,j} b_{i',\ell} b_{\ell,j'}}{b_{\ell,\ell}}$. The proof of the third inequality follows identically. $\square$

2.5 Permanental Inequalities

Lemma 2.5.1 (Permanent uncrossing)**.** *Let $B \in \mathbb{R}^{n\times n}_{\geq 0}$ and $x_1, x_2, y_1, y_2 \in \mathbb{R}^n_{\geq 0}$ and $w_{11}, w_{12}, w_{21}, w_{22} \in \mathbb{R}_{\geq 0}$. Then the following inequality holds:*

$$\operatorname{per}\begin{bmatrix} B & y_1 & y_2 \\ x_1^T & w_{11} & w_{12} \\ x_2^T & w_{21} & w_{22} \end{bmatrix} \cdot \operatorname{per}(B) \leq \operatorname{per}\begin{bmatrix} B & y_1 \\ x_1^T & w_{11} \end{bmatrix} \cdot \operatorname{per}\begin{bmatrix} B & y_2 \\ x_2^T & w_{22} \end{bmatrix} \tag{2.23}$$

$$+ \operatorname{per}\begin{bmatrix} B & y_2 \\ x_1^T & w_{12} \end{bmatrix} \cdot \operatorname{per}\begin{bmatrix} B & y_1 \\ x_2^T & w_{21} \end{bmatrix}. \tag{2.24}$$

Proof. We proceed by induction on n.

For the base case $n = 0$, both sides equal $w_{11}w_{22} + w_{12}w_{21}$.

The case $n = 1$ can be verified directly.

Now assume the lemma holds for $(n-1) \times (n-1)$ matrices. We first prove the result when all $w_{ij} = 0$.

Expanding the permanent on the left-hand side:

$$\operatorname{per}\begin{bmatrix} B & y_1 & y_2 \\ x_1^T & 0 & 0 \\ x_2^T & 0 & 0 \end{bmatrix} = \sum\nolimits_{i_1\neq i_2,\, j_1\neq j_2} x_{1,j_1}x_{2,j_2}y_{1,i_1}y_{2,i_2} \cdot \operatorname{per}\big(B(-\{i_1,i_2\},-\{j_1,j_2\})\big). \tag{2.25}$$

Dropping the distinctness constraints increases the sum:

$$\leq \sum\nolimits_{i_1,i_2,j_1,j_2} x_{1,j_1}x_{2,j_2}y_{1,i_1}y_{2,i_2} \cdot \operatorname{per}\big(B(-\{i_1,i_2\},-\{j_1,j_2\})\big). \tag{2.26}$$

Multiplying both sides by $\operatorname{per}(B)$:

$$\operatorname{per}\begin{bmatrix} B & y_1 & y_2 \\ x_1^T & 0 & 0 \\ x_2^T & 0 & 0 \end{bmatrix} \cdot \operatorname{per}(B) \leq \sum\nolimits_{i_1,i_2,j_1,j_2} x_{1,j_1}x_{2,j_2}y_{1,i_1}y_{2,i_2} \cdot \operatorname{per}\big(B(-\{i_1,i_2\},-\{j_1,j_2\})\big) \cdot \operatorname{per}(B). \tag{2.27}$$

By the inductive hypothesis, for each term:

$$\operatorname{per}\big(B(-\{i_1,i_2\},-\{j_1,j_2\})\big) \cdot \operatorname{per}(B) \leq \operatorname{per}\big(B(-\{i_1\},-\{j_1\})\big) \cdot \operatorname{per}\big(B(-\{i_2\},-\{j_2\})\big) \tag{2.28}$$

$$+ \operatorname{per}\big(B(-\{i_1\},-\{j_2\})\big) \cdot \operatorname{per}\big(B(-\{i_2\},-\{j_1\})\big). \tag{2.29}$$

Substituting this bound:

$$\operatorname{per}\begin{bmatrix} B & y_1 & y_2 \\ x_1^T & 0 & 0 \\ x_2^T & 0 & 0 \end{bmatrix} \cdot \operatorname{per}(B) \leq \sum_{i_1,i_2,j_1,j_2} x_{1,j_1} x_{2,j_2} y_{1,i_1} y_{2,i_2} \Big[\operatorname{per}(B(-\{i_1\},-\{j_1\})) \cdot \operatorname{per}(B(-\{i_2\},-\{j_2\})) \tag{2.30}$$

$$+ \operatorname{per}(B(-\{i_1\},-\{j_2\})) \cdot \operatorname{per}(B(-\{i_2\},-\{j_1\})) \Big]. \tag{2.31}$$

Now we compute the right-hand side terms:

$$\operatorname{per}\begin{bmatrix} B & y_1 \\ x_1^T & 0 \end{bmatrix} = \sum\nolimits_{i_1,j_1} x_{1,j_1} y_{1,i_1} \cdot \operatorname{per}(B(-\{i_1\},-\{j_1\})), \tag{2.32}$$

$$\operatorname{per}\begin{bmatrix} B & y_2 \\ x_2^T & 0 \end{bmatrix} = \sum\nolimits_{i_2,j_2} x_{2,j_2} y_{2,i_2} \cdot \operatorname{per}(B(-\{i_2\},-\{j_2\})). \tag{2.33}$$

Thus, the product of these two terms equals the first sum in the RHS expansion above. Similarly, the cross terms:

$$\operatorname{per}\begin{bmatrix} B & y_2 \\ x_1^T & 0 \end{bmatrix} = \sum\nolimits_{i_2,j_1} x_{1,j_1} y_{2,i_2} \cdot \operatorname{per}(B(-\{i_2\},-\{j_1\})), \tag{2.34}$$

$$\operatorname{per}\begin{bmatrix} B & y_1 \\ x_2^T & 0 \end{bmatrix} = \sum\nolimits_{i_1,j_2} x_{2,j_2} y_{1,i_1} \cdot \operatorname{per}(B(-\{i_1\},-\{j_2\})). \tag{2.35}$$

Multiplying and adding these reproduces the second sum in the expansion of the upper bound. Therefore,

$$\operatorname{per}\begin{bmatrix} B & y_1 & y_2 \\ x_1^T & 0 & 0 \\ x_2^T & 0 & 0 \end{bmatrix} \cdot \operatorname{per}(B) \leq \operatorname{per}\begin{bmatrix} B & y_1 \\ x_1^T & 0 \end{bmatrix} \cdot \operatorname{per}\begin{bmatrix} B & y_2 \\ x_2^T & 0 \end{bmatrix} + \operatorname{per}\begin{bmatrix} B & y_2 \\ x_1^T & 0 \end{bmatrix} \cdot \operatorname{per}\begin{bmatrix} B & y_1 \\ x_2^T & 0 \end{bmatrix}. \tag{2.36}$$

Now consider the general case with arbitrary $w_{ij} \geq 0$. Define:

$$f(w) := \operatorname{per}\begin{bmatrix} B & y_1 \\ x_1^T & w_{11} \end{bmatrix} \cdot \operatorname{per}\begin{bmatrix} B & y_2 \\ x_2^T & w_{22} \end{bmatrix} + \operatorname{per}\begin{bmatrix} B & y_2 \\ x_1^T & w_{12} \end{bmatrix} \cdot \operatorname{per}\begin{bmatrix} B & y_1 \\ x_2^T & w_{21} \end{bmatrix} \tag{2.37}$$

$$- \operatorname{per}\begin{bmatrix} B & y_1 & y_2 \\ x_1^T & w_{11} & w_{12} \\ x_2^T & w_{21} & w_{22} \end{bmatrix} \cdot \operatorname{per}(B). \tag{2.38}$$

By multilinearity of the permanent in each row and column, we observe that $\nabla_w f = 0$. That is, each derivative $\frac{\partial f}{\partial w_{ij}}$ is zero because the positive and negative contributions cancel exactly. For example, consider the partial derivative $\frac{\partial f}{\partial w_{1,1}}$. The positive contribution comes from the term

$$\operatorname{per}\begin{bmatrix} B & y_1 \\ x_1^T & w_{1,1} \end{bmatrix} \cdot \operatorname{per}\begin{bmatrix} B & y_2 \\ x_2^T & w_{2,2} \end{bmatrix}$$

in the RHS, which contributes

$$\operatorname{per}(B) \cdot \operatorname{per} \begin{bmatrix} B & y_2 \\ x_2^T & w_{2,2} \end{bmatrix}$$

by linearity in $w_{1,1}$. An equal and opposite term appears on the LHS from the expansion of

$$\operatorname{per} \begin{bmatrix} B & y_1 & y_2 \\ x_1^T & w_{1,1} & w_{1,2} \\ x_2^T & w_{2,1} & w_{2,2} \end{bmatrix} \cdot \operatorname{per}(B),$$

which contributes

$$-\operatorname{per}(B) \cdot \operatorname{per} \begin{bmatrix} B & y_2 \\ x_2^T & w_{2,2} \end{bmatrix}.$$

These two terms cancel exactly, confirming that $\frac{\partial f}{\partial w_{1,1}} = 0$. Thus, f is constant with respect to w.

In particular, $f(w) = f(0)$, and we already showed $f(0) \geq 0$.

Hence, $f(w) \geq 0$ for all $w \geq 0$, completing the proof. □

Theorem 2.5.2. *Let A be an $\ell \times \ell$ matrix with non-negative entries satisfying*

- $a_{i,j} \leq f(j-i)$ *for* $i < j$
- $a_{i,j} \leq \frac{f(\ell-i+j)}{f(\ell)}$ *for* $j < i$
- $a_{i,i} \geq 1$.

Then $\operatorname{per}(A) \leq \left(1 + \frac{0.1}{\ell}\right) \prod_{s=1}^{\ell} a_{s,s}$.

Proof. If $\ell = 2$, we have

$$\operatorname{per}(A) \leq a_{1,1}a_{2,2} + \frac{f(1)f(1)}{f(2)} < 1.05 \cdot a_{1,1}a_{2,2}.$$

Here we used $a_{1,1}a_{2,2} \geq 1$.

So assume $\ell \geq 3$. Since each entry of A is non-negative, $\operatorname{per}(A)$ is a non-decreasing function in every entry of A. So we can assume that $a_{i,j} = f(j-i)$ for $i < j$ and $a_{i,j} = f(\ell - i + j)/f(\ell)$ for $j < i$ as this only increases the permanent.

We will inductively show that for any $s \in [\ell]$,

$$\operatorname{per}(A_{i,i}) \leq \left(1 + \frac{1.25}{\ell^4}\right)^{i-1} \prod_{s=1}^{i} a_{s,s}. \tag{2.39}$$

Where $A_{i,i}$ is the submatrix of A that contains the first i rows and columns from A. Then for all $s = \ell$, we have

$$\operatorname{per}(A) \leq \left(1 + \frac{1.25}{\ell^4}\right)^{\ell-1} \prod_{i=1}^{\ell} a_{i,i} \leq \prod_{i=1}^{\ell} a_{i,i} \cdot \left(1 + \frac{0.1}{\ell}\right).$$

Here, the last inequality follows from $\frac{x}{x+1} \le \log(1+x) \le x$ for $x > 0$.

Let $B_{i,j}$ denote the submatrix of A with row set $\{1, \ldots, \min\{i,j\} - 1\} \cup \{i\}$ and column set $\{1, \ldots, \min\{i,j\} - 1\} \cup \{j\}$. To establish inequality (2.39), we will use Lemma 2.5.3 and prove that for any $i \neq j$,

$$\frac{\operatorname{per}(B_{i,j})}{\operatorname{per}(A_{\min\{i,j\}-1,\min\{i,j\}-1})} \le \begin{cases} \binom{j}{i} f(j-i) & \text{if } i < j \\ 1.25 \frac{f(\ell-i+j)}{f(\ell)} & \text{if } j < i. \end{cases} \tag{2.40}$$

We will prove this Equation by induction on $\min\{i,j\}$. For the base case, $B_{i,1} = a_{i,1}$ for any $i \in [\ell]$ and $B_{1,j} = a_{1,j}$ for all $j \in [\ell]$, so the inequality follows by our assumption on $a_{i,j}$.

For $i \le j$, by Lemma 2.5.3,

$$\begin{aligned} \frac{\operatorname{per}(B_{i,j})}{\operatorname{per}(A_{i-1,i-1})} &\le a_{i,j} + \sum_{s=1}^{i-1} \frac{\operatorname{per}(B_{i,s}) \operatorname{per}(B_{s,j})}{\operatorname{per}(A_{s-1,s-1}) \operatorname{per}(A_{s,s})} \\ &= a_{i,j} + \sum_{s=1}^{i-1} \frac{\frac{\operatorname{per}(B_{i,s})}{\operatorname{per}(A_{s-1,s-1})} \frac{\operatorname{per}(B_{s,j})}{\operatorname{per}(A_{s-1,s-1})}}{\frac{\operatorname{per}(A_{s,s})}{\operatorname{per}(A_{s-1,s-1})}} \\ &\le a_{i,j} + \sum_{s=1}^{i-1} \frac{\operatorname{per}(B_{i,s})}{\operatorname{per}(A_{s-1,s-1})} \frac{\operatorname{per}(B_{s,j})}{\operatorname{per}(A_{s-1,s-1})}. \end{aligned}$$

Here we used $\operatorname{per}(A_{s,s}) \ge a_{s,s} \operatorname{per}(A_{s-1,s-1}) \ge \operatorname{per}(A_{s-1,s-1})$ as $a_{s,s} \ge 1$ and A is a non-negative matrix.

Using the inductive hypothesis, we have

$$\begin{aligned} & a_{i,j} + \sum_{s=1}^{i-1} \frac{\operatorname{per}(B_{i,s})}{\operatorname{per}(A_{s-1,s-1})} \frac{\operatorname{per}(B_{s,j})}{\operatorname{per}(A_{s-1,s-1})} \\ &\le f(j-i) + 1.25 \sum_{s=1}^{i-1} \binom{j}{s} f(j-s) \cdot \frac{f(\ell-i+s)}{f(\ell)}. \\ &= f(j-i) + 1.25 \cdot f(j-i) \sum_{s=1}^{i-1} \binom{j}{s} \frac{f(j-s) \cdot f(\ell-i+s)}{f(j-i) \cdot f(\ell)}. \end{aligned}$$

For $i \le j$, define $\gamma(i,j) := \sum_{s=1}^{i-1} \binom{j}{s} \frac{f(j-s) \cdot f(\ell-i+s)}{f(j-i)}$. Then it suffices to show that for $i \le j$,

$$1 + 1.25\gamma(i,j) \le b_{i,j} = \binom{j}{i}. \tag{2.41}$$

Note that

$$\frac{f(j-s) \cdot f(\ell-i+s)}{f(j-i) \cdot f(\ell)} = \left(\frac{(j-s)! \cdot (\ell-i+s)!}{(j-i)! \cdot \ell!} \right)^6 = \left(\frac{\binom{\ell-i+j}{\ell}}{\binom{\ell-i+j}{j-s}} \right)^6.$$

For any $1 \le s \le i-1$, $\frac{\binom{\ell-i+j}{\ell}}{\binom{\ell-i+j}{j-s}} \le \frac{(j-i+1)}{\ell}$. Therefore,

$$\frac{f(j-s)\cdot f(\ell-i+s)}{f(j-i)\cdot f(\ell)} \le \frac{\binom{\ell-i+j}{\ell}}{\binom{\ell-i+j}{j-s}} \cdot \frac{(j-i+1)^5}{\ell^5}.$$

Summing over all $s \in [i-1]$ gives

$$\gamma(i,j) = \sum_{s=1}^{i-1} \binom{j}{s} \frac{f(j-s)\cdot f(\ell-i+s)}{f(j-i)} \le \frac{(j-i+1)^5}{\ell^5} \cdot \sum_{s=1}^{i-1} \binom{j}{s} \cdot \frac{\binom{\ell-i+j}{\ell}}{\binom{\ell-i+j}{j-s}} \tag{2.42}$$

For any $s \in [i-1]$,

$$\binom{j}{s} \cdot \frac{\binom{\ell-i+j}{\ell}}{\binom{\ell-i+j}{j-s}} = \frac{j!\binom{\ell-i+j}{\ell}(\ell-i)!}{(\ell-i+j)!} \cdot \binom{\ell-i+s}{\ell-i} = \frac{j!(\ell-i)!}{\ell!(j-i)!}\binom{\ell-i+s}{\ell-i}.$$

Substituting this in Equation (2.42) gives

$$\gamma(i,j) \le \frac{(j-i+1)^5}{\ell^5} \cdot \frac{j!(\ell-i)!}{\ell!(j-i)!} \cdot \sum_{s=1}^{i-1} \binom{\ell-i+s}{\ell-i}. \tag{2.43}$$

For any positive integers a, b, x with $x \le a \le b$,

$$\binom{a}{x} + \binom{a+1}{x} + \binom{a+2}{x} + \ldots + \binom{b}{x} = \binom{b+1}{x+1} - \binom{a}{x+1}. \tag{2.44}$$

Using (2.44) with $a = \ell-i+1$, $b = \ell-1$, and $x = \ell-i$ gives $\sum_{s=1}^{i-1}\binom{\ell-i+s}{\ell-i} \le \binom{\ell}{\ell-i+1}$ and substituting this in Equation (2.42),

$$\gamma(i,j) \le \frac{(j-i+1)^5}{\ell^5} \cdot \frac{j!(\ell-i)!}{(j-i)!\ell!} \cdot \binom{\ell}{\ell-i+1} = \binom{j}{i} \cdot \frac{(j-i+1)^5 \cdot i}{\ell^5(\ell-i+1)}. \tag{2.45}$$

For $i = j$, we have $\gamma(i,i) \le \frac{i}{\ell^5(\ell-i+1)} \le \frac{1}{\ell^4}$, and therefore,

$$\frac{\mathrm{per}(A_{i,i})}{\mathrm{per}(A_{i-1,i-1})} \le a_{i,i} + 1.25\gamma(i,i) \le a_{i,i}\left(1 + \frac{1.25}{\ell^4}\right).$$

Now we will restrict ourselves to the case when $i < j$.

$$\gamma(i,j) \le \binom{j}{i} \cdot \frac{(j-i+1)^5 i}{\ell^5(\ell-i+1)} \le \binom{j}{i} \cdot \frac{(\ell-i+1)^4 i}{\ell^5} \le \binom{j}{i} \cdot \frac{(\ell-i+1)i}{\ell^2}. \tag{2.46}$$

where the last inequality follows from $\ell - i + 1 \le \ell$.

Since $(\ell-i+1)i$ is maximized at $i = (\ell+1)/2$, we have $\frac{(\ell-i+1)i}{\ell^2} \le \frac{(\ell+1)^2}{4\ell^2} \le 0.45$ for any $\ell \ge 3$.

Plugging this in (2.46) gives

$$1 + 1.25 \cdot \gamma(i,j) \leq 1 + 1.25 \cdot 0.45 \cdot \binom{j}{i} \leq 1 + 0.6 \cdot \binom{j}{i}.$$

For $j = 2$, i can only be 1 and this corresponds to an entry in the first column for which the bounds are trivially true. So, we only need to consider $j \geq 3$. Since $1 \leq i < j$, we have $\binom{j}{i} \geq j$. Furthermore, since $\ell \geq 4$ and $j \geq 3$, we have $1 \leq 0.4 \cdot j < 0.4\binom{j}{i}$. This gives

$$1 + 0.6 \cdot \binom{j}{i} \leq \binom{j}{i}.$$

Proving the sufficient condition in Equation2.41 concluding the proof of Equation (2.40) when $i \leq j$.

For $i > j$, we again have

$$\begin{aligned}\frac{\operatorname{per}(B_{i,j})}{\operatorname{per}(A_{j-1,j-1})} &\leq a_{i,j} + \sum_{s=1}^{j-1} \frac{\operatorname{per}(B_{i,s})\operatorname{per}(B_{s,j})}{\operatorname{per}(A_{s-1,s-1})\operatorname{per}(A_{s,s})} \\ &\leq a_{i,j} + \sum_{s=1}^{j-1} \frac{\operatorname{per}(B_{i,s})}{\operatorname{per}(A_{s-1,s-1})} \frac{\operatorname{per}(B_{s,j})}{\operatorname{per}(A_{s-1,s-1})}.\end{aligned}$$

Using the inductive hypothesis gives

$$\begin{aligned}&\sum_{s=1}^{j-1} \frac{\operatorname{per}(B_{i,s})}{\operatorname{per}(A_{s-1,s-1})} \frac{\operatorname{per}(B_{s,j})}{\operatorname{per}(A_{s-1,s-1})} \\ &\quad\leq 1.25 \sum_{s=1}^{j-1} \frac{f(\ell-i+s)}{f(\ell)} \cdot \binom{j}{s} \cdot f(j-s) \\ &\quad= \frac{1.25 f(\ell-i+j)}{f(\ell)} \cdot \sum_{s=1}^{j-1} \frac{f(\ell-i+s)}{f(\ell-i+j)} \cdot \binom{j}{s} \cdot f(j-s).\end{aligned}$$

Define $\gamma(i,j) := \sum_{s=1}^{j-1} \frac{f(\ell-i+s)}{f(\ell-i+j)} \cdot \binom{j}{s} \cdot f(j-s)$. So, for all $j < i$, to prove Equation (2.40), it suffices to show that $1 + 1.25\gamma(i,j) \leq 1.25$.

Note that for any $1 \leq s \leq j-1$,

$$\frac{f(\ell-i+s) \cdot f(j-s)}{f(\ell-i+j)} = 2\left(\frac{(\ell-i+s)! \cdot (j-s)!}{(\ell-i+j)!}\right)^6 = 2 \cdot \left(\frac{1}{\binom{\ell-i+j}{j-s}}\right)^6.$$

and as a result $\frac{1}{\binom{\ell-i+j}{j-s}} \leq \frac{1}{\ell-i+j}$. Therefore,

$$\frac{f(\ell-i+s) \cdot f(j-s)}{f(\ell-i+j)} \leq \frac{1}{\binom{\ell-i+j}{j-s}} \cdot \frac{2}{(\ell-i+j)^5}.$$

Substituting this bound in $\gamma(i,j)$ gives

$$\begin{aligned}\gamma(i,j) &\leq \frac{2}{(\ell-i+j)^5}\cdot\sum_{s=1}^{j-1}\binom{j}{s}\cdot\frac{1}{\binom{\ell-i+j}{j-s}}\\ &= \frac{2\cdot j!(\ell-i)!}{(\ell-i+j)^5\cdot(\ell-i+j)!}\cdot\sum_{s=1}^{j-1}\binom{\ell-i+s}{\ell-i}.\end{aligned}$$

Using (2.44) again, we get $\sum_{s=1}^{j-1}\binom{\ell-i+s}{\ell-i}\leq\binom{\ell-i+j}{\ell-i+1}$ and this gives

$$\begin{aligned}\gamma(i,j) &\leq \frac{2\cdot j!(\ell-i)!}{(\ell-i+j)^5\cdot(\ell-i+j)!}\cdot\binom{\ell-i+j}{\ell-i+1}\\ &= \frac{2\cdot j}{(\ell-i+j)^5\cdot(\ell-i+1)}\leq\frac{2j}{j^5}=\frac{2}{j^4}\leq 0.125.\end{aligned}\tag{2.47}$$

where the last inequality follows from $j\geq 2$. Therefore, $1+1.25\gamma(i,j)\leq 1.25$ for $i>j$. □

Lemma 2.5.3. *Let A be an $\ell\times\ell$ non-negative matrix with non-zero diagonal entries. Let $A_{s,s}$ denote the principal submatrix of A formed by the first s rows and columns with $A_{0,0}$ being the empty matrix with permanent 1. Let $B_{i,j}$ denote the submatrix of A with row set $\{1,\ldots,\min\{i,j\}-1\}\cup\{i\}$ and column set $\{1,\ldots,\min\{i,j\}-1\}\cup\{j\}$. Then*

$$\frac{\operatorname{per}(A)}{\operatorname{per}(A_{\ell-1,\ell-1})}\leq a_{\ell,\ell}+\sum_{s=1}^{\ell-1}\frac{\operatorname{per}(B_{\ell,s})\cdot\operatorname{per}(B_{s,\ell})}{\operatorname{per}(A_{s,s})\cdot\operatorname{per}(A_{s-1,s-1})}.$$

Proof. We will prove this by induction on ℓ. For $\ell=2$,

$$\frac{\operatorname{per}(A)}{\operatorname{per}(A_{1,1})}=\frac{a_{1,1}a_{2,2}+a_{1,2}a_{2,1}}{a_{1,1}}=a_{1,1}+\frac{\operatorname{per}(B_{1,2})\operatorname{per}(B_{2,1})}{\operatorname{per}(A_{1,1})}.$$

Now assume the lemma holds for all $j<\ell$ for some $\ell\geq 3$. Then by Lemma 2.5.1,

$$\operatorname{per}(A)\operatorname{per}(A_{\ell-2,\ell-2})\leq\operatorname{per}(A_{\ell-1,\ell-1})\cdot\operatorname{per}(B_{\ell,\ell})+\operatorname{per}(B_{\ell,\ell-1})\operatorname{per}(B_{\ell-1,\ell}).$$

Diving the inequality by $\operatorname{per}(A_{\ell-1,\ell-1})\operatorname{per}(A_{\ell-2,\ell-2})$ gives

$$\frac{\operatorname{per}(A)}{\operatorname{per}(A_{\ell-1,\ell-1})}\leq\frac{\operatorname{per}(B_{\ell,\ell})}{\operatorname{per}(A_{\ell-2,\ell-2})}+\frac{\operatorname{per}(B_{\ell,\ell-1})\operatorname{per}(B_{\ell-1,\ell})}{\operatorname{per}(A_{\ell-1,\ell-1})\cdot\operatorname{per}(A_{\ell-2,\ell-2})}.\tag{2.48}$$

Using the inductive hypothesis on $B_{\ell,\ell}$, we get

$$\frac{\operatorname{per}(B_{\ell,\ell})}{\operatorname{per}(A_{\ell-2,\ell-2})}\leq a_{\ell,\ell}+\sum_{s=1}^{\ell-2}\frac{\operatorname{per}(B_{\ell,s})\operatorname{per}(B_{s,\ell})}{\operatorname{per}(A_{s,s})\operatorname{per}(A_{s-1,s-1})}$$

Here we used the fact that the submatrix of $B_{\ell,\ell}$ with row set $\{1,\dots,j-1\}\cup\{\ell\}$ and column set $\{1,\dots,j\}$ is $B_{\ell,j}$ and the submatrix of $B_{\ell,\ell}$ with row set $\{1,\dots,i\}$ and column set $\{1,\dots,i-1\}\cup\{\ell\}$ is $B_{i,\ell}$. Substituting this in Equation (2.48) gives

$$\frac{\operatorname{per}(A)}{\operatorname{per}(A_{\ell-1,\ell-1})} \le a_{\ell,\ell} + \sum_{s=1}^{\ell-1} \frac{\operatorname{per}(B_{\ell,s})\operatorname{per}(B_{s,\ell})}{\operatorname{per}(A_{s,s})\operatorname{per}(A_{s-1,s-1})}.$$

□

Corollary 2.5.4. *Let A be an $\ell\times\ell$ matrix with entries satisfying*

- $a_{i,j}\le f(j-i)$ *for* $i<j$
- $a_{i,j}\le \frac{f(\ell-i+j)}{f(\ell)}$ *for* $j<i$
- $a_{i,i}=1$.

Then $|\det(A)|\ge 1-\frac{0.1}{\ell}$.

Proof. Expanding the determinant of A gives

$$\det(A)=\sum_{\sigma\in\mathcal{S}_\ell}\operatorname{sign}(\sigma)\prod_{i=1}^{\ell}a_{i,\sigma(i)} = 1+\sum_{\sigma\in\mathcal{S}_\ell\setminus \mathrm{id}_\ell}\operatorname{sign}(\sigma)\prod_{i=1}^{\ell}a_{i,\sigma(i)}.$$

Where id_ℓ is the identity permutation. Using triangle inequality,

$$|\det(A)|\ge 1-\sum_{\sigma\in\mathcal{S}_\ell\setminus \mathrm{id}_\ell}\prod_{i=1}^{\ell}|a_{i,\sigma(i)}|. \tag{2.49}$$

Let $|A|$ denote the matrix whose entries are equal to the absolute values of A. Then by Theorem 2.5.2, we have $\operatorname{per}(|A|)\le 1+\frac{0.1}{\ell}$.

Expanding the permanent of $|A|$ gives

$$\operatorname{per}(|A|)=\sum_{\sigma\in\mathcal{S}_\ell}\prod_{i=1}^{\ell}|a_{i,\sigma(i)}| = 1+\sum_{\sigma\in\mathcal{S}_\ell\setminus id_\ell}\prod_{i=1}^{\ell}|a_{i,\sigma(i)}|.$$

Therefore, we have

$$\sum_{\sigma\in\mathcal{S}_\ell\setminus id_\ell}\prod_{i=1}^{\ell}|a_{i,\sigma(i)}|\le\frac{0.1}{\ell}. \tag{2.50}$$

Plugging the bounds from Equation (2.50) into (2.49) completes the proof. □

2.6 Future Directions

A central open question is whether an approximation algorithm with a guarantee of $O(1)^d$ can be designed, which would match the best-known hardness bounds. While our results achieve a $d^{O(d)}$-approximation, we have shown that this is, in fact, a tight bound for our matroid intersection–based approach. This suggests that fundamentally new algorithmic ideas are needed to make further progress.

2.7 Appendix for Chapter 2

2.7.1 Omitted proofs from Section 2.2

Proof of Lemma 2.2.4. Recall the statement: Let S be a set with $\mathrm{vol}(S) > 0$ and let $u_j \notin S$. Then for any $v_i \in S$, we have

$$w_0(u_j, v_i) = -\log \frac{\mathrm{vol}(S + u_j - v_i)}{\mathrm{vol}(S)}.$$

Assume without loss of generality that $v_i = v_1$ and $S = \{v_1, \ldots, v_d\}$. Let $u_j = \sum_{i=1}^{d} a_{i,j} v_i$. Also, write

$$v_1 = v_1^{\perp} + \sum_{i=2}^{d} b_i v_i,$$

where $v_1^{\perp}$ is orthogonal to the span of $S \setminus \{v_1\}$. Then we can express u_j as

$$u_j = a_{1,j} v_1^{\perp} + \sum_{i=2}^{d} (a_{1,j} b_i + a_{i,j}) v_i.$$

Let $\mathrm{vol}(X)$ denote the k-dimensional volume of the parallelepiped spanned by a set $X \subseteq \mathbb{R}^d$ with $|X| = k$. Then,

$$\mathrm{vol}(S) = \mathrm{vol}(S \setminus \{v_1\}) \cdot \|v_1^{\perp}\|,$$

and

$$\mathrm{vol}(S + u_j - v_1) = \mathrm{vol}(S \setminus \{v_1\}) \cdot |a_{1,j}| \cdot \|v_1^{\perp}\|,$$

since adding u_j in place of v_1 contributes a component orthogonal to $S \setminus \{v_1\}$ of magnitude $|a_{1,j}| \cdot \|v_1^{\perp}\|$.

Thus,

$$-\log \frac{\mathrm{vol}(S + u_j - v_1)}{\mathrm{vol}(S)} = -\log \frac{\mathrm{vol}(S \setminus \{v_1\}) \cdot |a_{1,j}| \cdot \|v_1^{\perp}\|}{\mathrm{vol}(S \setminus \{v_1\}) \cdot \|v_1^{\perp}\|} = -\log |a_{1,j}|,$$

which completes the proof. □

Proof of Observation 2.2.7. Recall the statement: If C is an f-violating cycle, then

$$\prod_{(u,v)\in C: u\in L,\, v\in R} |a_{vu}| > f(|C|/2).$$

Since C is f-violating, the total weight under the w_0 function satisfies

$$\ell(C) := \sum_{(u,v)\in C} w_0(u,v) < -\log f(|C|/2).$$

Note that $w_0(u,v)$ is nonzero only for forward arcs ($u \in L, v \in R$). Therefore,

$$\ell(C) = \sum_{(u,v)\in C: u\in L, v\in R} w_0(u,v) = -\log\left(\prod_{(u,v)\in C: u\in L, v\in R} |a_{vu}|\right).$$

Combining both expressions for $\ell(C)$, we get:

$$-\log\left(\prod_{(u,v)\in C: u\in L, v\in R} |a_{vu}|\right) < -\log f(|C|/2).$$

Exponentiating both sides yields the desired inequality:

$$\prod_{(u,v)\in C: u\in L, v\in R} |a_{vu}| > f(|C|/2).$$

□

Proof of Lemma 2.2.9. Algorithm 1 searches for an f-violating cycle in $G(S)$ of minimum length. In the i-th iteration, it checks for a negative-weight cycle of exactly $2i$ edges using the weight function w_i. This is done by running Bellman-Ford (see Chapter 8, Section 8.3 of [172]) from each vertex for $2i$ steps, and checking whether any vertex has a negative-distance path to itself.

A negative cycle under w_i with at most $2i$ edges corresponds to an f-violating cycle. Since earlier iterations ruled out such cycles with fewer than $2i$ edges, any cycle found in iteration i must use exactly $2i$ edges. Moreover, since w_i is an increasing function of i, any shorter cycle C' with $2i' < 2i$ that was not f-violating under $w_{i'}$ cannot become negative under w_i.

Now, suppose there exists an f-violating cycle C with $|C| = 2\ell$. Under weights w_ℓ, we have:

$$w_\ell(C) = \sum_{(u,v)\in C}\left(\frac{\log f(\ell)}{\ell} + w_0(u,v)\right) = \log f(\ell) + w_0(C) < 0,$$

since $w_0(C) < -\log f(\ell)$. Thus, C is a negative-weight cycle under w_ℓ, and the algorithm will detect it.

Finally, let C be the cycle returned by the algorithm. If a shorter f-violating cycle C' existed, it would have been detected in an earlier iteration, contradicting the choice of C. Hence, C is minimal. □

Proof of Lemma 2.2.11. Recall the statement of the Lemma 2.2.11: Let S be a basis, let X and Y be sets with $|X| = |Y| = \ell$ and $Y \subseteq S$. Let A be the $d \times \ell$ matrix of coefficients so that $V_X = V_S A$, and let A_C be the $\ell \times \ell$ submatrix of only the coefficients corresponding to columns in Y. If $T = (S \cup X) \backslash Y$ then $\mathrm{vol}(T)^2 = \mathrm{vol}(S)^2 \cdot \det(A_C A_C^\top)$.

Order the columns of V_S so that Y makes up the first ℓ columns of V_S. Let A' be the $(d-\ell) \times \ell$ submatrix of A consisting of the remaining columns not already in A_C. Then

$$V_T = V_S \begin{bmatrix} A_C & 0 \\ A' & I_{d-\ell} \end{bmatrix},$$

which implies that

$$\det(T) = \det(S) \cdot \det(A_C).$$

□

2.7.2 Ommited Lemmas and Proofs from Section 2.3

Linear Algebraic Lemmas, $r < d$

Lemma 2.7.1. *Let* $S, T \subseteq [n]$ *such that* $\det(V_S^\top V_S) > 0$, $\det(V_T^\top V_T) > 0$ *and* $|S| = |T|$. *Then the ratio of their squared volumes can be expressed as*

$$\frac{\mathrm{vol}^2(T)}{\mathrm{vol}^2(S)} = \det \begin{bmatrix} B^\top B & -A^\top \\ A & (V_S^\top V_S)^{-1} \end{bmatrix}. \tag{2.51}$$

Where $B := \mathrm{proj}_S^\perp \cdot V_T$ *and* $A := (V_S^\top V_S)^{-1} V_S^\top V_T$.

Proof. The matrix in RHS of Equation (2.51) can be expanded as

$$\begin{bmatrix} B^\top B & -A^\top \\ A & (V_S^\top V_S)^{-1} \end{bmatrix} = \begin{bmatrix} V_T^\top \left(I - V_S (V_S^\top V_S)^{-1} V_S^\top\right) V_T & -V_T^\top V_S (V_S^\top V_S)^{-1} \\ (V_S^\top V_S)^{-1} V_S^\top V_T & (V_S^\top V_S)^{-1} \end{bmatrix}$$

Using Schur's determinant formula,

$$\begin{aligned} \det \begin{bmatrix} B^\top B & -A^\top \\ A & (V_S^\top V_S)^{-1} \end{bmatrix} &= \det(V_S^\top V_S)^{-1} \det\left(B^\top B + A^\top V_S^\top V_S A\right) \\ &= \frac{\det\left(B^\top B + A^\top V_S^\top V_S A\right)}{\det\left(V_S^\top V_S\right)}. \end{aligned}$$

It suffices to show that

$$\det(V_T^\top V_T) = \det\left(B^\top B + A^\top V_S^\top V_S A\right). \tag{2.52}$$

On the other hand, $V_T = V_S A + B$ by definition of A and B. Furthermore,

$$V_S^\top B = V_S^\top \left(I - V_S (V_S^\top V_S)^{-1} V_S^\top\right) V_T = 0.$$

As a result, $V_T^\top V_T = B^\top B + A^\top V_S^\top V_S A$ and Equation (2.52) follows. □

The following corollary shows that the ratio of volumes only depends on the symmetric difference of the two sets.

Corollary 2.7.2. *Let $S \subseteq [n]$ with $\mathrm{vol}(S) > 0$ and let $Y \subseteq S$, $X \subseteq [n] \backslash S$ with $|X| = |Y|$ such that $\mathrm{vol}(S - Y + X) > 0$. Let $V_X = V_S A + B$ where $A_{i,j} = \mathrm{coef}_S(X_j, S_i)$ and $B^\top V_S = 0$. Then*

$$\frac{\mathrm{vol}^2(S - Y + X)}{\mathrm{vol}^2(S)} = \det \begin{bmatrix} B^\top B & -A_{Y,X}^\top \\ A_{Y,X} & (V_S^\top V_S)^{-1}_{Y,Y} \end{bmatrix}. \tag{2.53}$$

.

Proof. Let $T = S - Y + X$ and $R = S \cap T$. Reorder elements of S and T. Then

$$V_T = V_S \begin{bmatrix} A_{Y,X} & 0 \\ A_{R,X} & I_{|R|} \end{bmatrix} + \begin{bmatrix} B & 0 \end{bmatrix}.$$

By Lemma 2.7.1,

$$\begin{aligned} \frac{\mathrm{vol}^2(S - Y + X)}{\mathrm{vol}^2(S)} &= \det \begin{bmatrix} B^\top B & 0 & -A_{Y,X}^\top & -A_{R,X}^\top \\ 0 & 0 & 0 & I_{|R|} \\ A_{Y,X} & 0 & (V_S^\top V_S)^{-1}_{Y,Y} & (V_S^\top V_S)^{-1}_{Y,R} \\ A_{Y,X} & I_{|R|} & (V_S^\top V_S)^{-1}_{R,Y} & (V_S^\top V_S)^{-1}_{R,R} \end{bmatrix} \\ &= \det \begin{bmatrix} B^\top B & -A_{Y,X}^\top \\ A_{Y,X} & (V_S^\top V_S)^{-1}_{Y,Y} \end{bmatrix}. \end{aligned}$$

□

Finally, the ratio of volumes is lower bounded by the determinant matrix of the coefficient function.

Corollary 2.7.3. *Let $S \subseteq [n]$ with $\mathrm{vol}(S) > 0$, and let $Y \subseteq S$, $X \subseteq [n] \setminus S$ with $|X| = |Y|$ such that $\mathrm{vol}(S - Y + X) > 0$. Then*

$$\frac{\mathrm{vol}^2(S - Y + X)}{\mathrm{vol}^2(S)} \geq \det(A_{Y,X})^2, \tag{2.54}$$

where $V_X = V_S A + B$ with $B^\top V_S = 0$.

Proof. Using Corollary 2.7.2, we have

$$\frac{\mathrm{vol}^2(S - Y + X)}{\mathrm{vol}^2(S)} = \det \begin{bmatrix} B^\top B & -A_{Y,X}^\top \\ A_{Y,X} & (V_S^\top V_S)^{-1}_{Y,Y} \end{bmatrix}.$$

Using Lemma 2.7.4 completes the proof. □

Lemma 2.7.4. *Let M be a square block matrix defined as $M = \begin{pmatrix} A & B \\ -B^T & C \end{pmatrix}$, where A, B, and C are $\ell \times \ell$ matrices. If A and C are positive semidefinite (PSD), then $\det(M) \geq \det(B)^2$.*

Proof. First, assume A is positive definite (PD), which implies it is invertible. The determinant of the block matrix M can be expressed using the Schur's formula:

$$\det(M) = \det(A)\det(C + B^T A^{-1} B)$$

Since A is PD, its inverse A^{-1} is also PD. The matrix $B^T A^{-1} B$ is a congruence of a PD matrix, making it PSD. Given that C is also PSD, and the sum of two PSD matrices is PSD, we can use the property that for any two $\ell \times \ell$ PSD matrices X and Y, $\det(X + Y) \geq \det(Y)$. Applying this with $X = C$ and $Y = B^T A^{-1} B$, we get:

$$\det(C + B^T A^{-1} B) \geq \det(B^T A^{-1} B)$$

Substituting this inequality back into the expression for $\det(M)$:

$$\det(M) \geq \det(A)\det(B^T A^{-1} B) = \det(A)\det(B^T)\det(A^{-1})\det(B)$$

Since $\det(B^T) = \det(B)$ and $\det(A^{-1}) = 1/\det(A)$, this simplifies to:

$$\det(M) \geq \det(B)^2$$

This result extends from the PD case to the general PSD case for A by a continuity argument, since any PSD matrix can be viewed as the limit of a sequence of PD matrices (e.g., $A_\varepsilon = A + \varepsilon I$ as $\varepsilon \to 0^+$). □

Lemma 2.7.5 (Determinant rank-2 update)**.** *Let $P \in \mathbb{R}^{\ell \times \ell}$ and $x, y, z, w \in \mathbb{R}^\ell$. Define*

$$Q := c_1 y x^\top + c_2 z x^\top + c_3 y w^\top + c_4 z w^\top$$

for scalars $c_1, c_2, c_3, c_4 \in \mathbb{R}$, and let $B := P + Q$. Then:

$$\begin{aligned} |\det(B)| \geq |\det(P)| &- \sum_{j=1}^{\ell} \left|\det\left(P^{(1)}, \ldots, Q^{(j)}, \ldots, P^{(\ell)}\right)\right| \\ &- \sum_{j \neq k} \left|\det\left(P^{(1)}, \ldots, c_1 y x_j, \ldots, c_4 z w_k, \ldots, P^{(\ell)}\right)\right| \\ &- \sum_{j \neq k} \left|\det\left(P^{(1)}, \ldots, c_2 z x_j, \ldots, c_3 y w_k, \ldots, P^{(\ell)}\right)\right|. \end{aligned}$$

Proof. Define rank-1 matrices $Q_1 := c_1 y x^\top$, $Q_2 := c_2 z x^\top$, $Q_3 := c_3 y w^\top$, $Q_4 := c_4 z w^\top$, and let $Q_0 := P$. By multilinearity:

$$\det(P + Q) = \det(Q_0 + Q_1 + Q_2 + Q_3 + Q_4) = \sum_{h \in \mathcal{H}} \det(Q[h]),$$

where $\mathcal{H} := \{h : [\ell] \to \{0, 1, 2, 3, 4\}\}$ and $Q[h]$ denotes the matrix with column j from $Q_{h(j)}$.

Since each Q_i is rank-1, $\det(Q[h]) = 0$ whenever two columns come from the same Q_i with $i > 0$. Moreover, since the column span of Q_1, Q_3 and Q_2, Q_4 are equal, $\det(Q[h]) = 0$ when two columns have a non-zero index with the same parity. Thus, we restrict to:

1. $\mathcal{H}_a$: only one column from Q_a, rest from Q_0,
2. $\mathcal{H}_{a,b}$: two distinct columns from Q_a and Q_b, rest from Q_0.

Then:

$$\det(P+Q) = \det(P) + \sum_{a=1}^{4} \sum_{h \in \mathcal{H}_a} \det(Q[h]) + \sum_{a<b} \sum_{h \in \mathcal{H}_{a,b}} \det(Q[h]).$$

Observe:

1. $\mathcal{H}_{1,3}$ and $\mathcal{H}_{2,4}$ yield zero contributions since their columns are linearly dependent (all colinear with x or w),
2. Pairs $(1,2)$ and $(3,4)$ cancel: for $h \in \mathcal{H}_{1,2}$ with $h(j_1) = 1$, $h(j_2) = 2$,

   $$\det(Q[h]) = -\det(Q[\hat{h}]),$$

   where $\hat{h}$ flips indices. So, $\sum_{h \in \mathcal{H}_{1,2}} \det(Q[h]) = 0$, and likewise for $(3,4)$.

Only $\mathcal{H}_a$ (single-column replacements) and $\mathcal{H}_{1,4}$, $\mathcal{H}_{2,3}$ remain.

Finally, use multilinearity to write:

$$\sum_{a=1}^{4} \sum_{h \in \mathcal{H}_a} \det(Q[h]) = \sum_{j=1}^{\ell} \det\left(P^{(1)}, \ldots, Q^{(j)}, \ldots, P^{(\ell)}\right),$$

and expand the remaining $\mathcal{H}_{1,4}$ and $\mathcal{H}_{2,3}$ terms explicitly to get the full bound.

This completes the proof. □

Proof of Lemma 2.3.15. For part (a), the product of the diagonal entries of B is exactly

$$\prod_{i=1}^{\ell} b_{i,i} = \exp(-w(C)) \geq f(\ell)\,,$$

where the last inequality follows from the fact that C is an f-violating cycle.

For part (b), we first bound every off-diagonal entry of matrix B as a function of its diagonal entries and then apply Lemma 2.5.2.

We will show that B satisfies $b_{i,j} \leq f(j-i)/\prod_{s=i+1}^{j-1} w_{s,s}$ when $i < j$ and $w_{i,j} \leq \frac{f(\ell-i+j)}{f(\ell)} \prod_{s=j}^{i} b_{s,s}$ when $j < i < \ell$ and $b_{\ell,j} \leq f(j)/\prod_{s=1}^{j-1} b_{s,s}$ when $j \leq \ell$. In this case, apply the following operations to B to get $\widehat{B}$:

- Multiply the columns $1 < j \leq \ell$ with $\prod_{s=1}^{j-1} b_{s,s}$.
- Divide the rows $1 \leq i \leq \ell - 1$ by $\prod_{s=1}^{i} b_{s,s}$.
- Divide last row by $f(\ell)$.

Then $\operatorname{per}(B) = \operatorname{per}(\widehat{B}) \cdot f(\ell)$ and the entries of $\widehat{B}$ satisfy

- $\widehat{b}_{i,i} = 1$ for $i \in [\ell - 1]$ and $b_{\ell,\ell} = \prod_{s=1}^{\ell} b_{s,s}/f(\ell) \geq 1$
- $\widehat{b}_{i,j} \leq f(i-j)$ for $i < j$ and
- $\widehat{b}_{i,j} \leq f(\ell - i + j)/f(\ell)$ for $j < i$

By Theorem 2.5.2, $\operatorname{per}(\widehat{B}) \leq (1 + 0.1/\ell) \prod_{t=1}^{\ell} \widehat{b}_{t,t} = (1 + 0.1/\ell) \prod_{t=1}^{\ell} b_{t,t}/f(\ell)$. As a result,

$$\operatorname{per}(B) = f(\ell) \cdot \operatorname{per}(\widehat{B}) \leq (1 + 0.1/\ell) \prod_{t=1}^{\ell} b_{t,t} \leq 1.05 \prod_{t=1}^{\ell} b_{t,t}.$$

For $i, j \in [\ell]$ with $i < j$, define the cycles $C^{\mathrm{I}}_{i,j} := (u_j \xrightarrow{\mathrm{I}} v_i \to u_{i+1} \xrightarrow{\mathrm{I}} v_{i+1} \dots v_{j-1} \to u_j)$ and $C^{\mathrm{II}}_{i,j} := (u_j \xrightarrow{\mathrm{II}} v_i \to u_{i+1} \xrightarrow{\mathrm{I}} v_{i+1} \dots v_{j-1} \to u_j)$. Both $C^{\mathrm{I}}_{i,j}$ and $C^{\mathrm{II}}_{i,j}$ contain $2(j-i)$ arcs and vertex sets $\operatorname{ver}(C^{\mathrm{I}}_{i,j}) = \operatorname{ver}(C^{\mathrm{II}}_{i,j})$ are a proper subset of $\operatorname{ver}(C)$. So, by minimality of C, we know that $C^{\mathrm{I}}_{i,j}$ and $C^{\mathrm{II}}_{i,j}$ are not f-violating cycles.

Therefore, $\exp(-w(C^{\mathrm{I}}_{i,j})) = |\operatorname{coef}_S(u_j, v_i)| \cdot \prod_{s=i+1}^{j-1} \exp(-w(u_s \xrightarrow{\mathrm{I}} v_s)) < f(j-i)$, and

$$|\operatorname{coef}_S(u_j, v_i)| < \frac{f(j-i)}{\prod_{s=i+1}^{j-1} \exp(-w(u_s \xrightarrow{\mathrm{I}} v_s))}. \tag{2.55}$$

Using a similar argument for $C^{\mathrm{II}}_{i,j}$ gives

$$\left\|u_j^{\perp}\right\| / \left\|v_i^{\perp}\right\| < \frac{f(j-i)}{\prod_{s=i+1}^{j-1} \exp(-w(u_s \xrightarrow{\mathrm{I}} v_s))}. \tag{2.56}$$

Combining (2.55) and (2.56), we get

$$\begin{aligned} w_{i,j} &= \max\{\, |\operatorname{coef}_S(u_j, v_i)|,\ \left\|u_j^{\perp}\right\| / \left\|v_i^{\perp}\right\| \,\} \\ &\leq \frac{f(j-i)}{\prod_{s=i+1}^{j-1} \exp(-w(u_s \xrightarrow{\mathrm{I}} v_s))} = \frac{f(j-i)}{\prod_{s=i+1}^{j-1} |b_{s,s}|}. \end{aligned}$$

For $i, j \in [\ell - 1]$ with $j < i$, define $C^{\mathrm{I}}_{i,j} := (v_\ell \to u_1 \xrightarrow{\mathrm{I}} v_1 \dots u_j \xrightarrow{\mathrm{I}} v_i \to u_{i+1} \dots u_\ell \xrightarrow{\mathrm{II}} v_\ell)$ and $C^{\mathrm{II}}_{i,j} := (v_\ell \to u_1 \xrightarrow{\mathrm{I}} v_1 \dots u_j \xrightarrow{\mathrm{II}} v_i \to u_{i+1} \dots u_\ell \xrightarrow{\mathrm{II}} v_\ell)$. Both $C^{\mathrm{I}}_{i,j}$ and $C^{\mathrm{II}}_{i,j}$ contain $2(\ell - i + j)$ arcs and $\operatorname{ver}(C^{\mathrm{I}}_{i,j}) = \operatorname{ver}(C^{\mathrm{II}}_{i,j})$ is a proper subset of $\operatorname{ver}(C)$. So, they are not f-violating cycles. Therefore,

$$\begin{aligned} \exp(-w(C^{\mathrm{I}}_{i,j})) &= |\operatorname{coef}_S(u_j, v_i)| \cdot \prod_{s=1}^{j-1} \exp(-w(u_s \xrightarrow{\mathrm{I}} v_s)) \\ &\quad \cdot \prod_{k=i+1}^{\ell-1} \exp(-w(u_s \xrightarrow{\mathrm{I}} v_s)) \cdot \exp(-w(u_\ell \xrightarrow{\mathrm{II}} v_\ell)) \\ &< f(\ell - j + i). \end{aligned} \tag{2.57}$$

Since C is an f-violating cycle, we also have

$$\prod_{s=1}^{\ell-1} \exp(-w(u_s \xrightarrow{\text{I}} v_s)) \cdot \exp(-w(u_\ell \xrightarrow{\text{II}} v_\ell)) > f(\ell). \tag{2.58}$$

Dividing (2.57) by (2.58) gives

$$|\operatorname{coef}_S(u_j, v_i)| < \frac{f(\ell - i + j)}{f(\ell)} \cdot \prod_{s=j}^{i} \exp(-w(u_s \xrightarrow{\text{I}} v_s)) \,. \tag{2.59}$$

Similarly,

$$\left\|u_j^\perp\right\| / \left\|v_i^\perp\right\| < \frac{f(\ell - i + j)}{f(\ell)} \cdot \prod_{s=j}^{i} \exp(-w(u_s \xrightarrow{\text{I}} v_s)) \,. \tag{2.60}$$

Summing (2.59) and (2.60) gives

$$\begin{aligned} b_{i,j} = \max\{|\operatorname{coef}_S(u_j, v_i)|,\, \left\|u_j^\perp\right\| / \left\|v_i^\perp\right\| \} &\leq \frac{f(\ell - i + j)}{f(\ell)} \cdot \prod_{s=j}^{i} \exp(-w(u_s \xrightarrow{\text{I}} v_s)) \\ &\leq \frac{f(\ell - i + j)}{f(\ell)} \cdot \prod_{s=j}^{i} |b_{s,s}| \,. \end{aligned}$$

To bound $w_{\ell,j}$ for $j \leq \ell$, consider cycles $C^{\text{I}}_{\ell,j} := (v_\ell \to u_1 \xrightarrow{\text{I}} v_1 \dots u_j \xrightarrow{\text{I}} v_\ell)$ and $C^{\text{II}}_{\ell,j} := (v_\ell \to u_1 \xrightarrow{\text{I}} v_1 \dots u_j \xrightarrow{\text{II}} v_\ell)$. Both $C^{\text{I}}_{\ell,j}$ and $C^{\text{II}}_{\ell,j}$ contain $2i$ arcs and $\operatorname{ver}(C^{\text{I}}_{i,\ell}) = \operatorname{ver}(C^{\text{II}}_{i,\ell})$ is a proper subset of $\operatorname{ver}(C)$. So, they are not f-violating cycles. Following a similar argument to the $i < j < \ell$ case and comparing $w(C^{\text{I}}_{\ell,j}), w(C^{\text{I}}_{\ell,j})$ with $w(C)$ gives

$$b_{\ell,j} \leq \frac{f(j)}{f(\ell)} \cdot \prod_{s=j}^{\ell-1} |b_{s,s}| \cdot \exp(-w(u_\ell \xrightarrow{\text{II}} v_\ell)) < \frac{f(j)}{f(\ell)} \cdot \prod_{k=i}^{\ell-1} |w_{k,k}| \cdot w_{\ell,\ell} \,.$$

For part (c), the principal submatrix $W_{\ell-1,\ell-1}$ satisfies first two prerequisites of Theorem 2.5.2. Observe that $f(y-x)/f(y)$ is a non-increasing function of y for any fixed $x \in [\ell]$. Therefore, for any $j < i$,

$$b_{i,j} \leq \frac{f(\ell - i + j)}{f(\ell)} \cdot \prod_{s=j}^{i} b_{s,s} \leq \frac{f(\ell - 1 - i + j)}{f(\ell - 1)} \cdot \prod_{s=j}^{i} b_{s,s} \,.$$

So $W_{\ell-1,\ell-1}$ also satisfies the final prerequisite of 2.5.2. □

2.7.3 Examples

Example with all negative cycles containing a type II edge.

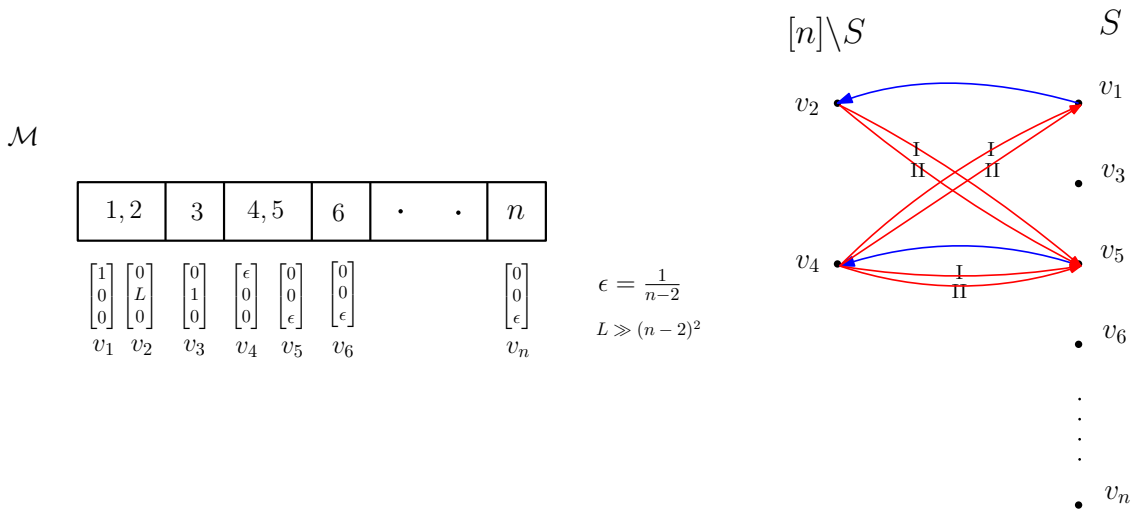
Example 2.7.6. Let $\mathcal{M}$ be a partition matroid on $[n]$ with partition $[n] = \{1,2\} \cup \{3\} \cup \{4,5\} \cup \{6\} \cup \cdots \cup \{n\}$ where the rank of each partition is 1. So the rank of the matroid is $n-2$. We will consider vectors in $\mathbb{R}^3$. Let the vectors associated with the matroid be $v_1 = e_1, v_2 = Le_2, v_3 = e_2, v_4 = \epsilon e_1$ and $e_5 = \ldots = e_n = \epsilon e_3$, where $\epsilon = 1/(n-2)$ and $L\epsilon^2 \gg 1$.

The optimal solution of the determinant maximization problem on $\mathcal{M}$ is $T = \{Le_2, e_2, \epsilon e_1, \epsilon e_3, \ldots, \epsilon e_3\}$ with $\det(TT^\top) = (L^2+1) \cdot \epsilon^2 \cdot (1-\epsilon^2)$. Let the current solution S be $\{e_1, e_2, \epsilon e_3, \ldots, \epsilon e_3\}$. Then $SS^\top = I$, $\det(SS^\top) = 1$.

The cycles in $G(S)$ along with their weights are

- $C_1 = (v_4 \xrightarrow{\text{I}} v_5)$ with $w(C_1) = \infty$
- $C_2 = (v_4 \xrightarrow{\text{II}} v_5)$ with $w(C_2) = -\log(1+\epsilon^2)(1-\epsilon^2) > 0$
- $C_3 = (v_1 \to v_2 \xrightarrow{\text{I}} v_5 \to v_4 \xrightarrow{\text{I}} v_1)$ with $w(C_3) = \infty$
- $C_4 = (v_1 \to v_2 \xrightarrow{\text{I}} v_5 \to v_4 \xrightarrow{\text{II}} v_1)$ with $w(C_4) = \infty$
- $C_5 = (v_1 \to v_2 \xrightarrow{\text{II}} v_5 \to v_4 \xrightarrow{\text{I}} v_1)$ with $w(C_5) = -\log((1+L^2) \cdot (1-\epsilon^2)) - \log(\epsilon^2)$
- $C_6 = (v_1 \to v_2 \xrightarrow{\text{II}} v_5 \to v_4 \xrightarrow{\text{II}} v_1)$ with $w(C_6) = \infty$

The only cycle with negative weight is C_5 and it contains a type II edge.



(a) Matroid *(b) Graph*

Figure 2.4: Example

Chapter 3

New Permanent Inequalities

3.1 Introduction

The permanent of a matrix, despite its deceptively simple definition, is notoriously difficult to compute. It is well-known that the exact computation of the permanent is #P-complete [184], situating it at the forefront of complexity theory and establishing its computational intractability for all but trivially sized matrices. Despite this difficulty, permanents play a critical role across diverse areas such as combinatorics, graph theory (particularly in counting perfect matchings in bipartite graphs [140]), quantum computing (specifically within boson sampling experiments aimed at demonstrating quantum advantage [1]), and statistical physics (e.g., in dimer covering models [108]).

Due to the permanent's computational complexity, significant effort has been directed towards deriving efficient upper bounds and approximation algorithms. Classical results, such as the Bregman-Minc inequality and its numerous refinements, form a rich and diverse body of work (e.g., [33, 96, 109, 128, 137, 139, 140, 165, 167, 169, 176, 178]). However, existing bounds have largely emerged from combinatorial, scaling, or probabilistic frameworks. This work charts a different course by developing a deterministic, iterative procedure rooted in linear algebraic principles. This creates a new algorithmic pathway that is not only theoretically sound but also directly computable.

This chapter introduces a new pathway for establishing upper bounds on the permanent of non-negative matrices. Our approach systematically adapts powerful and intuitive tools from determinant theory. Our main contributions are:

- **A Permanental Inverse:** We define a novel analogue of the matrix inverse, constructed from permanents of submatrices. While it lacks multiplicative inverse properties, this *permanental inverse* satisfies key structural inequalities that enable further analysis.

  Definition 3.1.1 (Permanental Inverse)**.** The *permanental inverse* of a non-negative matrix

$B \in \mathbb{R}_{\geq 0}^{d \times d}$ with $\operatorname{per}(B) \neq 0$ is the matrix C with entries

$$c_{i,j} = \frac{\operatorname{per}(B_{j,i})}{\operatorname{per}(B)},$$

where $B_{j,i}$ is the matrix obtained by removing the j^{th} row and i^{th} column of B. Use B^* to denote the permanental inverse of B. See §3.3.1 for more details regarding the permanental inverse.

- **Schur's Formula for Permanents:** Leveraging the permanental inverse, we establish an analogue of the Schur's formula for determinants to permanents. This result provides the core theoretical engine for our framework.

  Theorem 3.1.2 (Permanental Schur's Formula)**.** *Let $A \in \mathbb{R}_{\geq 0}^{n \times n}$ be a block matrix of the form*

  $$A = \begin{bmatrix} B & Y \\ X^\top & W \end{bmatrix}, \tag{3.1}$$

  where $B \in \mathbb{R}^{d \times d}$ has non-zero permanent. Then the permanent of A satisfies

  $$\operatorname{per}(A) \leq \operatorname{per}(B) \cdot \operatorname{per}\left(W + X^\top B^* Y\right). \tag{3.2}$$

  See §3.3.3 for a proof of Theorem 3.1.2.

- **A Constructive Algorithmic Bound:** We design an iterative procedure, the *Permanent Process*, inspired by Gaussian elimination. This algorithm yields a provable upper bound on the permanent, offering a transparent and constructive alternative to more abstract combinatorial bounds. See §3.3.4 for more details regarding the permanent process and its properties.

- **Provable Guarantees for Structured Matrices:** We show that for matrices exhibiting approximate diagonal dominance—a structure common in numerical linear algebra and network models—our bound yields strong theoretical guarantees. See §3.3.6 for more details.

Taken together, these results offer a fresh algorithmic perspective on permanents and expand the analytical toolbox for bounding them, with potential applications in combinatorics, statistical physics, and quantum computation.

3.2 Preliminaries

3.2.1 Notation

To discuss matrices, we will use the following standard notation. Let A be an $n \times n$ matrix with real-valued entries, denoted $A \in \mathbb{R}^{n \times n}$.

- **Matrix Entries:** $(A)_{i,j}$ or $a_{i,j}$ (the corresponding lowercase letter) refers to the entry in the i^{th} row and j^{th} column of A.
- **Submatrices by Selection:** For index sets $S, T \subseteq \{1, \ldots, n\}$, we denote by $A(S, T)$ the submatrix formed by taking the rows indexed by S and columns indexed by T.
- **Submatrices by Deletion:** We use several notations for submatrices formed by deleting rows or columns.
  - The matrix $A_{-i,\cdot}$ denotes the matrix obtained by deleting the i^{th} row and $A_{\cdot,-j}$ denotes the matrix obtained by deleting the j^{th} column.
  - $A_{i,j}$ denotes the matrix obtained by deleting both the i^{th} row and the j^{th} column.
  - For deleting multiple rows and columns, the notation $A(-S, -T)$ is shorthand for the submatrix formed by deleting the rows in set S and columns in set T.
- **Determinants and Permanents:**
  - $\det_A(S, T) := \det(A(S, T))$. For brevity, we often write $|A|$ for $\det(A)$.
  - $\mathrm{per}_A(S, T) := \mathrm{per}(A(S, T))$.
  - By convention, $\det_A(\emptyset, \emptyset) = \mathrm{per}_A(\emptyset, \emptyset) = 1$.
- **Entrywise Inequality:** The expression $A \geq B$ means that every entry in A is greater than or equal to the corresponding entry in B (i.e., $a_{i,j} \geq b_{i,j}$ for all i, j).
- **Functions:** For any function $f : [k] \to [d]$, let $\mathrm{img}_{f,S} := \{f(j) : j \in S\}$ be the image of S according to f; we simply write img_f when $S = [k]$.

3.2.2 Gaussian Elimination

Remark 3.2.1 (A Note on Convention)**.** The method described here is a specific variant of Gaussian elimination designed to produce a **lower triangular** matrix. This is a non-standard convention, as the standard algorithm is typically defined to produce an *upper triangular* matrix.

Let $A^{(t)}$ denote the state of the matrix at the beginning of step t, with the initial matrix being $A^{(1)} = A$. The goal is to iteratively transform A into a lower triangular matrix. The state of the matrix entries after the end of step t for some $1 \leq t \leq n-1$ is given by

$$a_{i,j}^{(t+1)} = \begin{cases} a_{i,j}^{(t)} - \frac{a_{i,t}^{(t)} a_{t,j}^{(t)}}{a_{t,t}^{(t)}}, & \text{for } j \geq t+1 \\ a_{i,j}^{(t)}, & \text{Otherwise.} \end{cases} \tag{3.3}$$

In simpler terms, at each step t, this process uses the pivot element $a_{t,t}$ to create zeros in all entries to its right, within the same row t.

Theorem 3.2.2 (Gaussian Elimination Invariant)**.** *The entries of the matrix $A^{(t)}$ are ratios of determinants of certain sub-matrices of A:*

$$a_{i,j}^{(t)} = \frac{\det_A([r-1]+\{i\},[r-1]+\{j\})}{\det_A([r-1],[r-1])}, \quad r = \min(j,t). \tag{3.4}$$

Corollary 3.2.3 (Determinant Property)**.** . *If $A^{(n)}$ is the final lower triangular matrix obtained after running the full elimination process on A, the product of its diagonal entries equals the determinant of A.*

$$\det(A) = \prod_{i=1}^{n} a_{i,i}^{(n)} \tag{3.5}$$

3.2.3 Schur's Formula

The Schur complement is a fundamental tool for working with block matrices.

Theorem 3.2.4 (Schur's Determinant Formula)**.** *Let $A \in \mathbb{R}^{n\times n}$ be a block matrix of the form*

$$A = \begin{bmatrix} B & Y \\ X^\top & W \end{bmatrix},$$

where $B \in \mathbb{R}^{d\times d}$ is an invertible matrix. Then the determinant of A is given by

$$\det(A) = \det(B)\cdot\det(W - X^\top B^{-1}Y). \tag{3.6}$$

*The matrix $S = W - X^\top B^{-1}Y$ is called the **Schur complement** of B in A.*

3.3 Generalizing Determinantal Concepts for the Permanent

This section extends familiar concepts like the matrix inverse and Schur's formula—which are fundamental for determinants—to the context of the matrix permanent.

3.3.1 The Permanental Inverse

Motivation from the Determinant

To start, recall that one way to define the inverse of an invertible matrix B is using Cramer's rule, where each entry of the inverse $C = B^{-1}$ is given by:

$$c_{i,j} = \frac{\det(B_{j,i})}{\det(B)}$$

Here, $B_{j,i}$ is the submatrix of B formed by removing row j and column i. This definition naturally gives us $B^{-1}B = BB^{-1} = I$. This provides a direct template for defining a similar concept for the permanent.

Definition 3.3.1 (Permanental Inverse)**.** For a non-negative matrix $B \in \mathbb{R}_{\geq 0}^{d \times d}$ with $\operatorname{per}(B) > 0$, the **permanental inverse**, denoted B^*, is the matrix with entries:

$$(B^*)_{i,j} = \frac{\operatorname{per}(B_{j,i})}{\operatorname{per}(B)}$$

Crucial Differences and Properties

Unlike the determinantal inverse, B^* does not typically recover the identity matrix. Instead, it satisfies a matrix inequality.

Claim 3.3.2. For a non-negative matrix B, we have $B^*B \geq I$ and $BB^* \geq I$. In general, $B^*B \neq BB^*$.

Proof. Evaluate the diagonal entries of B^*B as

$$(B^*B)_{ii} = \frac{1}{\operatorname{per}(B)} \sum_{j=1}^{d} b_{ji} \cdot \operatorname{per}(B_{j,i})$$

The sum $\sum_{j=1}^{d} b_{ji} \cdot \operatorname{per}(B_{j,i})$ is the Laplace expansion of the permanent of B along column i, which equals $\operatorname{per}(B)$. Thus, the diagonal entries are $(B^*B)_{ii} = \frac{\operatorname{per}(B)}{\operatorname{per}(B)} = 1$.

For the off-diagonal entries (where $k \neq i$), since B is a non-negative matrix, all its permanents and entries are non-negative. Thus, every term in the sum for $(B^*B)_{ik}$ is non-negative, meaning $(B^*B)_{ik} \geq 0$.

Combining these two points, the diagonal entries of B^*B are 1 and the off-diagonal entries are non-negative. By definition, this means $B^*B \geq I$. The proof for $BB^* \geq I$ follows a similar argument. □

Example. Let $B = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$. The permanent is $\operatorname{per}(B) = 1 \cdot 4 + 2 \cdot 3 = 10$. The permanental inverse B^* is:

$$B^* = \frac{1}{10} \begin{pmatrix} \operatorname{per}(B_{1,1}) & \operatorname{per}(B_{2,1}) \\ \operatorname{per}(B_{1,2}) & \operatorname{per}(B_{2,2}) \end{pmatrix} = \frac{1}{10} \begin{pmatrix} 4 & 2 \\ 3 & 1 \end{pmatrix}$$

Now, let's compute the products:

$$B^*B = \frac{1}{10} \begin{pmatrix} 4 & 2 \\ 3 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = \frac{1}{10} \begin{pmatrix} 10 & 16 \\ 6 & 10 \end{pmatrix} = \begin{pmatrix} 1 & 1.6 \\ 0.6 & 1 \end{pmatrix}$$

$$BB^* = \frac{1}{10} \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \begin{pmatrix} 4 & 2 \\ 3 & 1 \end{pmatrix} = \frac{1}{10} \begin{pmatrix} 10 & 4 \\ 24 & 10 \end{pmatrix} = \begin{pmatrix} 1 & 0.4 \\ 2.4 & 1 \end{pmatrix}$$

As demonstrated, both products B^*B and BB^* satisfy an entrywise inequality with respect to the identity matrix I, but they are not necessarily equal: $B^*B \neq BB^*$ in general. A straightforward consequence of Claim 3.3.2 is that

$$\operatorname{per}(B^*B) \geq 1 \quad \text{and} \quad \operatorname{per}(BB^*) \geq 1,$$

since both B^*B and BB^* dominate I entrywise. However, a sharper inequality is established in Theorem 3.3.3 (see next section), which implies that

$$\operatorname{per}(B^*) \cdot \operatorname{per}(B) \geq 1. \tag{3.7}$$

This is strictly stronger than the two previous inequalities, because

$$\operatorname{per}(B^*B),\, \operatorname{per}(BB^*) \;\geq\; \operatorname{per}(B^*) \cdot \operatorname{per}(B).$$

The final inequality follows from the fact that the permanent is super-multiplicative on non-negative square matrices; that is, for any such matrices C, D, we have $\operatorname{per}(CD) \geq \operatorname{per}(C) \cdot \operatorname{per}(D)$.

3.3.2 An Inequality for the Permanental Inverse

The following inequality describes a relation between the minors of a non-negative matrix and its permanental inverse:

Theorem 3.3.3. *If B^* is the permanental inverse of a non-negative matrix B, then for any index sets S and T, the following inequality holds:*

$$\frac{\operatorname{per}(B(-S,-T))}{\operatorname{per}(B)} \leq \operatorname{per}(B^*(T,S)) \tag{3.8}$$

For context, the equivalent identity for determinants is an equality:

$$\frac{\det(B(-S,-T))}{\det(B)} = \det(B^{-1}(T,S))$$

We now examine some illustrative special cases of this inequality:

- **Case 1:** $S = T = [n]$. Then $B(-S,-T)$ is empty and $B^*(T,S) = B^*$, and the inequality reads:
$$\frac{1}{\operatorname{per}(B)} \;\leq\; \operatorname{per}(B^*) \quad\Longrightarrow\quad \operatorname{per}(B^*) \cdot \operatorname{per}(B) \;\geq\; 1\,.$$
This is the inequality from Equation3.7 before.
- **Case 2:** $S = \{i\}$, $T = \{j\}$. Then $B(-S,-T) = B_{i,j}$ is the $(n-1) \times (n-1)$ matrix obtained by deleting row i and column j, and the inequality becomes:
$$\frac{\operatorname{per}(B_{i,j})}{\operatorname{per}(B)} \;\leq\; \operatorname{per}((B^*)_{j,i}) = (B^*)_{j,i}\,,$$
which holds with equality by definition of the permanental inverse.

These special cases highlight the role of the permanental inverse as a natural upper bound on normalized minors of B. While the determinantal analogue yields exact identities due to multiplicativity, the permanent lacks such algebraic structure.

3.3.3 Schur's Formula for Permanents

In contrast to Schur's determinant formula (which yields an exact equality), the analogous relationship for matrix permanents turns out to be an inequality. We formalize this below.

Theorem 3.1.2 (Permanental Schur's Formula)**.** *Let $A \in \mathbb{R}^{n\times n}_{\geq 0}$ be a block matrix of the form*

$$A = \begin{bmatrix} B & Y \\ X^\top & W \end{bmatrix}, \tag{3.1}$$

where $B \in \mathbb{R}^{d\times d}$ has non-zero permanent. Then the permanent of A satisfies

$$\operatorname{per}(A) \leq \operatorname{per}(B) \cdot \operatorname{per}\left(W + X^\top B^* Y\right). \tag{3.2}$$

Proof strategy. The proof of Theorem 3.1.2 will use induction on k (the number of columns in Y). We first establish two auxiliary results: a formula for the permanent of a rank-1 block update (Observation 3.3.4) and a technical inequality (Lemma 3.3.5) referred to as the *row-uncrossing lemma*. After proving these, we proceed to the inductive step for the general case.

Observation 3.3.4 (Rank-1 Update Formula)**.** For any block matrix of the form $\begin{pmatrix} B & y \\ x^\top & w \end{pmatrix}$, where B is a $d \times d$ matrix, x and y are column vectors of length d, and w is a scalar, the permanent can be expanded as

$$\operatorname{per}\begin{pmatrix} B & y \\ x^\top & w \end{pmatrix} \;=\; \operatorname{per}(B) \,\cdot\, \Big(w \,+\, x^\top B^*\, y\Big).$$

Proof. We let $B^{(i\leftarrow y)}$ denote the $d \times d$ matrix obtained from B by replacing its i-th column with the vector y. Expanding the permanent of the block matrix along its last row gives:

$$\operatorname{per}\begin{pmatrix} B & y \\ x^\top & w \end{pmatrix} \;=\; w \cdot \operatorname{per}(B) \;+\; \sum_{i=1}^{d} x_i \cdot \operatorname{per}\big(B^{(i\leftarrow y)}\big).$$

Here the first term $w \cdot \operatorname{per}(B)$ corresponds to choosing the entry w in the last row, while each summand $x_i \cdot \operatorname{per}(B^{(i\leftarrow y)})$ corresponds to choosing the entry x_i from the last row and then taking all permutations in the remainder of the matrix that involve one element from the inserted column y. In particular, if y_j (the j-th entry of y) is used from that inserted column, it contributes a factor $x_i y_j$ and leaves a $(d-1) \times (d-1)$ submatrix $B_{j,i}$ (obtained by removing the j-th row and i-th column from B) for the rest of the permutation. Summing over all choices of j for each i, we can rewrite the above as

$$\operatorname{per}\begin{pmatrix} B & y \\ x^\top & w \end{pmatrix} \;=\; w \cdot \operatorname{per}(B) \;+\; \sum_{i=1}^{d}\sum_{j=1}^{d} x_i\, y_j \,\operatorname{per}(B_{j,i}).$$

Now, factor $\mathrm{per}(B)$ out of the summation. By the definition of B^*, we have $\frac{\mathrm{per}(B_{j,i})}{\mathrm{per}(B)} = (B^*)_{i,j}$. Thus,

$$\mathrm{per}\begin{pmatrix} B & y \\ x^\top & w \end{pmatrix} \;=\; \mathrm{per}(B)\Big(w + \sum_{i,j=1}^{d} x_i y_j \,\frac{\mathrm{per}(B_{j,i})}{\mathrm{per}(B)}\Big) \;=\; \mathrm{per}(B)\,\Big(w + x^\top B^*\, y\Big),$$

which confirms the formula. □

Lemma 3.3.5 (Row-Uncrossing Inequality)**.** *Let*

$$M = \begin{bmatrix} B & Y \\ X^\top & W \end{bmatrix},$$

where $B \in \mathbb{R}_{\geq 0}^{d\times d},\ X, Y \in \mathbb{R}_{\geq 0}^{d\times k},\ W \in \mathbb{R}_{\geq 0}^{k\times k}$ with $d \geq 0$ and $k \geq 1$. Then, for any fixed $i^* \in [k]$, the following inequality holds:

$$\mathrm{per}(M)\cdot\mathrm{per}(B) \;\leq\; \sum_{j=1}^{k} \mathrm{per}\begin{bmatrix} B & Y_{\cdot,-j} \\ X^\top_{-i^*,\cdot} & W_{i^*,j} \end{bmatrix} \cdot \mathrm{per}\begin{bmatrix} B & y_j \\ x_{i^*}^\top & w_{i^*,j} \end{bmatrix}. \tag{3.9}$$

Here, $Y_{\cdot,-j}$ denotes the matrix Y with its j-th column removed, and $X^\top_{-i^*,\cdot}$ denotes $X^\top$ with its i^*-th row removed (equivalently, removing the i^*-th column of X before transposing). Likewise, $W_{i^*,j}$ is the submatrix of W obtained by deleting the i^*-th row and j-th column.

Proof. The proof proceeds in two main stages:

1. We first establish the inequality in the special case where $W = 0$. This is done by explicitly expanding the permanents and applying an inductive argument on the dimension d.
2. We then extend the result to arbitrary non-negative matrices W. To do this, we define a function representing the difference between the two sides of the inequality. From the first step, we know that this function is non-negative when $W = 0$. We observe that the function is multilinear in the entries of W, and that all its partial derivatives are non-negative. Each partial derivative corresponds to an instance of the same inequality, but for a smaller value of k, allowing us to invoke the inductive hypothesis. These observations imply that the difference function remains non-negative for all non-negative W, thereby completing the proof.

The base cases $d = 0$ or $k = 1$: For the base case when $d = 0$, the inequality in (3.9) is

$$\mathrm{per}(W) \leq \sum_{1\leq j\leq k} \mathrm{per}(W_{i^*,j})\cdot w_{i^*,j}, \tag{3.10}$$

which holds with equality as this is precisely the Laplace expansion of $\mathrm{perm}(W)$ at row i^*. For the base case when $k = 1$, the inequality in (3.9) is

$$\mathrm{per}\begin{bmatrix} B & y \\ x^\top & w \end{bmatrix} \cdot \mathrm{per}(B) \leq \mathrm{per}(B)\cdot \mathrm{per}\begin{bmatrix} B & y \\ x^\top & w \end{bmatrix} \tag{3.11}$$

which is trivially true with equality. So assume that $d \geq 1$ and $k \geq 2$ in any case from here on.

The Special Case $W = 0$: We use induction on d. For $d \geq 1$, observe that when $d < k$, the LHS term is equal to zero when $W = 0$. Because a term in the permanent of M (with $W = 0$) is non-zero only if it selects $d-k$ elements from B and k elements from each of X and Y. Assume that $d \geq k$ and let $\mathcal{F}$ be the set of one-one functions mapping from $[k]$ to $[d]$. Expanding the LHS of (3.9) gives

$$\operatorname{per}\begin{bmatrix} B & Y \\ X^\top & 0 \end{bmatrix} \cdot \operatorname{per}(B) = \operatorname{per}(B) \cdot \sum_{f,g \in \mathcal{F}} \prod_{i \in [k]} x_{f(i),i} \cdot \prod_{j \in [k]} y_{g(j),j} \cdot \operatorname{per} B(-\text{img}_g, -\text{img}_f). \tag{3.12}$$

In order to similarly expand the RHS, define $\mathcal{F}_t$ for $t \in [k]$ as the set of functions mapping $[k]$ to $[d]$ such that $f \in \mathcal{F}_t$ is one-one when restricted to $[k]\backslash\{t\}$. Expanding the RHS of (3.9) gives

$$\sum_{1 \leq t \leq k} \operatorname{per}\begin{bmatrix} B & Y_{\cdot,-t} \\ X^\top_{-i^*,\cdot} & 0 \end{bmatrix} \cdot \operatorname{per}\begin{bmatrix} B & y_t \\ x_{i^*}^\top & 0 \end{bmatrix} = \tag{3.13}$$

$$\sum_{1 \leq t \leq k} \sum_{f' \in \mathcal{F}_{i^*}, g' \in \mathcal{F}_t} \prod_{i \in [k]} x_{f'(i),i} \cdot \prod_{j \in [k]} y_{g'(j),j} \operatorname{per}[B(-\text{img}_{g',[k]-t}, -\text{img}_{f',[k]-i^*})] \cdot \operatorname{per}(B_{g'(t),f'(i^*)}). \tag{3.14}$$

Since $\mathcal{F} \subseteq \mathcal{F}_t$ for any t, we can obtain a lower bound by only summing over $f', g' \in \mathcal{F}$. By exchanging the summations after this step gives

$$\geq \sum_{f',g' \in \mathcal{F}} \prod_{i \in [k]} x_{f'(i),i} \cdot \prod_{j \in [k]} y_{g'(j),j} \sum_{1 \leq t \leq k} \operatorname{per}[B(-\text{img}_{g',[k]-t}, -\text{img}_{f',[k]-i^*})] \cdot \operatorname{per}(B_{g'(t),f'(i^*)}). \tag{3.15}$$

It is sufficient to show that

$$\operatorname{per}(B) \cdot \operatorname{per} B(-\text{img}_g, -\text{img}_f) \leq \sum_{1 \leq t \leq k} \operatorname{per}[B(-\text{img}_{g,[k]-t}, -\text{img}_{f,[k]-i^*})] \cdot \operatorname{per}(B_{g(t),f(i^*)}) \tag{3.16}$$

for any $f, g \in \mathcal{F}$, $i^* \in [k]$. Equation (3.16) is in the form of (3.9) with the substitution

$$B, X, Y, i^*, d, k \leftarrow B(-\text{img}_g, -\text{img}_f)^\top, B(-\text{img}_g, \text{img}_f), B(\text{img}_g, -\text{img}_f)^\top, f(i^*), d-k, k. \tag{3.17}$$

We can conclude this case using inductive hypothesis.

Extension to $W \geq 0$: It remains to prove (3.9) for general W given that we have a proof for the case when $W = 0$. The first step is to observe that both the LHS and RHS of (3.9) are multi-linear functions with respect to the $w_{i,j}$ variables. For fixed B, X, Y, i^*, consider the function $h(W) := \sum_{1 \leq j \leq k} \operatorname{per}\begin{bmatrix} B & Y_{\cdot,-j} \\ X^\top_{-i^*,\cdot} & W_{i^*,j} \end{bmatrix} \cdot \operatorname{per}\begin{bmatrix} B & y_j \\ x_{i^*}^\top & w_{i^*,j} \end{bmatrix} - \operatorname{per}\begin{bmatrix} B & Y \\ X^\top & W \end{bmatrix} \cdot \operatorname{per}(B)$. Since we know

that $h(0) \geq 0$, it suffices to show that $\frac{\partial h(W)}{\partial w_{\alpha,\beta}} \geq 0$ for every $\alpha, \beta \in [k]$. Then that would imply $h(W) \geq h(0) \geq 0$.

We proceed by induction on k. For $\alpha, \beta \in [k]$, $\alpha \neq i^*$, the derivative $\frac{\partial h(W)}{\partial w_{\alpha,\beta}}$ is equal to

$$\sum_{j \in [k] \setminus \{\beta\}} \operatorname{per} \begin{bmatrix} B & Y([d], -\{j, \beta\}) \\ X([d], -\{i^*, \alpha\})^\top & W(-\{i^*, \alpha\}, -\{j, \beta\}) \end{bmatrix} \cdot \operatorname{per} \begin{bmatrix} B & y_j \\ x_{i^*}^\top & w_{i^*,j} \end{bmatrix} - \operatorname{per} \begin{bmatrix} B & Y_{\cdot,-\beta} \\ X_{-\alpha,\cdot}^\top & W_{\alpha,\beta} \end{bmatrix} \cdot \operatorname{per}(B) \tag{3.18}$$

which is non-negative using inductive hypothesis. The substitution being

$$B, X, Y, i^*, d, k \leftarrow B, X_{\cdot,-\alpha}, Y_{\cdot,-\beta}, i^*, d, k-1. \tag{3.19}$$

For $\alpha = i^*, \beta \in [k]$, the derivative $\frac{\partial h(W)}{\partial w_{i^*,\beta}}$ is equal to

$$\operatorname{per} \begin{bmatrix} B & Y_{\cdot,-\beta} \\ X_{-i^*,\cdot}^\top & W_{i^*,\beta} \end{bmatrix} \cdot \operatorname{per}(B) - \operatorname{per} \begin{bmatrix} B & Y_{\cdot,-\beta} \\ X_{-i^*,\cdot}^\top & W_{i^*,\beta} \end{bmatrix} \cdot \operatorname{per}(B) = 0. \tag{3.20}$$

This finishes the proof of the lemma. □

Proof of Theorem 3.1.2. We prove by induction on k. Equation (3.2) holds with equality for $k = 1$ using Observation 3.3.4. For $k \geq 2$,

using Lemma 3.3.5, we have

$$\operatorname{per} \begin{bmatrix} B & Y \\ X^\top & W \end{bmatrix} \cdot \operatorname{per}(B) \leq \sum_{1 \leq j \leq k} \operatorname{per} \begin{bmatrix} B & Y_{\cdot,-j} \\ X_{-1,\cdot}^\top & W_{1,j} \end{bmatrix} \cdot \operatorname{per} \begin{bmatrix} B & y_j \\ x_1^\top & w_{1,j} \end{bmatrix}. \tag{3.21}$$

Using inductive hypothesis, we have

$$\operatorname{per} \begin{bmatrix} B & Y_{\cdot,-j} \\ X_{-1,\cdot}^\top & W_{1,j} \end{bmatrix} \leq \operatorname{per}(B) \cdot \operatorname{per}(W_{1,j} + X_{-1,\cdot}^\top B^* Y_{\cdot,-j}). \tag{3.22}$$

substituting this gives

$$\operatorname{per} \begin{bmatrix} B & Y \\ X^\top & W \end{bmatrix} \cdot \operatorname{per}(B) \leq \operatorname{per}(B) \sum_{1 \leq j \leq k} \operatorname{per}(W_{1,j} + X_{-1,\cdot}^\top B^* Y_{\cdot,-j}) \cdot \operatorname{per} \begin{bmatrix} B & y_j \\ x_1^\top & w_{1,j}. \end{bmatrix} \tag{3.23}$$

Using Observation 3.3.4 gives

$$\leq \operatorname{per}(B)^2 \sum_{1 \leq j \leq k} \operatorname{per}(W_{1,j} + X_{-1,\cdot}^\top B^* Y_{\cdot,j}) \cdot (w_{1,j} + x_1^\top B^* y_j) \tag{3.24}$$

$$= \operatorname{per}(B)^2 \cdot \operatorname{per}(W + X^\top B^* Y). \tag{3.25}$$

concluding the proof that

$$\operatorname{per} \begin{bmatrix} B & Y \\ X^\top & W \end{bmatrix} \leq \operatorname{per}(B) \cdot \operatorname{per}(W + X^\top B^* Y). \tag{3.26}$$

□

Two permanent inequalities

The permanent version of Lemma 3.5.1 comes with an inequality.

Lemma 3.3.6. *The following inequality holds true:*

$$\operatorname{per}\begin{bmatrix} B & y_1 & y_2 \\ x_1^\top & w_{1,1} & w_{1,2} \\ x_2^\top & w_{2,1} & w_{2,2} \end{bmatrix} \cdot \operatorname{per}(B) \leq \operatorname{per}\begin{bmatrix} B & y_1 \\ x_1^\top & w_{1,1} \end{bmatrix} \cdot \operatorname{per}\begin{bmatrix} B & y_2 \\ x_2^\top & w_{2,2} \end{bmatrix} + \operatorname{per}\begin{bmatrix} B & y_2 \\ x_1^\top & w_{1,2} \end{bmatrix} \cdot \operatorname{per}\begin{bmatrix} B & y_1 \\ x_2^\top & w_{2,1} \end{bmatrix}. \tag{3.27}$$

For any matrix $B \in \mathbb{R}_{\geq 0}^{d\times d}$, vectors $x_i, y_i \in \mathbb{R}_{\geq 0}^{d\times 1}$, and scalars $w_{i,j} \geq 0$ with $i, j \in \{1, 2\}$.

Lemma 3.3.7. *For a matrix $W \in \mathbb{R}_{\geq 0}^{k\times k}$, vectors $x, y \in \mathbb{R}_{\geq 0}^{k}$, and scalar $b \neq 0$, let $C \in \mathbb{R}_{\geq 0}^{k\times k}$ be the matrix defined by $c_{i,j} := \operatorname{per}\begin{bmatrix} b & y_j \\ x_i & w_{i,j} \end{bmatrix} \cdot b^{-1} = w_{i,j} + x_i y_j / b$. The following inequality holds true:*

$$\operatorname{per}\begin{bmatrix} b & y^\top \\ x & W \end{bmatrix} \cdot b^{-1} \leq \operatorname{per}(C) \tag{3.28}$$

Both the above lemmas are special cases of Theorem 3.1.2.

3.3.4 The Permanent Process

Consider a process that is a modified version of the Gaussian elimination process. The state of the matrix entries after the end of step t for some $1 \leq t \leq n-1$ is given by

$$a_{i,j}^{(t+1)} = \begin{cases} a_{i,j}^{(t)} + \frac{a_{i,t}^{(t)} a_{t,j}^{(t)}}{a_{t,t}^{(t)}}, & \text{for } j \geq t+1 \\ a_{i,j}^{(t)}, & \text{Otherwise.} \end{cases} \tag{3.29}$$

It is not easy to find a clean closed form solution for the entries of $A^{(t)}$ with respect to the entries of A during the permanent process like we had for the Gaussian process in Theorem 3.2.2. However, something weaker can be proven that is strong enough to extend Corollary 3.2.3.

Theorem 3.3.8. *The diagonal entries of the matrix $A^{(n)}$ can be lower bounded by*

$$a_{t,t}^{(n)} = a_{t,t}^{(t)} \geq \frac{\operatorname{per}(A^{(t)}(-[t-1], -[t-1]))}{\operatorname{per}(A^{(t+1)}(-[t], -[t]))}. \tag{3.30}$$

for $1 \leq t \leq n$.

Proof. Observe that it is sufficient to prove just the case for $t = 1$ because, the second step of the permanent process essentially applies the first step of the permanent process on the sub-matrix $A^{(2)}(-\{1\}, -\{1\})$. What we want to show is that

$$\operatorname{per}(A^{(2)}(-\{1\}, -\{1\})) \geq \frac{\operatorname{per}(A)}{a_{1,1}}. \tag{3.31}$$

This follows directly from Lemma 3.3.7. □

Corollary 3.3.9. *The permanent of A is upper bounded by the product of the diagonal entries of $A^{(n)}$.*

$$\operatorname{per}(A) \leq \prod_{1 \leq i \leq n} a_{i,i}^{(i)} = \prod_{1 \leq i \leq n} a_{i,i}^{(n)}. \tag{3.32}$$

Proof. Multiplying the lower bounds for $a_{i,i}^{(n)}$ from Theorem 3.3.8 gives the required lower bound. □

Corollary 3.3.9 provides an algorithmic upper bound for the permanent of any non-negative matrix. In fact, any theoretical upper bound to the product of the diagonal entries of A after n steps of the permanent process to A can be used as an upper bound for $\operatorname{per}(A)$.

3.3.5 Recursive Upper Bounds

Expanding the recursive definition of the permanent process from (3.29), we obtain:

$$a_{i,j}^{(t)} = a_{i,j}^{(1)} + \sum_{1 \leq s < \min(t,j)} \frac{a_{i,s}^{(s)} a_{s,j}^{(s)}}{a_{s,s}^{(s)}}, \quad \forall t. \tag{3.33}$$

This recurrence suggests focusing on entries of the form $a_{i,j}^{(\min(i,j))}$, since they can be expressed in terms of similar entries with smaller indices. Substituting $t = \min(i,j)$ in (3.33) yields:

$$a_{i,j}^{(\min(i,j))} = a_{i,j}^{(1)} + \sum_{1 \leq s < \min(i,j)} \frac{a_{i,s}^{(s)} a_{s,j}^{(s)}}{a_{s,s}^{(s)}}. \tag{3.34}$$

Observe that in each term $a_{i,s}^{(s)}$ and $a_{s,j}^{(s)}$, the index s satisfies $s = \min(i,s) = \min(s,j)$ since $s < \min(i,j)$.

Define the shorthand:

$$u_{i,j} := a_{i,j}^{(\min(i,j))}.$$

Substituting this into (3.34) gives the recurrence:

$$u_{i,j} = a_{i,j} + \sum_{1 \leq s < \min(i,j)} \frac{u_{i,s} u_{s,j}}{u_{s,s}}. \tag{3.35}$$

Theorem 3.3.10. *Let $A \in \mathbb{R}_{\geq 0}^{n \times n}$ be a non-negative matrix. If a matrix $B \in \mathbb{R}_{\geq 0}^{n \times n}$ satisfies:*

$$a_{i,j} + \sum_{1 \leq s < \min(i,j)} \frac{b_{i,s} b_{s,j}}{a_{s,s}} \leq b_{i,j}, \tag{3.36}$$

then $u_{i,j} \leq b_{i,j}$ for all $i, j \in [n]$. Moreover, this conclusion remains valid even if the inequality in (3.36) *holds with equality.*

Proof. We prove the claim by induction on $t = \min(i, j)$.

Base case: If $t = 1$, then $u_{i,j} = a_{i,j} \leq b_{i,j}$ directly from the assumption.

Inductive step: Suppose the claim holds for all pairs (i, j) with $\min(i, j) < t$. Consider $\min(i, j) = t \geq 2$. Using the recurrence in (3.35), we have:

$$u_{i,j} = a_{i,j} + \sum_{1 \leq s < t} \frac{u_{i,s} u_{s,j}}{u_{s,s}} \tag{3.37}$$

$$\leq a_{i,j} + \sum_{1 \leq s < t} \frac{u_{i,s} u_{s,j}}{a_{s,s}} \tag{3.38}$$

$$\leq a_{i,j} + \sum_{1 \leq s < t} \frac{b_{i,s} b_{s,j}}{a_{s,s}} \leq b_{i,j}, \tag{3.39}$$

where the second inequality uses the inductive hypothesis $u_{i,s}, u_{s,j} \leq b_{i,s}, b_{s,j}$, and the last step uses the assumption in (3.36).

The same argument applies if the inequality in (3.36) holds with equality. Thus, the result holds in both the inequality and equality cases. □

Corollary 3.3.11. *Let $A \in \mathbb{R}_{\geq 0}^{n \times n}$, and let $B \in \mathbb{R}_{\geq 0}^{n \times n}$ satisfy* (3.36)*. Then:*

$$\operatorname{per}(A) \leq \prod_{i=1}^{n} b_{i,i}.$$

Proof. From Theorem 3.3.10, we have $u_{i,i} \leq b_{i,i}$ for all i. By Theorem 3.3.8, we know that $\operatorname{per}(A) \leq \prod_{i=1}^{n} u_{i,i}$. Combining both inequalities yields the desired bound. □

3.3.6 Theoretical Upper Bounds for Structured Matrices

We now illustrate how the recursive upper bound framework can be used to derive explicit upper bounds for permanents of structured matrices.

Recall from Theorem 3.3.10 that if two non-negative matrices $A, B \in \mathbb{R}_{\geq 0}^{n \times n}$ satisfy:

$$a_{i,j} + \sum_{1 \leq s < \min(i,j)} \frac{b_{i,s} b_{s,j}}{a_{s,s}} = b_{i,j} \quad \text{for all } i, j \in [n], \tag{3.40}$$

then:

$$\operatorname{per}(A) \leq \prod_{i=1}^{n} b_{i,i}.$$

This identity suggests a simple approach: to upper bound $\operatorname{per}(A)$, we can construct a matrix B satisfying (3.40) and then bound the entries of B in terms of those of A.

Lemma 3.3.12. *Let $A \in \mathbb{R}^{n\times n}_{\geq 0}$ be a non-negative matrix satisfying:*

$$\frac{(1+\varepsilon)^2}{\varepsilon} \sum_{s=1}^{\min(i,j)} \frac{a_{i,s}a_{s,j}}{a_{s,s}} \leq a_{i,j} \tag{3.41}$$

for some $\varepsilon > 0$. Then:

$$\operatorname{per}(A) \leq (1+\varepsilon)^n \cdot \prod_{i=1}^{n} a_{i,i}.$$

Proof. We aim to show that the matrix B defined via (3.40) satisfies $b_{i,j} \leq (1+\varepsilon)a_{i,j}$. The result will then follow from Corollary 3.3.11.

We proceed by induction on $\min(i,j)$. The base case $\min(i,j) = 1$ is immediate since $b_{i,j} = a_{i,j}$ in this case.

For $i, j \geq 2$, using the definition (3.40) and the inductive hypothesis, we have:

$$b_{i,j} = a_{i,j} + \sum_{1\leq s<\min(i,j)} \frac{b_{i,s}b_{s,j}}{a_{s,s}} \tag{3.42}$$

$$\leq a_{i,j} + (1+\varepsilon)^2 \sum_{1\leq s<\min(i,j)} \frac{a_{i,s}a_{s,j}}{a_{s,s}} \tag{3.43}$$

$$\leq a_{i,j} + \varepsilon a_{i,j} = (1+\varepsilon)a_{i,j}, \tag{3.44}$$

where the second step uses the inductive assumption $b_{i,s}, b_{s,j} \leq (1+\varepsilon)a_{i,s}, a_{s,j}$, and the third uses the assumption in the lemma.

This completes the inductive step and hence the proof. □

3.4 Future Directions

In this work, we have extended classical determinantal concepts—such as the matrix inverse, the Schur's formula (which underlies or generalizes many determinantal identities), and the Gaussian elimination process—to the setting of permanents of non-negative matrices. These analogues form a new and mathematically intriguing framework for reasoning about permanents, rooted in structured, algebraic identities rather than combinatorial or probabilistic heuristics.

Beyond their intrinsic interest, we use these tools to derive a deterministically computable upper bound on the permanent. While this bound can be loose on certain matrix families, we show that it performs well when the matrix satisfies a form of diagonal dominance. This opens several promising directions for future research:

- **Characterizing Favorable Matrix Classes:** Studying the behavior of the Permanent Process on specific, structured matrix families—such as stochastic, Toeplitz, or adjacency matrices of certain graph classes—to identify where the bound is tightest.

- **Hybrid Approaches:** Combining the deterministic, algebraic framework presented here with established probabilistic or combinatorial techniques to create new, more powerful hybrid approximation algorithms.
- **Improved Upper Bounds:** There may be potential to sharpen or specialize the upper bounds derived here for certain matrix types, especially by refining the permanental Schur's formula machinery.
- **Empirical Validation:** Implementing and benchmarking the permanent process across diverse matrix types could offer insight into its practical viability and guide further theoretical improvements.

3.5 Appendix for Chapter 3

Proof of Theorem 3.2.2. We prove using induction on t. The base case $t = 1$ is trivial. For $t > 1$, it is sufficient to prove the theorem for $j \geq t$ because for smaller j, the entry is determined at a smaller time step. The inductive step is to show that

$$\frac{\det_A([t-1]+\{i\},[t-1]+\{j\})}{\det_A([t-1],[t-1])} = \frac{\det_A([t-2]+\{i\},[t-2]+\{j\})}{\det_A([t-2],[t-2])} \tag{3.45}$$

$$- \frac{\frac{\det_A([t-2]+\{i\},[t-1])}{\det_A([t-2],[t-2])} \cdot \frac{\det_A([t-1],[t-2]+\{j\})}{\det_A([t-2],[t-2])}}{\frac{\det_A([t-1],[t-1])}{\det_A([t-2],[t-2])}}. \tag{3.46}$$

Simplifying gives

$$\det_A([t-1]+\{i\},[t-1]+\{j\}) \cdot \det_A([t-2],[t-2]) = \det_A([t-2]+\{i\},[t-2]+\{j\}) \cdot \det_A([t-1],[t-1]) \tag{3.47}$$

$$- \det_A([t-2]+\{i\},[t-1]) \cdot \det_A([t-1],[t-2]+\{j\}). \tag{3.48}$$

This is exactly the identity in Lemma 3.5.1 with

$$B = A([t-2],[t-2]);\ y_1 = A([t-2],\{t-1\}),\ y_2 = A([t-2],\{j\});$$
$$x_1^\top = A(\{t-1\},[t-2]),\ x_2^\top = A(\{i\},[t-2]);\ w = A(\{t-1,i\},\{t-1,j\}).$$

□

Proof of Corollary 3.2.3. Using Theorem 3.2.2, we have $a_{i,i}^{(n)} = \det_A([i],[i]) / \det_A([i-1],[i-1])$. Substituting this gives $\prod_{1 \leq i \leq n} a_{i,i}^{(n)} = \det_A([n],[n]) = \det(A)$. □

Lemma 3.5.1. *The following equality holds true:*

$$\begin{vmatrix} B & y_1 & y_2 \\ x_1^\top & w_{1,1} & w_{1,2} \\ x_2^\top & w_{2,1} & w_{2,2} \end{vmatrix} \cdot |B| = \begin{vmatrix} B & y_1 \\ x_1^\top & w_{1,1} \end{vmatrix} \cdot \begin{vmatrix} B & y_2 \\ x_2^\top & w_{2,2} \end{vmatrix} - \begin{vmatrix} B & y_2 \\ x_1^\top & w_{1,2} \end{vmatrix} \cdot \begin{vmatrix} B & y_1 \\ x_2^\top & w_{2,1} \end{vmatrix} \tag{3.49}$$

For any matrix $B \in \mathbb{R}^{d \times d}$, vectors $x_i, y_i \in \mathbb{R}^{d \times 1}$, and scalars $w_{i,j}$ with $i, j \in \{1, 2\}$.

Proof. Assume that $|B| \neq 0$. Using Schur's formula (see Theorem 3.2.4), we have

$$\begin{vmatrix} B & y_1 & y_2 \\ x_1^\top & w_{1,1} & w_{1,2} \\ x_2^\top & w_{2,1} & w_{2,2} \end{vmatrix} = |B| \cdot \det \begin{bmatrix} w_{1,1} - x_1^\top B^{-1} y_1 & w_{1,2} - x_1^\top B^{-1} y_2 \\ \\ w_{2,1} - x_2^\top B^{-1} y_1 & w_{2,2} - x_2^\top B^{-1} y_2 \end{bmatrix} \tag{3.50}$$

and $\begin{vmatrix} B & y_j \\ x_i^\top & w_{i,j} \end{vmatrix} = |B| \cdot (w_{i,j} - x_i^\top B^{-1} y_j)$ for $i, j \in \{1, 2\}$. Substituting these and factoring out $|B|^2$ from both LHS and RHS of Equation (3.49), gives

$$\det \begin{bmatrix} w_{1,1} - x_1^\top B^{-1} y_1 & w_{1,2} - x_1^\top B^{-1} y_2 \\ \\ w_{2,1} - x_2^\top B^{-1} y_1 & w_{2,2} - x_2^\top B^{-1} y_2 \end{bmatrix} = (w_{1,1} - x_1^\top B^{-1} y_1)(w_{2,2} - x_2^\top B^{-1} y_2) \tag{3.51}$$

$$- (w_{1,2} - x_1^\top B^{-1} y_2)(w_{2,1} - x_2^\top B^{-1} y_1) \tag{3.52}$$

which is true. The identity in Equation (3.49) should hold true even when $|B| = 0$ using continuity of the determinants with respect to the entries of the matrix. □

Part II

Fairness

Chapter 4

Nash Social Welfare Maximization

4.1 Introduction

Fair and efficient division of resources among agents is a fundamental problem arising in various fields [19, 31, 32, 160, 161, 190]. In the most general setup, we are given a set of m indivisible items $\mathcal{G}$, and a set of n agents, $\mathcal{A}$. Each agent has a valuation function $\mathbf{v}_i : 2^{\mathcal{G}} \to \mathbb{R}_{\geq 0}$ on subsets of the items, so that if agent $i \in \mathcal{A}$ were assigned all the items in $S \subseteq \mathcal{G}$, the value they receive would be $\mathbf{v}_i(S)$. The goal is to find an assignment of items to players, $\sigma : \mathcal{G} \to \mathcal{A}$, to maximize some function of the agents' valuations.

While there are many social welfare functions which can be used to evaluate the efficacy of an assignment of goods to the agents, the Nash Social Welfare function is well-known to interpolate between fairness and overall utility. It asks that we maximize the geometric mean of the agents' valuations:

$$\prod_{i \in \mathcal{A}} \mathbf{v}_i(\sigma^{-1}(i)),$$

where $\sigma^{-1}(i) = \{j \in \mathcal{G} : \sigma(j) = i\}$ is the set of items assigned to agent i. The unweighted Nash Social Welfare function first appeared as the solution to an arbitration scheme proposed by Nash for two-person bargaining games and was later generalized to multiple players [115, 144]. Since then, it has been widely used in numerous fields to model resource allocation problems. An attractive feature of the objective is that it is invariant under scaling by any of the agent's valuations, and therefore, each agent can specify its valuation in its own units (see [43] for a detailed treatment). While the theory of Nash Social Welfare objective was initially developed for divisible items, more recently, it has been applied in the context of indivisible items. We refer the reader to [42] for a comprehensive overview of the problem in the latter setting. Indeed, optimizing the Nash Social Welfare objective also implies notions of fairness, such as *envy-free* allocation in an approximate sense [20, 42].

In an instance of the weighted Nash Social Welfare problem, every agent $i \in \mathcal{A}$ has a weight $w_i \geq 0$ such that $\sum_{i \in \mathcal{A}} w_i = 1$. The goal is to find an assignment of items, $\sigma : \mathcal{G} \to \mathcal{A}$, to

maximize the following welfare function:

$$\prod_{i \in \mathcal{A}} \left(\mathbf{v}_i(\sigma^{-1}(i))\right)^{w_i}. \tag{4.1}$$

In this work, we will consider the case of additive valuations, for which there are numbers $v_{ij} \geq 0$ for each $i \in \mathcal{A}$ and $j \in \mathcal{G}$ such that $\mathbf{v}_i(S) = \sum_{j \in S} v_{ij}$. For ease of notation, we will work with the log objective and denote

$$\mathrm{NSW}(\sigma) = \sum_{i \in \mathcal{A}} w_i \log \left(\sum_{j \in \sigma^{-1}(i)} v_{ij} \right). \tag{4.2}$$

Let $\mathrm{OPT} = \max_{\sigma: \mathcal{G} \to \mathcal{A}} \mathrm{NSW}(\sigma)$ denote the optimal log objective. The case where $w_i = \frac{1}{n}$ for each $i \in \mathcal{A}$ is the much-studied "symmetric" or unweighted Nash Social Welfare problem.

The Nash Social Welfare function with weights (also referred to as asymmetric or non-symmetric Nash Social Welfare) was first studied in the seventies [100, 112] in the context of two-person bargaining games. For example, in the bargaining context, it allows different agents to have different weights. Due to this flexibility, problems in many diverse domains can be modeled using the weighted objective, including bargaining theory [43, 123], water resource allocation [82, 106], and climate agreements [191]. In the context of indivisible goods, the study of this problem has been much more recent [85, 86, 87]. There have also been attempts to extend the ideas of envy-freeness to the weighted setting [44, 180], where the situation is more complicated than in the unweighted setting. There are multiple possible generalizations, and maximizing the weighted Nash Social Welfare does not always guarantee the same envy-freeness conditions. In this work, we aim to shed light on the weighted Nash Social Welfare problem, mainly focusing on mathematical programming relaxations for the problem.

4.1.1 Preliminaries and Notation

In order to state the key results of this work, we need the following preliminaries and related notation.

KL-Divergence. For two probability distributions $\mathbf{p}, \mathbf{q}$ over the same discrete domain $\mathcal{X}$, the KL-divergence between $\mathbf{p}$ and $\mathbf{q}$ is defined as

$$D_{\mathrm{KL}}(\mathbf{p} \,\|\, \mathbf{q}) = \sum_{x \in \mathcal{X}} p(x) \log \left(\frac{p(x)}{q(x)} \right).$$

It is well-known, via Gibbs's inequality, that the KL-divergence between two distributions is non-negative and is zero if and only if $\mathbf{p}$ and $\mathbf{q}$ are identical.

Moreover, if $\mathbf{u}$ is the uniform distribution on $\mathcal{X}$ and $\mathbf{p}$ is an arbitrary distribution on the same domain, then

$$D_{\mathrm{KL}}(\mathbf{p} \,\|\, \mathbf{u}) = \log |\mathcal{X}| - \sum_{x \in \mathcal{X}} p(x) \log \frac{1}{p(x)}.$$

Feasibility Polytope. Consider a complete bipartite graph $G = (\mathcal{G} \cup \mathcal{A}, E)$ where E contains an edge (i, j) for each $i \in \mathcal{A}$ and $j \in \mathcal{G}$. Let $\mathcal{M}(\mathcal{A})$ denote the set of all matchings in G of size $|\mathcal{A}|$, i.e., matchings which have an edge incident to every vertex in $\mathcal{A}$. The convex hull of $\mathcal{M}(\mathcal{A})$, denoted by $\mathcal{P}(\mathcal{A}, \mathcal{G})$, is defined by the following polytope.

Definition 4.1.1 (Feasibility Polytope)**.** For a set of m indivisible items, $\mathcal{G}$, and a set of n agents, $\mathcal{A}$, the feasibility polytope, denoted by $\mathcal{P}(\mathcal{A}, \mathcal{G})$, is defined as

$$\mathcal{P}(\mathcal{A}, \mathcal{G}) := \left\{ \mathbf{b} \in \mathbb{R}_{\geq 0}^{|\mathcal{A}| \times |\mathcal{G}|} \; : \; \sum_{j \in \mathcal{G}} b_{ij} = 1 \; \forall i \in \mathcal{A} \; , \sum_{i \in \mathcal{A}} b_{ij} \leq 1 \; \forall j \in \mathcal{G} \right\}.$$

The constraint $\sum_{j \in \mathcal{G}} b_{ij} = 1$ is called the Agent constraint for agent i, and the constraint $\sum_{i \in \mathcal{A}} b_{ij} \leq 1$ is referred to as the Item constraint for item j.

4.1.2 Our Results and Contributions

Our main contributions are (1) to show equivalence between different mathematical relaxations for the Nash Social Welfare problem (see Appendix 4.6.2), (2) generalize these formulations to give new mathematical formulations for the weighted NSW and finally (3) use the equivalence and generalize the algorithms for symmetric case to obtain improved approximation algorithms for weighted NSW.

Equivalence of Convex Programs

Our first result relates two previous convex programming relaxations for the unweighted Nash Social Welfare problem presented in [61] and [7].

Building on the algorithm of [60], [61] introduced the following relaxation for the unweighted Nash Social Welfare problem.

$$\begin{aligned}
\max_{b} \quad & \frac{1}{n} \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{G}} b_{ij} \log (v_{ij}) - \frac{1}{n} \sum_{j \in \mathcal{G}} \left(\sum_{i \in \mathcal{A}} b_{ij} \right) \log \left(\sum_{i \in \mathcal{A}} b_{ij} \right) && \text{(CVX-Unweighted)} \\
\text{s.t.} \quad & \sum_{j} b_{ij} = 1 \quad \forall i \in \mathcal{A} \\
& \sum_{i} b_{ij} \leq 1 \quad \forall j \in \mathcal{G} \\
& b_{ij} \geq 0 \quad \forall (i, j) \in \mathcal{A} \times \mathcal{G}.
\end{aligned}$$

They showed that (CVX-Unweighted) is a convex relaxation of the Nash Social Welfare objective, and the prices used by the algorithm presented in [60] can be obtained as dual variables of (CVX-Unweighted). Interestingly, the convex relaxation is not in terms of the assignment variables. Indeed, given an optimal assignment $\sigma : \mathcal{G} \to \mathcal{A}$, the corresponding solution to

(CVX-Unweighted) will set the variables b_{ij} as follows:

$$b_{ij} = \begin{cases} \frac{v_{ij}}{\sum_{k \in \sigma^{-1}(i)} v_{ik}} & \text{if } \sigma(j) = i \\ 0 & \text{otherwise.} \end{cases} \tag{4.3}$$

One can verify that $\mathbf{b}$ satisfies all the constraints in (CVX-Unweighted), and its objective value is equal to the logarithm of the geometric mean of the valuations.

A different convex programming relaxation, (LogConcave-Unweighted), for unweighted NSW was presented by Anari et al. [7]. They showed that the objective of (LogConcave-Unweighted) is a log-concave function in x and convex in $\log y$ and used inequalities about stable polynomials to give an e-approximation for unweighted NSW.

$$\begin{aligned} \max_{\mathbf{x} \geq \mathbf{0}} \inf_{\mathbf{y} > \mathbf{0}} \quad & \frac{1}{n} \sum_{i \in \mathcal{A}} \log \left(\sum_{j \in \mathcal{G}} x_{ij}\, v_{ij}\, y_j \right) \\ \text{s.t.} \quad & \sum_{i \in \mathcal{A}} x_{ij} = 1 \quad \forall j \in \mathcal{G} \\ & \prod_{j \in S} y_j \geq 1 \quad \forall S \in \binom{\mathcal{G}}{n}. \end{aligned} \tag{LogConcave-Unweighted}$$

Here, $\binom{\mathcal{G}}{n}$ denotes the collection of subsets of $\mathcal{G}$ of size n, where $n = |\mathcal{A}|$.

On the surface, (LogConcave-Unweighted) and (CVX-Unweighted), and their corresponding rounding algorithms are quite different: [61] uses intuition from economics and market equilibrium to both arrive at (CVX-Unweighted) and also to round it, while [7] uses the properties of log-concave polynomials to round (LogConcave-Unweighted). However, our next result shows that these two convex programs indeed optimize the same objective.

Theorem 4.1.2. *For any instance $(\mathcal{A}, \mathcal{G}, \mathbf{v})$ of unweighted Nash Social Welfare, the optimal values of* (LogConcave-Unweighted) *and* (CVX-Unweighted) *are the same.*

It is easy to transfer an optimal solution of one program to a solution to the other by setting

$$b_{ij} = \frac{x_{ij} v_{ij}}{\sum_{j'} x_{ij'} v_{ij'}} \quad \text{or} \quad x_{ij} = \frac{b_{ij}}{\sum_{i'} b_{ij}},$$

but verifying that the objective value remains the same uses the optimality conditions. These transformations do not necessarily preserve objective value for non-optimal fractional solutions. In Appendix 4.6.2, we more systematically show how one program could be derived from the other, by using a sequence of variable changes and convex duality to show that the programs are equivalent.

New Relaxations in the Weighted Case

Besides providing a novel connection between two very different approaches to the unweighted problem, Theorem 4.1.2 is also vital to derive our main algorithm for weighted Nash Social Welfare. Independently generalizing either of these approaches to the weighted case is challenging:

[60, 61] use intuition from economics to arrive at (CVX-Unweighted), and these concepts do not generalize to the weighted case. On the other hand, there is a natural convex generalization of (LogConcave-Unweighted) for the weighted case which is still log-concave, but the objective is no longer stable, and therefore the machinery introduced in [7] cannot be used to analyze it.

Our second contribution is to propose new relaxations for the weighted version of the Nash Social Welfare problem. By replacing the uniform weights in (CVX-Unweighted) with the weights w_i we get the first generalization.

$$\begin{aligned}
\max_{\mathbf{b}} \quad & \sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{G}} w_i\, b_{ij} \log v_{ij} - \sum_{j\in\mathcal{G}}\sum_{i\in\mathcal{A}} w_i\, b_{ij} \log\left(\sum_{i'\in\mathcal{A}} b_{i'j}\right) && \text{(NCVX-Weighted)}\\
\text{s.t.} \quad & \sum_{j\in\mathcal{G}} b_{ij} = 1 \quad \forall i \in \mathcal{A}\\
& \sum_{i\in\mathcal{A}} b_{ij} \le 1 \quad \forall j \in \mathcal{G}\\
& b_{ij} \ge 0 \quad \forall (i,j) \in \mathcal{A}\times\mathcal{G}
\end{aligned}$$

Unfortunately, the objective function to this program is not convex when the weights are not uniform. As an alternative, we can also generalize (LogConcave-Unweighted) as follows

$$\begin{aligned}
\max_{\mathbf{x}\ge 0}\min_{\mathbf{y}>0} \quad & \sum_{i\in\mathcal{A}} w_i \log\left(\sum_{j\in\mathcal{G}} x_{ij}\; v_{ij}\; y_j^{1/w_i}\right) && \text{(LogConcave-Weighted)}\\
\text{s.t.} \quad & \sum_{i\in\mathcal{A}} x_{ij} = 1 \quad \forall j \in \mathcal{G}\\
& \prod_{j\in S} y_j \ge 1 \quad \forall S \in \binom{\mathcal{G}}{n}.
\end{aligned}$$

This program is still convex in the weighted case and by repeating the same transformations used to prove Theorem 4.1.2 we can start from (LogConcave-Weighted) and get an equivalent convex program using the b variables.

$$\begin{aligned}
\max_{\mathbf{b}} \quad & \sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{G}} w_i\, b_{ij} \log v_{ij} - \sum_{j\in\mathcal{G}}\sum_{i\in\mathcal{A}} w_i\, b_{ij} \log\left(\sum_{i'\in\mathcal{A}} w_i\, b_{i'j}\right) + \sum_{i\in\mathcal{A}} w_i \log w_i && \text{(CVX-Weighted)}\\
\text{s.t.} \quad & \sum_{j\in\mathcal{G}} b_{ij} = 1 \quad \forall i \in \mathcal{A}\\
& \sum_{i\in\mathcal{A}} b_{ij} \le 1 \quad \forall j \in \mathcal{G}\\
& b_{ij} \ge 0 \quad \forall (i,j) \in \mathcal{A}\times\mathcal{G}
\end{aligned}$$

By setting $\mathbf{b}$ to be the same value as (4.3), it is natural to see that both programs are indeed relaxations.

Theorem 4.1.3. (NCVX-Weighted)*,* (LogConcave-Weighted)*, and* (CVX-Weighted) *are relaxations of the weighted Nash Social Welfare problem.*

Moreover, when the weights are symmetric, i.e., $w_i = 1/n$ *for all* $i \in \mathcal{A}$*, the programs* (CVX-Weighted) *and* (NCVX-Weighted) *are equivalent to the convex program* (CVX-Unweighted)*.*

We formally prove Theorem 4.1.3 in Section 4.2, and we show that (CVX-Weighted) is equivalent to (LogConcave-Weighted) in Appendix 4.6.2. Both (NCVX-Weighted) and (CVX-Weighted) will both be useful in constructing our approximation algorithm.

Note that (NCVX-Weighted) and (CVX-Weighted) have the same constraints and feasible region, $\mathcal{P}(\mathcal{A}, \mathcal{G})$, and the only difference is in the objective functions. We define

$$f_{\text{ncvx}}(\mathbf{b}) := \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{G}} w_i \, b_{ij} \log v_{ij} - \sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} w_i \, b_{ij} \log \left(\sum_{i' \in \mathcal{A}} b_{i'j} \right), \quad \text{and}$$

$$f_{\text{cvx}}(\mathbf{b}) := \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{G}} w_i \, b_{ij} \log v_{ij} - \sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} w_i \, b_{ij} \log \left(\sum_{i' \in \mathcal{A}} w_{i'} \, b_{i'j} \right) + \sum_{i \in \mathcal{A}} w_i \log w_i.$$

With these definitions we can compactly write the two different formulations

$\max_{\mathbf{b}} \quad f_{\text{cvx}}(\mathbf{b})$ $\text{s.t.} \quad \mathbf{b} \in \mathcal{P}(\mathcal{A}, \mathcal{G})$	$\max_{\mathbf{b}} \quad f_{\text{ncvx}}(\mathbf{b})$ $\text{s.t.} \quad \mathbf{b} \in \mathcal{P}(\mathcal{A}, \mathcal{G})$
(CVX-Weighted)	(NCVX-Weighted)

We can also relate the objective value of the three programs.

Lemma 4.1.4. *The two programs* (CVX-Weighted) *and* (LogConcave-Weighted) *have the same objective value.*

Moreover, for any any feasible $\mathbf{b} \in \mathcal{P}(\mathcal{A}, \mathcal{G})$ *and weights* $w_1, \ldots, w_n > 0$ *with* $\sum_{i \in \mathcal{A}} w_i = 1$*,*

$$0 \leq f_{\text{cvx}}(\mathbf{b}) - f_{\text{ncvx}}(\mathbf{b}) \leq D_{\text{KL}}(\mathbf{w} \,\|\, \mathbf{u}) = \log n - \sum_{i \in \mathcal{A}} w_i \log \frac{1}{w_i},$$

where $\mathbf{u}$ *denotes the uniform distribution, and* $\mathbf{w}$ *is the vector of weights.*

Rounding Algorithm.

Since (CVX-Weighted) is structurally similar to (CVX-Unweighted) we may hope to apply the rounding algorithm from [61], but unfortunately (CVX-Weighted) lacks a crucial property: optimal solutions of (CVX-Weighted) need not be acyclic. Furthermore, the integrality gap of

(CVX-Weighted) is non-trivial even in the case when there are exactly n items; if there are exactly n items and all valuations are 1, then setting $b_{ij} = \frac{1}{n}$ for all i, j gives an objective value of $D_{\mathrm{KL}}(\mathbf{w} \,\|\, \mathbf{u})$, while the integral optimal value is zero. To circumvent these issues, we use (NCVX-Weighted) as an intermediate step in our rounding algorithm. This non-convex program has the desired property: given a feasible point $\mathbf{b}$, one can efficiently find another point $\widetilde{\mathbf{b}}$ without decreasing the objective f_{ncvx} such that the graph formed by support of $\widetilde{\mathbf{b}}$ is a forest, as stated in the following lemma. We formally define the support graphs in Definition 4.3.1.

Lemma 4.1.5. *Let $\overline{\mathbf{b}}$ be any feasible point in $\mathcal{P}(\mathcal{A}, \mathcal{G})$. Then there exists an acyclic solution, $\mathbf{b}^{\mathrm{forest}}$, in the support of $\overline{\mathbf{b}}$ such that*

$$f_{\mathrm{ncvx}}(\mathbf{b}^{\mathrm{forest}}) \geq f_{\mathrm{ncvx}}(\overline{\mathbf{b}}).$$

Moreover, such a solution can be found in time polynomial in $|\mathcal{A}|$ and $|\mathcal{G}|$.

Next, we establish that one can efficiently round any feasible point whose support graph is a forest to an integral assignment.

Theorem 4.1.6. *For a Nash Social Welfare instance $(\mathcal{A}, \mathcal{G}, \mathbf{v}, \mathbf{w})$, given a vector $\mathbf{b} \in \mathcal{P}(\mathcal{A}, \mathcal{G})$ such that the support of $\mathbf{b}$ is a forest, there exists a deterministic polynomial time algorithm (Algorithm 6) which returns an assignment $\sigma : \mathcal{G} \to \mathcal{A}$ such that*

$$\mathrm{NSW}(\sigma) \geq f_{\mathrm{cvx}}(\mathbf{b}) - D_{\mathrm{KL}}(\mathbf{w} \,\|\, \mathbf{u}) - 2\log 2 - \frac{1}{2e}.$$

By combining Lemma 4.1.5, Theorem 4.1.6, and Lemma 4.1.4, we obtain an approximation algorithm with approximation ratio

$$\exp\left(2\log 2 + \frac{1}{2e} + 2D_{\mathrm{KL}}(\mathbf{w} \,\|\, \mathbf{u})\right) \approx 4.81 \cdot \exp\left(2\log n - 2\sum_{i=1}^{n} w_i \log \frac{1}{w_i}\right)$$

for the weighted Nash Social Welfare problem with additive valuations. When all the weights are the same, this gives a constant factor approximation.

Theorem 1.3.1. *Let $(\mathcal{A}, \mathcal{G}, \mathbf{v}, \mathbf{w})$ be an instance of the weighted Nash Social Welfare problem with $\sum_{i \in \mathcal{A}} w_i = 1$ and $|\mathcal{A}| = n$ agents. There exists a polynomial time algorithm (Algorithm 5) that, given $(\mathcal{A}, \mathcal{G}, \mathbf{v}, \mathbf{w})$, returns an assignment $\sigma : \mathcal{G} \to \mathcal{A}$ such that*

$$\mathrm{NSW}(\sigma) \geq \mathrm{OPT} - 2\log 2 - \frac{1}{2e} - 2 \cdot D_{\mathrm{KL}}(\mathbf{w} \,\|\, \mathbf{u}),$$

where OPT *is the optimal log-objective for the instance and $D_{\mathrm{KL}}(\mathbf{w} \| \mathbf{u}) = \log n - \sum_{i \in \mathcal{A}} w_i \log \frac{1}{w_i}$.*

Algorithm 5 requires solving the convex program CVX-Weighted and for this reason is not a strongly polynomial-time algorithm. This is in contrast to the techniques used in the unweighted case, where it is possible to calculate dual variables in strongly polynomial time [150].

We remark that our algorithm for rounding (NCVX-Weighted) (Algorithm 6) is the same as that in [60]. However, our analysis is quite different. Rather than using ideas from market

interpretations of the problem, we utilize properties of (CVX-Weighted) and (NCVX-Weighted), which generalize to both the unweighted and the weighted versions of the problem.

We call $\mathcal{P}(\mathcal{A}, \mathcal{G})$ the feasibility polytope of $(\mathcal{A}, \mathcal{G})$ and will refer to points in $\mathcal{P}(\mathcal{A}, \mathcal{G})$ as either feasible points or solutions.

4.1.3 Related Work

The problem of finding the allocation that maximizes the Nash Social Welfare objective is an NP-hard problem, as was proven by [146]. Additionally, [125] showed that finding such an allocation is also APX-hard. From an algorithmic perspective, the first constant factor approximation for the unweighted version was provided in [60] using analogies from market equilibrium. [61] provided an improved analysis of the algorithm from [60] and introduced a convex programming relaxation. Using an entirely different approach, [7] also provided a constant factor approximation for the unweighted variant, where their analysis employed the theory of log-concave polynomials. The best-known approximation factor with linear valuations of 1.45 is due to [20], where they provide a pseudo-polynomial-time algorithm that finds an allocation that is envy-free up to one good and also Pareto efficient. Their algorithm is entirely combinatorial and runs in polynomial time when the valuations are bounded.

Another setting of interest is when the valuation of each agent is submodular instead of additive. For instance, [86] gave a constant factor approximation algorithm for maximizing the unweighted Nash Social Welfare function when the agents' valuations are Rado, a special subclass of submodular functions. In the weighted case, the approximation factor of this algorithm depends on the ratio of the maximum weight to the minimum weight. A constant-factor approximation algorithm for the unweighted case with submodular valuations was provided in prior work [127]. More recently, [87] gave a local search-based algorithm to obtain an $O(nw_{\max})$-approximation for the weighted case and a 4-approximation for the unweighted case with submodular valuations. Note that this $O(nw_{\max})$-approximation factor was also the previously best-known approximation for the weighted case, even when considering additive valuations.

Observe that the KL-divergence term $D_{\mathrm{KL}}(\mathbf{w} \,\|\, \mathbf{u}) = \left(\log n - \sum_{i \in \mathcal{A}} w_i \log \frac{1}{w_i}\right)$ in Theorem 1.3.1 is always upper bounded by $\log(nw_{\max})$, which is exactly the guarantee of previous work [87]. In many settings, the term $2 \cdot D_{\mathrm{KL}}(\mathbf{w} \,\|\, \mathbf{u})$ can be significantly smaller than $nw_{\max}$. For example, consider the setting where $w_1 = \frac{1}{\log n}$ and $w_i = \frac{1}{n-1}(1 - \frac{1}{\log n})$ for $i = 2, \ldots, n$, i.e., one agent has a significantly higher weight than the others. Then

$$\begin{aligned} D_{\mathrm{KL}}(\mathbf{w} \,\|\, \mathbf{u}) &= \frac{1}{\log n} \log\left(\frac{n}{\log n}\right) + \left(1 - \frac{1}{\log n}\right) \log\left(\frac{n}{n-1}\left(1 - \frac{1}{\log n}\right)\right) \\ &\leq 1 + \log\left(\frac{n}{n-1}\right) \leq 2. \end{aligned}$$

In this case, our results imply an $O(1)$-approximation, while previous results imply an $O(\frac{n}{\log n})$-approximation.

Following the initial release of this work there has been substantial progress on approximation algorithms for the weighted Nash Social Welfare for the case of additive [77] and more general submodular valuations [78]. In both cases they obtained a constant factor approximation algorithm. Their approach is based on rounding a fractional solution to the configuration formulation of the problem.

4.1.4 Structure

The proof of Theorem 4.1.2 follows straightforwardly from applying convex duality and making a couple changes of variables. We defer the proof to Appendix 4.6.2 for those who are interested in the details. In Section 4.2 we show that the generalized mathematical programs are relaxations for the weighted Nash Social Welfare problem. A summary of the approximation algorithm and its analysis is given in Section 4.3. This should make it clear why we have arrived at the given approximation factor. Most of the technical work is contained in Section 4.4, in which we analyse the rounding algorithm on tree solutions. We end with conclusions and a few open problems in Section 4.5.

4.2 Relaxations for Weighted Nash Social Welfare

In this section we verify that the programs for weighted Nash Social Welfare which we gave in the introduction are indeed relaxations.

Lemma 4.2.1. *The two programs* (NCVX-Weighted) *and* (CVX-Weighted) *are relaxations of the weighted Nash Social Welfare problem.*

Moreover, when the weights are symmetric, i.e., $w_i = 1/n$ *for all* $i \in \mathcal{A}$*, the programs* (CVX-Weighted) *and* (NCVX-Weighted) *are equivalent to the convex program* (CVX-Unweighted).

Proof. Let $\sigma : \mathcal{G} \to \mathcal{A}$ be the optimal assignment for the instance, $(\mathcal{A}, \mathcal{G}, \mathbf{v}, \mathbf{w})$. For each agent $i \in \mathcal{A}$, define $V_i = \sum_{j \in \sigma^{-1}(i)} v_{ij}$. Using σ, we define a vector $\mathbf{b} \in \mathcal{P}(\mathcal{A}, \mathcal{G})$ as

$$b_{ij} := \begin{cases} \frac{v_{ij}}{V_i} & \text{if } \sigma(j) = i \\ 0 & \text{otherwise.} \end{cases}$$

It is easy to verify that $\sum_{i \in \mathcal{A}} b_{ij} \leq 1$ for each $j \in \mathcal{G}$ and $\sum_{i \in \mathcal{G}} b_{ij} = 1$ for each $i \in \mathcal{A}$. We will now show that $f_{\text{cvx}}(\mathbf{b})$ and $f_{\text{ncvx}}(\mathbf{b})$ are both equal to $\text{NSW}(\sigma)$.

$$\begin{aligned} f_{\text{cvx}}(\mathbf{b}) &= \sum_{j \in \mathcal{G}} \frac{w_{\sigma(j)} v_{\sigma(j)j}}{V_{\sigma(j)}} \log\left(\frac{V_{\sigma(j)}}{w_{\sigma(j)}}\right) + \sum_{i \in \mathcal{A}} w_i \log w_i \\ &= \sum_{i \in \mathcal{A}} w_i \sum_{j \in \sigma^{-1}(i)} \frac{v_{ij}}{V_i} \log\left(\frac{V_i}{w_i}\right) + \sum_{i \in \mathcal{A}} w_i \log w_i \\ &\overset{(i)}{=} \sum_{i \in \mathcal{A}} w_i \log\left(\frac{V_i}{w_i}\right) + \sum_{i \in \mathcal{A}} w_i \log w_i = \sum_{i \in \mathcal{A}} w_i \log V_i = \text{NSW}(\sigma)\,, \end{aligned}$$

where (i) follows from definition of V_i.

Similarly, we have

$$\begin{aligned} f_{\text{ncvx}}(\mathbf{b}) &= \sum_{j\in\mathcal{G}} \frac{w_{\sigma(j)} v_{\sigma(j)j}}{V_{\sigma(j)}} \log v_{\sigma(j)j} - \sum_{j\in\mathcal{G}} \frac{w_{\sigma(j)} v_{\sigma(j)j}}{V_{\sigma(j)}} \log\left(\frac{v_{\sigma(j)j}}{V_{\sigma(j)}}\right) \\ &= \sum_{j\in\mathcal{G}} \frac{w_{\sigma(j)} v_{\sigma(j)j}}{V_{\sigma(j)}} \log V_{\sigma(j)} = \sum_{i\in\mathcal{A}} w_i \sum_{j\in\sigma^{-1}(i)} \frac{v_{ij}}{V_i} \log V_i \\ &= \sum_{i\in\mathcal{A}} w_i \log V_i = \text{NSW}(\sigma). \end{aligned}$$

For the second claim in the lemma, when $w_i = 1/n$ for each i, for any $\mathbf{b} \in \mathcal{P}(\mathcal{A}, \mathcal{G})$, we have

$$\begin{aligned} f_{\text{cvx}}(\mathbf{b}) =& \frac{1}{n} \sum_{i\in\mathcal{A}} \sum_{j\in\mathcal{G}} b_{ij} \log v_{ij} - \frac{1}{n} \sum_{i\in\mathcal{A}} \sum_{j\in\mathcal{G}} b_{ij} \log\left(\frac{\sum_{i'\in\mathcal{A}} b_{i'j}}{n}\right) - \log n \\ =& \frac{1}{n} \sum_{i\in\mathcal{A}} \sum_{j\in\mathcal{G}} b_{ij} \log v_{ij} - \frac{1}{n} \sum_{i\in\mathcal{A}} \sum_{j\in\mathcal{G}} b_{ij} \log\left(\sum_{i'\in\mathcal{A}} b_{i'j}\right) \\ &+ \frac{1}{n} \sum_{i\in\mathcal{A}} \sum_{j\in\mathcal{G}} b_{ij} \log n - \log n \\ =& \frac{1}{n} \sum_{i\in\mathcal{A}} \sum_{j\in\mathcal{G}} b_{ij} \log v_{ij} - \frac{1}{n} \sum_{i\in\mathcal{A}} \sum_{j\in\mathcal{G}} b_{ij} \log\left(\sum_{i'\in\mathcal{A}} b_{i'j}\right), \end{aligned}$$

where we used $\sum_{j\in\mathcal{G}} b_{ij} = 1$ for every i in the last inequality. Similarly, substituting $w_i = 1/n$ for each i in f_{ncvx} completes the proof. □

For the sake of completeness we will now verify that (LogConcave-Weighted) is also a relaxation. This program is not used in our algorithm, but in Appendix 4.6.2 we use it to derive (CVX-Weighted). Our proof relies on the same key inequality used in [7], which we repeat here for completeness.

Lemma 4.2.2. *Let $\sigma : \mathcal{G} \to \mathcal{A}$ be an assignment, and $y \in \mathbb{R}^{\mathcal{G}}$ a non-negative vector such that $\prod_{j\in S} y_j \geq 1$ for all $S \in \binom{\mathcal{G}}{n}$. If $S_i := \{j \in \mathcal{G} : \sigma(j) = 1\}$ then*

$$\sum_{i\in\mathcal{A}} \log\left(\sum_{j\in S_i} v_{ij} y_j\right) \geq \sum_{i\in\mathcal{A}} \log\left(\sum_{j\in S_i} v_{ij}\right)$$

Proof. We will expand the left-hand sum inside the logarithm. Let

$$\mathcal{S} = \left\{ S \in \binom{\mathcal{G}}{n} : |S \cap S_i| = 1 \, \forall i \in \mathcal{A} \right\}$$

denote the collection of transversals across the agents' assignments. Then

$$\sum_{i\in\mathcal{A}} \log\left(\sum_{j\in S_i} v_{ij}\ y_j\right) = \log\left(\sum_{S\in\mathcal{S}}\left(\prod_{j\in S} y_j\right)\prod_{j\in S} v_{\sigma(j)j}\right)$$
$$\geq \log\left(\sum_{S\in\mathcal{S}}\prod_{j\in S} v_{\sigma(j)j}\right) = \sum_{i\in\mathcal{A}}\log\left(\sum_{j\in S_i} v_{ij}\right).$$

The only inequality uses the fact that $\prod_{j\in S} y_j \geq 1$ term-by-term in the sum. □

Lemma 4.2.3. *The program* (LogConcave-Weighted) *is a relaxation for the weighted Nash Social Welfare problem.*

Proof. Let $\sigma : \mathcal{G} \to \mathcal{A}$ be the optimal assignment for the instance, $(\mathcal{A}, \mathcal{G}, \mathbf{v}, \mathbf{w})$. For each agent $i \in \mathcal{A}$, define $V_i = \sum_{j\in\sigma^{-1}(i)} v_{ij}$. Define a vector $\mathbf{x}$ as follows

$$x_{ij} = \begin{cases} 1 & \text{if } \sigma(j) = i \\ 0 & \text{otherwise} \end{cases}.$$

It is easy to see that $\sum_{i\in\mathcal{A}} x_{ij} = 1$ for all $j \in \mathcal{G}$. If we set $y_j = 1$ for all $j \in \mathcal{G}$, then

$$\sum_{i\in\mathcal{A}} w_i \log\left(\sum_{j\in\mathcal{G}} x_{ij}\ v_{ij}\ y_j^{1/w_i}\right) = \sum_{i\in\mathcal{A}} w_i \log V_i = \mathrm{NSW}(\sigma).$$

Now we will show that the infimum is no smaller.

For each i, let $S_i = \{j \in \mathcal{G} : x_{ij} = 1\}$ be the allocation corresponding to $\mathbf{x}$. Then

$$\sum_{i\in\mathcal{A}} w_i \log\left(\sum_{j\in\mathcal{G}} x_{ij}\, v_{ij}\, y_j^{1/w_i}\right) - \sum_{i\in\mathcal{A}} w_i \log\left(\sum_{j\in\mathcal{G}} x_{ij}\, v_{ij}\right)$$
$$= \sum_{i\in\mathcal{A}} w_i \log\left(\frac{\sum_{j\in S_i} v_{ij}\ y_j^{1/w_i}}{\sum\limits_{j\in S_i} v_{ij}}\right). \tag{4.4}$$

We will show that the right-hand side is non-negative.

Now for positive reals $c_1, \ldots, c_m$ with $\sum_{j=1}^{m} c_j = 1$, and $0 \leq p \leq q$, the weighted power mean inequality states that for any $\mathbf{z} \in \mathbb{R}^m_{\geq 0}$,

$$\left(\sum_{j=1}^{m} c_j z_j^p\right)^{1/p} \leq \left(\sum_{j=1}^{m} c_j z_j^q\right)^{1/q}. \tag{4.5}$$

This inequality follows from Jensen's inequality.

For each $i \in \mathcal{A}$, define $q_i = \frac{1}{w_i}$ and $c_j^{(i)} = \frac{v_{ij}}{\sum_{j \in S_i} v_{ij}}$ for every $j \in S_i$. Since $q_i = \frac{1}{w_i} \geq 1$, using Equation 4.5, we get

$$w_i \log \left(\frac{\sum_{j \in S_i} v_{ij}\ y_j^{1/w_i}}{\sum_{j \in S_i} v_{ij}} \right) \geq \log \left(\frac{\sum_{j \in S_i} v_{ij}\ y_j}{\sum_{j \in S_i} v_{ij}} \right)$$

for each agent i. Summing this inequality over all agents and applying Lemma 4.2.2 gives

$$\sum_{i \in \mathcal{A}} w_i \log \left(\frac{\sum_{j \in S_i} v_{ij}\ y_j^{1/w_i}}{\sum_{j \in S_i} v_{ij}} \right) \geq \sum_{i \in \mathcal{A}} \log \left(\frac{\sum_{j \in S_i} v_{ij}\ y_j}{\sum_{j \in S_i} v_{ij}} \right) \geq 0.$$

Finally, substituting this in (4.4) gives

$$\sum_{i \in \mathcal{A}} w_i \log \left(\sum_{j \in \mathcal{G}} x_{ij}\ v_{ij}\ y_j^{1/w_i} \right) \geq \sum_{i \in \mathcal{A}} w_i \log \left(\sum_{j \in \mathcal{G}} x_{ij} v_{ij} \right).$$

□

The proof that (CVX-Weighted) and (LogConcave-Weighted) are equivalent is deferred to Appendix 4.6.2, and follows from a sequence of convex duals and changes of variables. The relationship between the objectives of (CVX-Weighted) and (NCVX-Weighted) is summarized in the following lemma.

Lemma 4.2.4. *For any any feasible* $\mathbf{b}$ *and weights* $w_1, \ldots, w_n > 0$ *with* $\sum_{i \in \mathcal{A}} w_i = 1$,

$$0 \leq f_{\text{cvx}}(\mathbf{b}) - f_{\text{ncvx}}(\mathbf{b}) \leq D_{\text{KL}}(\mathbf{w} \,\|\, \mathbf{u}) = \log n - \sum_{i \in \mathcal{A}} w_i \log \frac{1}{w_i}.$$

Proof. We will show that

$$f_{\text{cvx}}(\mathbf{b}) - f_{\text{ncvx}}(\mathbf{b}) = D_{\text{KL}}(\mathbf{w} \,\|\, \mathbf{u}) - D_{\text{KL}}(\mu \,\|\, \theta)\,,$$

where μ, θ are two probability distributions on $\mathcal{G}$ given by

$$\mu(j) = \sum_{i \in \mathcal{A}} w_i\, b_{ij} \quad \text{and} \quad \theta(j) = \frac{\sum_{i \in \mathcal{A}} b_{ij}}{n}.$$

Using $\sum_{i \in \mathcal{A}} w_i = 1$ and $\sum_{j \in \mathcal{G}} b_{ij} = 1$ for each $i \in \mathcal{A}$, one can verify that $\sum_{j \in \mathcal{G}} \mu(j) = 1 = \sum_{j \in \mathcal{G}} \theta(j)$.

Expanding the difference between the functions gives

$$f_{\text{cvx}}(\mathbf{b}) - f_{\text{ncvx}}(\mathbf{b}) = \sum_{i \in \mathcal{A}} w_i \log w_i - \sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} w_i\, b_{ij} \log \left(\sum_{i' \in \mathcal{A}} w_{i'}\, b_{i'j} \right)$$

$$
\begin{aligned}
&\quad + \sum_{i,j} w_i\, b_{ij} \log\left(\sum_{i' \in \mathcal{A}} b_{i'j}\right) \\
&= \sum_{i \in \mathcal{A}} w_i \log w_i - \sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} w_i\, b_{ij} \log\left(\frac{\sum_{i' \in \mathcal{A}} w_{i'}\, b_{i'j}}{\sum_{i' \in \mathcal{A}} b_{i'j}}\right) \\
&= \sum_{i \in \mathcal{A}} w_i \log w_i + \sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} w_i\, b_{ij} \log n - \sum_{j \in \mathcal{G}} \mu(j) \log\left(\frac{\mu(j)}{\theta(j)}\right) \\
&= \sum_{i \in \mathcal{A}} w_i \log(n w_i) - \sum_{j \in \mathcal{G}} \mu(j) \log\left(\frac{\mu(j)}{\theta(j)}\right) \qquad \text{(using } \textstyle\sum_j b_{ij} = 1\text{)} \\
&= D_{\mathrm{KL}}(\mathbf{w} \,\|\, \mathbf{u}) - D_{\mathrm{KL}}(\mu \,\|\, \theta).
\end{aligned}
$$

As $D_{KL}(\mu, \theta) \geq 0$, the above equation implies

$$
f_{\mathrm{cvx}}(\mathbf{b}) - f_{\mathrm{ncvx}}(\mathbf{b}) \leq D_{\mathrm{KL}}(\mathbf{w} \,\|\, \mathbf{u}).
$$

For the lower bound, it suffices to show that $D_{\mathrm{KL}}(\mu \,\|\, \theta) \leq D_{\mathrm{KL}}(\mathbf{w} \,\|\, \mathbf{u})$. To see this, we expand the definition:

$$
\begin{aligned}
D_{\mathrm{KL}}(\mu \,\|\, \theta) &= \sum_{j \in \mathcal{G}} \left(\sum_{i \in \mathcal{A}} w_i b_{ij}\right) \log\left(\frac{n \sum_{i \in \mathcal{A}} w_i b_{ij}}{\sum_{i \in \mathcal{A}} b_{ij}}\right) \\
&= \log(n) + \sum_{j \in \mathcal{G}} \left(\sum_{i \in \mathcal{A}} w_i b_{ij}\right) \log\left(\frac{\sum_{i \in \mathcal{A}} w_i b_{ij}}{\sum_{i \in \mathcal{A}} b_{ij}}\right) \\
&= \log(n) + \sum_{j \in \mathcal{G}} \left(\sum_{i \in A} b_{ij}\right) \left(\sum_{i \in \mathcal{A}} \frac{b_{ij}}{\sum_{i \in A} b_{ij}} w_i\right) \log\left(\sum_{i \in \mathcal{A}} \frac{b_{ij}}{\sum_{i \in A} b_{ij}} w_i\right) \\
&\leq \log(n) + \sum_{j \in \mathcal{G}} \left(\sum_{i \in A} b_{ij}\right) \left(\sum_{i \in \mathcal{A}} \frac{b_{ij}}{\sum_{i \in A} b_{ij}}\right) w_i \log(w_i) \\
&= \log(n) + \sum_{j \in \mathcal{A}} w_i \log(w_i) \sum_{j \in \mathcal{G}} b_{ij} \\
&= \log(n) + \sum_{i \in \mathcal{A}} w_i \log(w_i) = D_{\mathrm{KL}}(\mathbf{w} \,\|\, \mathbf{u}).
\end{aligned}
$$

Here, the only inequality uses the convexity of $x \log(x)$, and the last equality follows from the feasibility of $\mathbf{b}$. □

4.3 Approximation Algorithm

Before describing our algorithm, we need the following definitions.

Definition 4.3.1 (Support Graph)**.** For a vector $\mathbf{b} \in \mathcal{P}(\mathcal{A}, \mathcal{G})$, the support graph of $\mathbf{b}$, denoted by $G_{\text{supp}}(\mathbf{b})$, is a bipartite graph with vertex set $\mathcal{A} \cup \mathcal{G}$. For any $i \in \mathcal{A}$ and $j \in \mathcal{G}$, the edge (i, j) belongs to the edge set of G if and only if $b_{ij} > 0$.

Definition 4.3.2 (Acyclic Solution)**.** A vector $\mathbf{b} \in \mathcal{P}(\mathcal{A}, \mathcal{G})$ is called an acyclic solution if the support graph of $\mathbf{b}$, $G_{\text{supp}}(\mathbf{b})$, does not contain any cycles.

For ease of notation, given any feasible point $\mathbf{b} \in \mathcal{P}(\mathcal{A}, \mathcal{G})$, we use vector $\mathbf{q} \in \mathbb{R}^{|\mathcal{G}|}$ to denote the projection of $\mathbf{b}$ to $\mathcal{G}$, i.e.,

$$q_j := \sum_{i \in \mathcal{A}} b_{ij}$$

for each $j \in \mathcal{G}$. Since $\mathbf{q}$ is completely defined by $\mathbf{b}$, with abuse of notation, we will interchangeably use $\mathcal{P}(\mathcal{A}, \mathcal{G})$ to denote feasible vectors $\mathbf{b}$ as well as $(\mathbf{b}, \mathbf{q})$. Similarly, we will use $f_{\text{ncvx}}(\mathbf{b}, \mathbf{q})$ and $f_{\text{cvx}}(\mathbf{b}, \mathbf{q})$ to also denote the objective $f_{\text{ncvx}}(\mathbf{b})$ and $f_{\text{cvx}}(\mathbf{b})$, respectively. With a slight abuse of notation, we define

$$f_{\text{ncvx}}(\mathbf{b}, \mathbf{q}) := \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{G}} w_i \, b_{ij} \log v_{ij} - \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{G}} w_i \, b_{ij} \log q_j.$$

for any $\mathbf{b} \in \mathcal{P}(\mathcal{A}, \mathcal{G})$ and its projection $\mathbf{q} \in \mathbb{R}^{|G|}$.

Our main algorithm, Algorithm 5, begins by finding the optimal solution $\overline{\mathbf{b}}$ to the convex program (CVX-Weighted). It then constructs another feasible point, $\mathbf{b}^{\text{forest}}$, in support of $\overline{\mathbf{b}}$ such that the support graph of $\mathbf{b}^{\text{forest}}$ is a forest and f_{ncvx} at $\mathbf{b}^{\text{forest}}$ is at least f_{ncvx} at $\overline{\mathbf{b}}$. In the final step, the algorithm rounds $\mathbf{b}^{\text{forest}}$ to an integral solution using Algorithm 6. Theorem 4.1.6 establishes a bound on the rounding error incurred during Algorithm 6.

Algorithm 5 Approximation Algorithm for Weighted Nash Social Welfare

Require: NSW instance $(\mathcal{A}, \mathcal{G}, \mathbf{v}, \mathbf{w})$

1: $\overline{\mathbf{b}} \leftarrow$ optimal solution of (CVX-Weighted)
2: $\overline{\mathbf{q}} \leftarrow$ vector in $\mathbb{R}^{|\mathcal{G}|}$ with $\overline{q}_j = \sum_{i \in \mathcal{A}} \overline{b}_{ij}$
3: $(\mathbf{b}^{\text{forest}}, \mathbf{q}^{\text{forest}}) \leftarrow$ acyclic solution in support of $\overline{\mathbf{b}}$ such that $f_{\text{ncvx}}(\mathbf{b}^{\text{forest}}) \geq f_{\text{ncvx}}(\overline{\mathbf{b}})$
4: $\sigma \leftarrow$ output of Algorithm 6 with input $(\mathcal{A}, \mathcal{G}, \mathbf{v}, \mathbf{w}, \mathbf{b}^{\text{forest}}, \mathbf{q}^{\text{forest}})$

Ensure: σ

Lemma 4.1.5, which we re-state below for the reader's convenience, guarantees the existence of $\mathbf{b}^{\text{forest}}$, ensuring that the algorithm is well-defined. It is worth mentioning that for the unweighted case, the existence of an acyclic optimum was utilized by [60, 61] for the convex program (CVX-Unweighted). In the weighted setting, this structural property is not inherited by the convex program (CVX-Weighted) but by the non-convex program (NCVX-Weighted).

Lemma 4.1.5. *Let $\overline{\mathbf{b}}$ be any feasible point in $\mathcal{P}(\mathcal{A}, \mathcal{G})$. Then there exists an acyclic solution, $\mathbf{b}^{\text{forest}}$, in the support of $\overline{\mathbf{b}}$ such that*

$$f_{\text{ncvx}}(\mathbf{b}^{\text{forest}}) \geq f_{\text{ncvx}}(\overline{\mathbf{b}}).$$

Moreover, such a solution can be found in time polynomial in $|\mathcal{A}|$ and $|\mathcal{G}|$.

Proof. Let $G_{\text{supp}}(\bar{\mathbf{b}})$ contain a cycle $(i_0, j_0, i_1, \ldots, j_{\ell-1}, i_\ell)$ with $i_0 = i_\ell$, where $i_x \in \mathcal{A}$ and $j_y \in \mathcal{G}$. The main idea is to modify the variables $\bar{\mathbf{b}}$ on this cycle while ensuring the value of $\bar{\mathbf{q}}$ does not change. If $\bar{\mathbf{q}}$ is fixed, then $f_{\text{ncvx}}(\cdot, \bar{\mathbf{q}})$ is linear in the input, and as a result, we can *cancel* the cycle by considering the following vector. Define $\boldsymbol{\delta} \in \mathbb{R}^{|\mathcal{A}|\times|\mathcal{G}|}$ with $\delta_{i_x j_x} := 1$ and $\delta_{i_{x+1} j_x} := -1$ for $x \in \{0, \ldots, \ell - 1\}$, and $\delta_{ij} := 0$ otherwise.

Note that $\sum_{i\in\mathcal{A}} \delta_{ij} = 0$ for any item j. As a result, for each $j \in \mathcal{G}$,

$$\sum_{i\in\mathcal{A}} \big(\bar{b}_{ij} + \varepsilon\delta_{ij}\big) = \sum_{i\in\mathcal{A}} \bar{b}_{ij} = \bar{q}_j.$$

Therefore, the change in f_{ncvx} is given by

$$\begin{aligned} f_{\text{ncvx}}(\bar{\mathbf{b}} + \varepsilon\boldsymbol{\delta}, \bar{\mathbf{q}}) - f_{\text{ncvx}}(\bar{\mathbf{b}}, \bar{\mathbf{q}}) &= \sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{G}} \varepsilon\, w_i\, \delta_{ij} \log v_{ij} - \sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{G}} \varepsilon\, w_i\, \delta_{ij} \log \bar{q}_j \\ &:= \varepsilon\; h(\boldsymbol{\delta}, \bar{\mathbf{q}}). \end{aligned}$$

Note that $h(\boldsymbol{\delta}, \bar{\mathbf{q}})$ is a linear function in $\boldsymbol{\delta}$. So, if $h(\boldsymbol{\delta}, \bar{\mathbf{q}}) > 0$, then setting $\varepsilon = \max_x b_{i_{x+1} j_x}$ ensures that $f_{\text{ncvx}}(\bar{\mathbf{b}} + \varepsilon\boldsymbol{\delta}, \bar{\mathbf{q}}) \geq f_{\text{ncvx}}(\bar{\mathbf{b}}, \bar{\mathbf{q}})$, and $\bar{\mathbf{b}} + \varepsilon\boldsymbol{\delta} \in \mathcal{P}(\mathcal{A}, \mathcal{G})$. In addition, the number of cycles in $G_{\text{supp}}(\bar{\mathbf{b}} + \varepsilon\boldsymbol{\delta})$ is strictly less than the number of cycles in $G_{\text{supp}}(\bar{\mathbf{b}})$.

Similarly, if $h(\boldsymbol{\delta}, \bar{\mathbf{q}}) \leq 0$, setting $\varepsilon = -\max_x \bar{b}_{i_x j_x}$ gives the same guarantees. Iterating this cycle canceling process until the support contains no cycles leads to the required solution. □

Finally, we can use Lemma 4.2.4 to bound the difference between f_{cvx} and f_{ncvx}.

By combining Lemma 4.1.5 with Lemma 4.2.4, we obtain the following corollary.

Corollary 4.3.3. *Let $\bar{\mathbf{b}}$ be any feasible point in $\mathcal{P}(\mathcal{A}, \mathcal{G})$. Then, there exists an acyclic solution, $\mathbf{b}^{\text{forest}}$, in the support of $\bar{\mathbf{b}}$ such that*

$$f_{\text{cvx}}(\mathbf{b}^{\text{forest}}) \geq f_{\text{cvx}}(\bar{\mathbf{b}}) - D_{\text{KL}}(\mathbf{w} \,\|\, \mathbf{u}).$$

Moreover, such a $\mathbf{b}^{\text{forest}}$ can be found in time polynomial in $|\mathcal{A}|$ and $|\mathcal{G}|$.

Proof. Given a feasible point $\bar{\mathbf{b}} \in \mathcal{P}(\mathcal{A}, \mathcal{G})$, Lemma 4.2.4 implies that

$$f_{\text{cvx}}(\bar{\mathbf{b}}) - D_{\text{KL}}(\mathbf{w} \,\|\, \mathbf{u}) \leq f_{\text{ncvx}}(\bar{\mathbf{b}}).$$

Now, we apply Lemma 4.1.5 to get an acyclic solution $\mathbf{b}^{\text{forest}}$, in the support of $\bar{\mathbf{b}}$ such that

$$f_{\text{ncvx}}(\bar{\mathbf{b}}) \leq f_{\text{ncvx}}(\mathbf{b}^{\text{forest}}).$$

Finally, using the other side of Lemma 4.2.4, we see that

$$f_{\text{ncvx}}(\mathbf{b}^{\text{forest}}) \leq f_{\text{cvx}}(\mathbf{b}^{\text{forest}}).$$

Combining these inequalities gives the desired conclusion. □

Before presenting Algorithm 6, we give the proof of Theorem 1.3.1, which now follows directly from Theorem 4.1.6 and Corollary 4.3.3, as outlined below.

Proof of Theorem 1.3.1. Let $(\overline{\mathbf{b}}, \overline{\mathbf{q}})$ and $(\mathbf{b}^{\text{forest}}, \mathbf{q}^{\text{forest}})$ denote the feasible points defined in Step 1 and Step 3 of Algorithm 5, respectively. Let $\sigma^\star$ be the assignment returned by Algorithm 6 on input $(\mathbf{b}^{\text{forest}}, \mathbf{q}^{\text{forest}})$. By Theorem 4.1.6, we have

$$\begin{aligned}\text{NSW}(\sigma^\star) &\geq f_{\text{cvx}}(\mathbf{b}^{\text{forest}}, \mathbf{q}^{\text{forest}}) - D_{\text{KL}}(\mathbf{w} \,\|\, \mathbf{u}) - 2\log 2 - \frac{1}{2e} \\ &\overset{(i)}{\geq} f_{\text{cvx}}(\overline{\mathbf{b}}, \overline{\mathbf{q}}) - 2 \cdot D_{\text{KL}}(\mathbf{w} \,\|\, \mathbf{u}) - 2\log 2 - \frac{1}{2e} \\ &\overset{(ii)}{\geq} \text{OPT} - 2 \cdot D_{\text{KL}}(\mathbf{w} \,\|\, \mathbf{u}) - 2\log 2 - \frac{1}{2e}.\end{aligned}$$

Here, (i) follows from Corollary 4.3.3 and (ii) follows from Lemma 4.2.1. □

4.3.1 Rounding an Acyclic Solution

Given an acyclic solution $\mathbf{b}$, Algorithm 6 returns an assignment, $\sigma^\star$, such that $\text{NSW}(\sigma^\star)$ is comparable to $f_{\text{cvx}}(\mathbf{b})$, as stated in Theorem 4.1.6.

Algorithm 6 Algorithm for Rounding an Acyclic Solution

Require: NSW instance $(\mathcal{A}, \mathcal{G}, \mathbf{v}, \mathbf{w})$, acyclic solution $(\mathbf{b}, \mathbf{q}) \in \mathcal{P}(\mathcal{A}, \mathcal{G})$

1: $(\mathbf{b}^\star, \mathbf{q}^\star) \leftarrow$ optimal solution of (CVX-Weighted) restricted to the support of $(\mathbf{b}, \mathbf{q})$

2: $F^\star \leftarrow G_{\text{supp}}(\mathbf{b}^\star)$ with every tree rooted at an agent node

3: $\widetilde{F} \leftarrow$ forest obtained by removing edges between item j and its children in $F^\star$ whenever $q_j^\star < \frac{1}{2}$ ▷ pruning step

4: $L_i^\star \leftarrow$ set of leaf children of agent i in $\widetilde{F}$; $\quad L^\star \leftarrow \bigcup_i L_i^\star$

5: $M^\star \leftarrow$ matching between $\mathcal{A} \to \mathcal{G} \setminus L^\star$ in $\widetilde{F}$ maximizing:

$$w_{\widetilde{F}}(M) := \sum_{i \in \mathcal{A}} w_i \log \left(v_{iM(i)} + \sum_{j \in L_i^\star} v_{ij} \right)$$

6: $\sigma^\star \leftarrow$ assignment of $\mathcal{G}$ to $\mathcal{A}$ with $\sigma^\star(j) = i$ if $j \in L_i^\star \cup M^\star(i)$ ▷ matching step

Ensure: $\sigma^\star$

In the first step, Algorithm 6 finds an optimal solution, denoted by $\mathbf{b}^\star$, to the convex program (CVX-Weighted) restricted to the support of $\mathbf{b}$, i.e., $\mathbf{b}^\star$ is the optimal solution to (CVX-Weighted) on input $(\mathcal{A}, \mathcal{G}, \widetilde{\mathbf{v}}, \mathbf{w})$, where $\widetilde{v}_{ij} = 0$ if $b_{ij} = 0$, and $\widetilde{v}_{ij} = v_{ij}$ otherwise. This step is crucial as it allows us to utilize the stability properties of stationary points of (CVX-Weighted).

[0]If agent i in unmatched in M, we let $v_{iM(i)} = 0$

Next, the algorithm implements a "pruning" step to sparsify $\mathbf{b}^\star$: it removes edges between any item with $q_j^\star < 1/2$ and its children in $F^\star$. Here, $F^\star$ is the support graph of $\mathbf{b}^\star$ with every tree rooted at agent nodes. This step is equivalent to assigning each item j with $q_j^\star < 1/2$ to its parent agent in $F^\star$. As a result, any item with $q_j^\star < 1/2$ is a leaf in the pruned forest, $\widetilde{F}$. Since removing edges will exclude certain items from being assigned to some agents, pruning can lead to a sub-optimal solution. We bound this loss in objective by showing the existence of a fractional solution $(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}})$ whose support graph is a subset of the pruned forest, $\widetilde{F}$, and $f_{\text{cvx}}(\mathbf{b}^{\text{pruned}})$ is comparable to $f_{\text{cvx}}(\mathbf{b}^\star)$. For concrete details, see Section 4.4.

It is important to emphasize that the algorithm does not need to find the solution $(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}})$. The mere existence of $(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}})$ is enough to guarantee that the assignment returned by the algorithm will be good, as explained below.

After the pruning step, the algorithm assigns every leaf item in the pruned forest to its parent. We use $L_i^\star$ to denote the set of leaf items whose parent is agent i and $L^\star = \cup_{i \in \mathcal{A}} L_i^\star$ to denote the set of all leaf items in the pruned forest. So, each agent i receives all the items in the bundle $L_i^\star$. In the matching step, the algorithm assigns at most one additional item to each agent by finding a maximum weight matching between agents $\mathcal{A}$ and items $\mathcal{G} \backslash L^\star$ (the set of non-leaf items in the pruned forest). This matching is determined using an augmented weight function, denoted by $w_{\widetilde{F}}$. The weight of a matching M between $\mathcal{A}$ and $\mathcal{G} \setminus L^\star$ in the pruned forest is defined as follows:

$$w_{\widetilde{F}}(M) := \sum_{i \in \mathcal{A}} w_i \log \left(v_{iM(i)} + \sum_{j \in L_i^\star} v_{ij} \right) ,$$

where $v_{iM(i)} = 0$ if i is not matched in M. Observe that this weight function exactly captures the weighted Nash Social Welfare objective when agent i is assigned the item set $S_i := M(i) \cup L_i^\star$ for each $i \in \mathcal{A}$. Moreover, finding the optimal matching M can be easily formulated as a maximum weight matching problem in a bipartite graph.

Since the standard linear programming relaxation for the bipartite matching problem is integral, it is enough to demonstrate the existence of a *fractional matching* with a large weight $w_{\widetilde{F}}$ in the pruned forest. In Section 4.4.2, we show how to construct a fractional matching corresponding to $\mathbf{b}^{\text{pruned}}$, such that the weight of this matching is comparable to the objective $f_{\text{ncvx}}(\mathbf{b}^{\text{pruned}})$. We emphasize that this matching corresponding to $\mathbf{b}^{\text{pruned}}$ is only required for the sake of analysis: to lower bound the performance of the matching returned by the algorithm. We do not need to know $\mathbf{b}^{\text{pruned}}$ for the execution of the algorithm.

4.4 Rounding via the Non-Convex Relaxation

In this section, we prove Theorem 4.1.6 by establishing properties of support-restricted optimal solutions of (CVX-Weighted). First, in Lemma 4.4.1, we show that any optimum whose support is restricted to a forest can be "pruned" to a feasible solution while only losing a constant factor in the objective. Specifically, we show that given a support restricted optimum $(\mathbf{b}^\star, \mathbf{q}^\star)$, we can construct a feasible solution $(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}})$ such that any item with $q_j^{\text{pruned}} < 1/2$ is a leaf in

support graph of $\mathbf{b}^{\text{pruned}}$, and $f_{\text{cvx}}(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}}) \geq f_{\text{cvx}}(\mathbf{b}^\star, \mathbf{q}^\star) - \log 2$.

Second, in Lemma 4.4.2, we demonstrate the existence of a matching in the support graph of $\mathbf{b}^{\text{pruned}}$ such that the augmented weight function of this matching differs from $f_{\text{ncvx}}(\mathbf{b}^{\text{pruned}})$ by a constant factor. After presenting these two lemmas, we provide the proof of Theorem 4.1.6.

Lemma 4.4.1. *Let $(\mathbf{b}^\star, \mathbf{q}^\star)$ be the optimal solution of* (CVX-Weighted) *in the support of some acyclic feasible point $\mathbf{b}^{\text{forest}}$. Let F be a directed forest formed by $G_{\text{supp}}(\mathbf{b}^\star)$ when every tree is rooted at an agent node. Then, there exists an acyclic feasible point $(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}})$ in $\mathcal{P}(\mathcal{A}, \mathcal{G})$ such that $G_{\text{supp}}(\mathbf{b}^{\text{pruned}})$ is a subgraph of $G_{\text{supp}}(\mathbf{b}^\star)$ and*

- *$q_j^{\text{pruned}} \geq q_j^\star$ for any item j with $q_j^\star \geq 1/2$,*
- *each item with $q_j^\star < 1/2$ is a leaf in $G_{\text{supp}}(\mathbf{b}^{\text{pruned}})$ connected to its parent in F, and*
- *$f_{\text{cvx}}(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}}) \geq f_{\text{cvx}}(\mathbf{b}^\star, \mathbf{q}^\star) - \log 2$.*

The proof of Lemma 4.4.1 relies on the stability properties of optimal solutions of (CVX-Weighted), as outlined in Section 4.4.1.

Lemma 4.4.2. *Let $(\mathbf{b}, \mathbf{q})$ be an acyclic solution in $\mathcal{P}(\mathcal{A}, \mathcal{G})$ such that every item with $q_j < 1/2$ is a leaf in $G_{\text{supp}}(\mathbf{b})$. Let $S : \mathcal{A} \to 2^{\mathcal{G}}$ be a function such that for each agent i, $S(i)$ is a subset of the leaf items connected to agent i in $G_{\text{supp}}(\mathbf{b})$, and $S(i)$ contains all children of agent i with $q_j < 1/2$. Then, there exists a matching M in $G_{\text{supp}}(\mathbf{b})$ between the vertices in $\mathcal{A}$ and $\{\mathcal{G} \setminus \cup_i \{S(i)\}\}$ such that*

$$\sum_{i \in \mathcal{A}} w_i \log \left(v_{iM(i)} + \sum_{j \in S(i)} v_{ij} \right) \geq f_{\text{ncvx}}(\mathbf{b}, \mathbf{q}) - \log 2 - \frac{1}{2e},$$

where $v_{iM(i)} = 0$ if agent i is not matched in M.

We prove this lemma in Section 4.4.2.

Proof of Theorem 4.1.6. Given $(\mathbf{b}, \mathbf{q})$ such that $G_{\text{supp}}(\mathbf{b})$ is a forest, let $(\mathbf{b}^\star, \mathbf{q}^\star)$ be the optimal solution of (CVX-Weighted) restricted to support of $\mathbf{b}$, let $\widetilde{F}$ denote the forest obtained after pruning $G_{\text{supp}}(\mathbf{b}^\star)$. Let $L_i^\star$ denote the set of leaf children of agent i in $\widetilde{F}$.

Let $(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}})$ be a feasible solution guaranteed by Lemma 4.4.1 on input $(\mathbf{b}^\star, \mathbf{q}^\star)$. Since Lemma 4.4.1 guarantees that $G_{\text{supp}}(\mathbf{b}^{\text{pruned}})$ is a subset of $G_{\text{supp}}(\mathbf{b}^\star)$, and every item with $q_j^\star < 1/2$ is a leaf in $G_{\text{supp}}(\mathbf{b}^{\text{pruned}})$, we conclude that $G_{\text{supp}}(\mathbf{b}^{\text{pruned}})$ is a subgraph of $\widetilde{F}$.

In addition, $L_i^\star$ is a subset of the leaf children of i in $G_{\text{supp}}(\mathbf{b}^{\text{pruned}})$ as $G_{\text{supp}}(\mathbf{b}^{\text{pruned}})$ is a subgraph of $\widetilde{F}$. Furthermore, if $q_j^{\text{pruned}} < 1/2$, then we claim that j is a leaf in $G_{\text{supp}}(\mathbf{b}^{\text{pruned}})$ with parent i such that $j \in L_i^\star$ in $\widetilde{F}$. Since $q_j^{\text{pruned}} < 1/2$, by the first point of Lemma 4.4.1, we have $q_j^\star < 1/2$. As a result, item j is a leaf in $G_{\text{supp}}(\mathbf{b}^{\text{pruned}})$ connected to its parent in $\widetilde{F}$. So, item j would be pruned in $\widetilde{F}$, and therefore, by definition, $j \in L_i^\star$.

Therefore, for each agent i, the set $L_i^\star$ is a subset of the set of leaves of agent i in $G_{\text{supp}}(\mathbf{b}^{\text{pruned}})$, and $L_i^\star$ contains all the items with $q_j^{\text{pruned}} < 1/2$ in $G_{\text{supp}}(\mathbf{b}^{\text{pruned}})$. So, the function $S(i) = L_i^\star$ satisfies the constraints of Lemma 4.4.2 with input $(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}})$.

Using Lemma 4.4.2 on $(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}})$ with function $S(i) = L_i^\star$, we conclude that there exists a matching, M, in $G_{\text{supp}}(\mathbf{b}^{\text{pruned}})$ such that

$$\begin{aligned}\sum_{i\in\mathcal{A}} w_i \log\left(v_{iM(i)} + \sum_{j\in L_i^\star} v_{ij}\right) &= \sum_{i\in\mathcal{A}} w_i \log\left(v_{iM(i)} + \sum_{j\in S(i)} v_{ij}\right)\\ &\geq f_{\text{ncvx}}(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}}) - \log 2 - \frac{1}{2e}.\end{aligned}$$

Since $G_{\text{supp}}(\mathbf{b}^{\text{pruned}})$ is a subgraph of $\widetilde{F}$, the matching M is also present in $\widetilde{F}$. Therefore, the matching $M^\star$ (and corresponding assignment $\sigma^\star$) returned by Algorithm 6 satisfies

$$\begin{aligned}\text{NSW}(\sigma^\star) = \sum_{i\in\mathcal{A}} w_i \log\left(v_{iM^\star(i)} + \sum_{j\in L_i^\star} v_{ij}\right) &\overset{(i)}{\geq} \sum_{i\in\mathcal{A}} w_i \log\left(v_{iM(i)} + \sum_{j\in L_i^\star} v_{ij}\right)\\ &\overset{(ii)}{\geq} f_{\text{ncvx}}(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}}) - \log 2 - \frac{1}{2e}\\ &\overset{(iii)}{\geq} f_{\text{cvx}}(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}}) - D_{\text{KL}}(\mathbf{w} \,\|\, \mathbf{u}) - \log 2 - \frac{1}{2e}\\ &\overset{(iv)}{\geq} f_{\text{cvx}}(\mathbf{b}^\star, \mathbf{q}^\star) - D_{\text{KL}}(\mathbf{w} \,\|\, \mathbf{u}) - 2\log 2 - \frac{1}{2e}\\ &\overset{(v)}{\geq} f_{\text{cvx}}(\mathbf{b}, \mathbf{q}) - D_{\text{KL}}(\mathbf{w} \,\|\, \mathbf{u}) - 2\log 2 - \frac{1}{2e}.\end{aligned}$$

Here, (i) follows from the optimality of $M^\star$, (ii) follows from Lemma 4.4.2, (iii) follows from Lemma 4.2.4, (iv) follows from Lemma 4.4.1, and (v) follows from the optimality of $\mathbf{b}^\star$. □

4.4.1 Pruning Small Items

In this section, we prove Lemma 4.4.1 by establishing some properties of the set of (support restricted) optimal solutions of (CVX-Weighted) in Lemma 4.4.3 and Lemma 4.4.4.

First, we show that any optimal solution of (CVX-Weighted) is relatively stable, i.e., the change in function value when moving away from the optimal solution can be quantified in terms of how much we deviate from that solution. We formalize the stability property as follows.

Lemma 4.4.3. *Let* $(\mathbf{b}^\star, \mathbf{q}^\star)$ *be the optimal solution of* (CVX-Weighted) *in the support of some acyclic feasible point* $\mathbf{b}^{\text{forest}}$. *Let* $(\mathbf{b}, \mathbf{q})$ *be a feasible point in* $\mathcal{P}(\mathcal{A}, \mathcal{G})$ *such that the support of* $\mathbf{b}$ *is a subset of the support of* $\mathbf{b}^\star$, *and for any* $j \in \mathcal{G}$, *if* $q_j^\star = 1$, *then* $q_j = 1$. *Then*

$$f_{\text{cvx}}(\mathbf{b}^\star, \mathbf{q}^\star) - f_{\text{cvx}}(\mathbf{b}, \mathbf{q}) = \sum_{j\in\mathcal{G}} \sum_{i\in\mathcal{A}} w_i\, b_{ij} \log\left(\frac{\sum_{i'\in\mathcal{A}} w_{i'}\, b_{i'j}}{\sum_{i\in\mathcal{A}} w_{i'}\, b^\star_{i'j}}\right).$$

Second, in Lemma 4.4.4, we show that any acyclic optimal solution of (CVX-Weighted) can be pruned to a feasible solution, denoted by $\mathbf{b}^{\text{pruned}}$, which is amenable to rounding. Specifically, we show that given a first-order stationary point $(\mathbf{b}^\star, \mathbf{q}^\star)$, we can construct a feasible solution $(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}})$ such that any item with $q_j^{\text{pruned}} < 1/2$ is a leaf in support of $\mathbf{b}^{\text{pruned}}$ and $b_{ij}^{\text{pruned}} \leq \min\{1, 2b_{ij}^\star\}$ for any agent i and item j.

Lemma 4.4.4. *Let $(\mathbf{b}^\star, \mathbf{q}^\star)$ be an acyclic feasible point in $\mathcal{P}(\mathcal{A}, \mathcal{G})$. Let F be a directed forest formed by $G_{\text{supp}}(\mathbf{b}^\star)$ when every tree is rooted at an arbitrary agent node. Then, there exists a feasible solution $(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}})$ such that $G_{\text{supp}}(\mathbf{b}^{\text{pruned}})$ is a subgraph of $G_{\text{supp}}(\mathbf{b}^*)$,*

- *$q_j^\star \leq q_j^{\text{pruned}}$ for each item j with $q_j^\star \geq 1/2$,*
- *each item with $q_j^\star < 1/2$ is a leaf in $G_{\text{supp}}(\mathbf{b}^{\text{pruned}})$ connected to its parent in F, and*
- *for any $(i, j) \in \mathcal{A} \times \mathcal{G}$, $b_{ij}^{\text{pruned}} \leq \min\{1, 2 \cdot b_{ij}^\star\}$.*

Before proving Lemma 4.4.3 and Lemma 4.4.4, we use them to prove Lemma 4.4.1.

Proof of Lemma 4.4.1. By Lemma 4.4.4, there exists a feasible solution $(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}})$ such that the support graph, $G_{\text{supp}}(\mathbf{b}^{\text{pruned}})$, is a subgraph of $G_{\text{supp}}(\mathbf{b})$ and $(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}})$ satisfies the first two items claimed in the lemma. Furthermore, for any $(i, j) \in \mathcal{A} \times \mathcal{G}$, $b_{ij}^{\text{pruned}} \leq \min\{1, 2 \cdot b_{ij}^\star\}$.

Using Lemma 4.4.3, the difference in objective from $(\mathbf{b}^\star, \mathbf{q}^\star)$ to $(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}})$ is bounded as follows

$$\begin{aligned} &f_{\text{cvx}}(\mathbf{b}^\star, \mathbf{q}^\star) - f_{\text{cvx}}(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}}) \\ &\qquad = \sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} w_i \, b_{ij}^{\text{pruned}} \log \left(\frac{\sum_{i' \in \mathcal{A}} w_{i'} \, b_{i'j}^{\text{pruned}}}{\sum_{i' \in \mathcal{A}} w_{i'} \, b_{i'j}^\star} \right). \end{aligned}$$

Since $b_{ij}^{\text{pruned}} \leq \min\{1,\, 2\, b_{ij}^\star\}$, we have $\sum_i w_i \, b_{ij}^{\text{pruned}} \leq 2 \sum_i w_i \, b_{ij}^\star$ for each (i, j).

$$f_{\text{cvx}}(\mathbf{b}^\star, \mathbf{q}^\star) - f_{\text{cvx}}(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}}) \leq \sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} w_i \, b_{ij}^{\text{pruned}} \log 2. \tag{4.6}$$

The feasibility of $\mathbf{b}^{\text{pruned}}$ implies

$$\sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} w_i \, b_{ij}^{\text{pruned}} = \sum_{i \in \mathcal{A}} w_i \sum_{j \in \mathcal{G}} b_{ij}^{\text{pruned}} = \sum_{i \in \mathcal{A}} w_i = 1.$$

Plugging this bound in Equation (4.6) completes the proof. □

In the rest of this section, we will provide proofs of the two component lemmas. The proof of Lemma 4.4.3 consists of considering the Lagrangian relaxation to investigate the optimality conditions.

Proof of Lemma 4.4.3. We can define the Lagrangian relaxation of CVX-Weighted with additional real variables λ_i for each $i \in \mathcal{A}$, $\eta_j \geq 0$ for each $j \in \mathcal{G}$, and $\alpha_{ij} \geq 0$ for every $(i,j) \in \mathcal{A} \times \mathcal{G}$.

$$L(b, q; \lambda, \eta) = f_{\text{ncvx}}(b, q) + \sum_{i \in \mathcal{A}} \left(1 - \sum_{j \in \mathcal{G}} b_{ij}\right) + \sum_{j \in \mathcal{G}} \eta_j \left(1 - \sum_{i \in \mathcal{A}} b_{ij}\right) + \alpha_{ij} b_{ij}.$$

Recall that

$$f_{\text{ncvx}}(b, q) = \sum_{i \in \mathcal{A}, j \in \mathcal{G}} w_i b_{ij} \log v_{ij} - \sum_{i \in \mathcal{A}, j \in \mathcal{G}} w_i b_{ij} \log \left(\sum_{i' \in \mathcal{A}} w_{i'} b_{i'j}\right) + \sum_{i \in \mathcal{A}} w_i \log w_i.$$

If $\mathbf{b}^\star$ is an optimal solution of (CVX-Weighted), then using the KKT conditions, there exist real numbers λ_i for each $i \in \mathcal{A}$, $\eta_j \geq 0$ for each $j \in \mathcal{G}$, and $\alpha_{ij} \geq 0$ for every $(k,l) \in \mathcal{A} \times \mathcal{G}$ such that

$$\left.\frac{\partial L}{\partial b_{kl}}\right|_{b^\star} = w_k \log v_{kl} - w_k - w_k \log \left(\sum_{i' \in \mathcal{A}} w_{i'} b^\star_{i'l}\right) - \lambda_k - \eta_l + \alpha_{kl} = 0.$$

In addition, by complementary slackness, we have $\eta_j(1 - \sum_{i \in \mathcal{A}} b^\star_{ij}) = 0$ for each item j and $\alpha_{ij} b^\star_{ij} = 0$ for each $(i,j) \in \mathcal{A} \times \mathcal{G}$. Using these complementary slackness conditions, if $b^\star_{ij} > 0$, then

$$w_i \log v_{ij} = w_i + w_i \log \left(\sum_{i' \in \mathcal{A}} w_{i'} b^\star_{i'j}\right) + \lambda_i + \eta_j. \tag{4.7}$$

Now, expanding the difference between the two function values, we get

$$\begin{aligned} f_{\text{cvx}}(\mathbf{b}^\star, \mathbf{q}^\star) - f_{\text{cvx}}(\mathbf{b}, \mathbf{q}) = & \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{G}} \left(b^\star_{ij} - b_{ij}\right) \cdot w_i \log v_{ij} \\ & - \sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} w_i \, b^\star_{ij} \log \left(\sum_{i' \in \mathcal{A}} w_{i'} \, b^\star_{i'j}\right) \\ & + \sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} w_i \, b_{ij} \log \left(\sum_{i' \in \mathcal{A}} w_{i'} \, b_{i'j}\right). \end{aligned} \tag{4.8}$$

Substituting the value of v_{ij} from Equation (4.7) in equation (4.8) gives

$$\begin{aligned} & f_{\text{cvx}}(\mathbf{b}^\star, \mathbf{q}^\star) - f_{\text{cvx}}(\mathbf{b}, \mathbf{q}) \\ & \quad = \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{G}} (b^\star_{ij} - b_{ij}) \left(w_i \log \left(\sum_{i' \in \mathcal{A}} w_{i'} \, b^\star_{i'j}\right) + \lambda_i + w_i + \eta_j\right) \\ & \qquad - \sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} w_i \, b^\star_{ij} \log \left(\sum_{i' \in \mathcal{A}} w_{i'} \, b^\star_{i'j}\right) \end{aligned}$$

$$
\begin{aligned}
&+ \sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} w_i \, b_{ij} \log \left(\sum_{i' \in \mathcal{A}} w_{i'} \, b_{i'j} \right) \\
&= \sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} w_i \, b_{ij} \log \left(\frac{\sum_{i' \in \mathcal{A}} w_{i'} \, b_{i'j}}{\sum_{i' \in \mathcal{A}} w_{i'} \, b^\star_{i'j}} \right) \\
&+ \sum_{i \in \mathcal{A}} (\lambda_i + w_i) \left(\sum_{j \in \mathcal{G}} b^\star_{ij} - \sum_{j \in \mathcal{G}} b_{ij} \right) \\
&+ \sum_{j \in \mathcal{G}} \eta_j \left(\sum_{i \in \mathcal{A}} b^\star_{ij} - \sum_{i \in \mathcal{A}} b_{ij} \right).
\end{aligned}
$$

Using $\sum_{j \in \mathcal{G}} b_{ij} = \sum_{j \in \mathcal{G}} b^\star_{ij} = 1$ for every $i \in \mathcal{A}$, we get

$$
\begin{aligned}
&f_{\text{cvx}}(\mathbf{b}^\star, \mathbf{q}^\star) - f_{\text{cvx}}(\mathbf{b}, \mathbf{q}) \\
&\qquad = \sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} w_i \, b_{ij} \log \left(\frac{\sum_{i' \in \mathcal{A}} w_{i'} \, b_{i'j}}{\sum_{i' \in \mathcal{A}} w_{i'} \, b^\star_{i'j}} \right) + \sum_{j \in \mathcal{G}} \eta_j \left(q^\star_j - q_j \right),
\end{aligned}
$$

where the last equation follows from the definitions of q_j and $q^\star_j$.

Note that by complementary slackness, $\eta_j(1 - q^\star_j) = 0$ for any $j \in \mathcal{G}$. So if $q^\star_j < 1$, then $\eta_j = 0$ and therefore $\eta_j(q^\star_j - q_j) = 0$. If $q^\star_j = 1$, then by the hypothesis of the Lemma, $q_j = 1$, and again we obtain that $\eta_j(q^\star_j - q_j) = 0$. Using this bound in the above equation gives

$$
f_{\text{cvx}}(\mathbf{b}^\star, \mathbf{q}^\star) - f_{\text{cvx}}(\mathbf{b}, \mathbf{q}) = \sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} w_i \, b_{ij} \log \left(\frac{\sum_{i' \in \mathcal{A}} w_{i'} \, b_{i'j}}{\sum_{i' \in \mathcal{A}} w_{i'} \, b^\star_{i'j}} \right).
$$

□

Before proving Lemma 4.4.4, we need the following lemma about the feasibility of a solution when we decrease the b_{ij} for some edge (j, i) in the support forest of $\mathbf{b}$.

Lemma 4.4.5. *Let $(\mathbf{b}, \mathbf{q})$ be an acyclic feasible point in $\mathcal{P}(\mathcal{A}, \mathcal{G})$, and let F be a directed forest formed by $G_{\text{supp}}(\mathbf{b})$ when every tree is rooted at an arbitrary agent node. For a non-root agent i in F, let item j be its parent. Then, for any $0 \leq \delta \leq \min\{b_{ij},\ 1 - b_{ij}\}$, there exists a feasible solution, $(\mathbf{b}^\delta, \mathbf{q}^\delta)$ such that $b^\delta_{ij} = b_{ij} - \delta$, $q^\delta_j = q_j - \delta$, $q^\delta_{j'} \geq q_{j'}$ for all $j' \in \mathcal{G} \backslash \{j\}$, and*

$$
b^\delta_{i'j'} \begin{cases} \leq \min\{1,\ 2b_{i'j'}\} & \text{if } i', j' \in T(i) \\ = b_{i'j'} & \text{otherwise}, \end{cases}
$$

where $T(x)$ denotes the sub-tree rooted at x in F.

The proof of this lemma will be deferred to Appendix 4.6.1. For now, we use it to complete the proof of Lemma 4.4.4.

Proof of Lemma 4.4.4. We will iteratively build $(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}})$ so that it satisfies these properties while ensuring it remains feasible. For a vertex $x \in \mathcal{A} \cup \mathcal{G}$, let $\text{par}(x)$ denote its parent in $G_{\text{supp}}(\mathbf{b}^\star)$, let $C(x)$ denote the set of its children in $G_{\text{supp}}(\mathbf{b}^\star)$, and let $T(x)$ denote the sub-tree rooted at vertex x in $G_{\text{supp}}(\mathbf{b}^\star)$.

Consider an item j with $q_j^\star < 1/2$. To make the vertex corresponding to j a leaf, the algorithm removes all the edges between item j and its children $C(j)$. To reflect this change, we will update the solution $(\mathbf{b}^\star, \mathbf{q}^\star)$ to an intermediate solution $(\widetilde{\mathbf{b}}, \widetilde{\mathbf{q}})$ such that the support of $\widetilde{\mathbf{b}}$ does not contain any edges between item j and its children. To maintain feasibility, we require:

$$\begin{aligned} \widetilde{q}_j &= \widetilde{b}_{\text{par}(j)j} = b^\star_{\text{par}(j)j} \\ \widetilde{b}_{ij} &= 0 \text{ for all } i \in C(j) \end{aligned} \tag{4.9}$$

Note that $q_j^\star < 1/2$ implies $b_{ij}^\star < 1/2$ for each $i \in C(j)$. As a result, the decrease in b_{ij} satisfies

$$b_{ij}^\star - \widetilde{b}_{ij} \le \min\{b_{ij}^\star, 1 - b_{ij}^\star\}$$

for each $i \in C(j)$. So, we update $(\widetilde{\mathbf{b}}, \widetilde{\mathbf{q}})$ by iteratively applying Lemma 4.4.5 to edge $(j \to i)$ with $\delta = b_{ij}$ for each $i \in C(j)$. The updated solution satisfies $\widetilde{b}_{ij} = 0$ for each $i \in C(j)$ and $\widetilde{q}_j = q_j - \sum_{i \in C(j)} b_{ij} = b_{\text{par}(j)j} < 1/2$. Note that $T(j)$ is the disjoint union of the sub-trees rooted at nodes in $C(j)$. So for distinct $i_1, i_2 \in C(j)$, updating the edge $(j \to i_1)$ (and the sub-tree for i_1) does not affect the b values for any edge in $T(i_2)$ and vice versa. Therefore, by Lemma 4.4.5, we have $q_{j'}^\star \le \widetilde{q}_{j'}$ for any item $j' \in T(j)$ and $\widetilde{b}_{i'j'} \le \min\{1, 2b^\star_{i'j'}\}$ for any $i', j' \in T(j)$.

Since every item with $q_j^\star < 1/2$ must become a leaf, we repeat the above process for any such item. The following fact is crucial to bound the values after multiple pruning processes: Pruning item j only changes b values for edges in $T(j)$, and item j becomes a leaf after that. So, if we prune ancestors of j after pruning j, the b values of edges in $T(j)$ do not change further.

Let $(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}})$ be the solution obtained by pruning the set of items $J = \{j \in \mathcal{G} : q_j^\star < 1/2\}$ in decreasing order of their height[1]. Pruning item j does not decrease the q value of any item other than j. Therefore, if $q_j^{\text{pruned}} < 1/2$, then $q_j^\star < 1/2$, so item j has been pruned and is a leaf. For any item j with $q_j^\star \ge 1/2$, its q value only increases when its nearest ancestor is pruned, and this is the only time its q-value changes. So we conclude that $q_j^{\text{pruned}} \ge q_j^\star$ for each $j \in \mathcal{G}$.

To establish the third claim of the lemma, observe that the b-value of any edge in $G_{\text{supp}}(\mathbf{b}^\star)$ changes at most twice during the pruning process: If $q_j^\star \ge 1/2$, then item j itself is not pruned, and the b values of edges incident to j may change only when the nearest ancestor of j is pruned. By Lemma 4.4.5, $b_{ij}^{\text{pruned}} \le \min\{1,\ 2b_{ij}^\star\}$ for each $i \in \mathcal{A}$. If $q_j^\star < 1/2$, the b value of any edge from j to its children becomes zero when j is pruned, satisfying the claim. The b value of the edge $(\text{par}(j) \to j)$ does not change when we prune j, and it may increase when the nearest ancestor of j in J is pruned. If so, we have $b^{\text{pruned}}_{\text{par}(j)j} \le \min\{1,\ 2b^\star_{\text{par}(j)j}\}$. □

[1]Note that pruning items in decreasing order of their height is only an artifact of the analysis. The algorithm can prune items with $q_j^\star < 1/2$ in any order.

4.4.2 Fractional Matching and Analysis

In this section, we prove Lemma 4.4.2, which completes the proof of Theorem 4.1.6.

We establish Lemma 4.4.2 by proving two inequalities (in Lemmas 4.4.6 and 4.4.7) about the properties of f_{ncvx} at any feasible point whose support is a forest. Lemma 4.4.6 shows that f_{ncvx} can be upper bounded by a linear function in $\mathbf{b}$ while only losing a constant factor.

Lemma 4.4.6. *Let $(\mathbf{b}, \mathbf{q})$ be an acyclic solution in $\mathcal{P}(\mathcal{A}, \mathcal{G})$ such that every item with $q_j < 1/2$ is a leaf in $G_{\text{support}}(\mathbf{b})$. Let $S : \mathcal{A} \to 2^{\mathcal{G}}$ be a function such that for each agent i, $S(i)$ is a subset of leaf items connected to agent i in $G_{\text{supp}}(\mathbf{b})$, and $S(i)$ contains all children of agent i with $q_j < 1/2$. Then*

$$\sum_{i \in \mathcal{A}} w_i \left(\sum_{j \notin S(i)} b_{ij} \log v_{ij} + \sum_{j \in S(i)} b_{ij} \log \left(\sum_{j \in S(i)} v_{ij} \right) \right) \geq f_{\text{ncvx}}(\mathbf{b}, \mathbf{q}) - \log 2 - \frac{1}{2e}.$$

Lemma 4.4.7 demonstrates how the linear function obtained in Lemma 4.4.6 can be used as a lower bound for the maximum weight matching with the augmented weight function. A crucial component of the proof of this lemma is the fact that any feasible $\mathbf{b}$ in $\mathcal{P}(\mathcal{A}, \mathcal{G})$ corresponds to a point in the matching polytope where all agents are matched.

Lemma 4.4.7. *Let $(\mathbf{b}, \mathbf{q})$ be an acyclic solution in $\mathcal{P}(\mathcal{A}, \mathcal{G})$ such that every item with $q_j < 1/2$ is a leaf in $G_{\text{support}}(\mathbf{b})$. Let $S : \mathcal{A} \to 2^{\mathcal{G}}$ be a function such that for each agent i, $S(i)$ is a subset of leaf items connected to agent i in $G_{\text{supp}}(\mathbf{b})$, and $S(i)$ contains all children of agent i with $q_j < 1/2$. Then, there exists a matching M in $G_{\text{supp}}(\mathbf{b})$ between vertices in $\mathcal{A}$ and $\{\mathcal{G} \setminus \cup_i \{S(i)\}\}$ such that*

$$\sum_{i \in \mathcal{A}} w_i \log \left(v_{iM(i)} + \sum_{j \in S(i)} v_{ij} \right) \geq \sum_{i \in \mathcal{A}} w_i \left(\sum_{j \notin S(i)} b_{ij} \log v_{ij} + \sum_{j \in S(i)} b_{ij} \log \left(\sum_{j \in S(i)} v_{ij} \right) \right), \tag{4.10}$$

where $v_{iM(i)} = 0$ if agent i is not matched in M.

Lemma 4.4.6 and Lemma 4.4.7 together establish Lemma 4.4.2. In the rest of this section, we provide the proofs of Lemma 4.4.6 and Lemma 4.4.7.

The following lemma is an application of Gibbs's inequality and is used in the proof of Lemma 4.4.6.

Lemma 4.4.8. *Given positive reals $z_1, \ldots, z_d$, for any $y_1, y_2, \ldots, y_d \geq 0$,*

$$\sum_{j=1}^{d} y_j \log \left(\sum_{j=1}^{d} z_j \right) - \sum_{j=1}^{d} y_j \log \left(\sum_{j=1}^{d} y_j \right) \geq \sum_{j=1}^{d} y_j \log z_j - \sum_{j=1}^{d} y_j \log y_j.$$

Proof. Define vectors $\mathbf{y} = (y_1, \ldots, y_d)$ and $\mathbf{z} = (z_1, \ldots, z_d)$. Then $\widetilde{\mathbf{y}} = \frac{\mathbf{y}}{\|\mathbf{y}\|_1}$ and $\widetilde{\mathbf{z}} = \frac{\mathbf{z}}{\|\mathbf{z}\|_1}$ define two probability distributions on $[d]$. The inequality is equivalent to $D_{\mathrm{KL}}(\widetilde{\mathbf{y}} \,\|\, \widetilde{\mathbf{z}}) \geq 0$. □

Proof of Lemma 4.4.6. Let $S := \cup_i \{S(i)\}$. Recall that

$$\begin{aligned} f_{\mathrm{ncvx}}(\mathbf{b}, \mathbf{q}) &= \sum_{i \in \mathcal{A}} w_i \sum_{j \in \mathcal{G}} b_{ij} \log v_{ij} - \sum_{i \in \mathcal{A}} w_i \sum_{j \in \mathcal{G}} b_{ij} \log q_j \\ &= \sum_{i \in \mathcal{A}} w_i \sum_{j \notin S(i)} b_{ij} \log v_{ij} - \sum_{i \in \mathcal{A}} w_i \sum_{j \notin S(i)} b_{ij} \log q_j \\ &\quad + \sum_{i \in \mathcal{A}} w_i \sum_{j \in S(i)} \left(b_{ij} \log v_{ij} - b_{ij} \log b_{ij} \right), \end{aligned} \tag{4.11}$$

where the last equation follows from the fact that every item in $S(i)$ is a leaf, i.e., if $j \in S(i)$, then $b_{i'j} = 0$ for every $i' \neq i$.

For an item $j \notin S$, we have $q_j \geq 1/2$. As a result,

$$- \sum_{i \in \mathcal{A}} w_i \, b_{ij} \log q_j \leq \log 2 \sum_{i \in \mathcal{A}} w_i \, b_{ij}. \tag{4.12}$$

Plugging this bound into Equation (4.11) gives

$$\begin{aligned} f_{\mathrm{ncvx}}(\mathbf{b}, \mathbf{q}) \leq & \sum_{i \in \mathcal{A}} w_i \sum_{j \notin S(i)} b_{ij} \log v_{ij} + \log 2 \sum_{i \in \mathcal{A}} w_i \sum_{j \notin S(i)} b_{ij} \\ & + \sum_{i \in \mathcal{A}} w_i \left(\sum_{j \in S(i)} b_{ij} \log v_{ij} - b_{ij} \log b_{ij} \right). \end{aligned} \tag{4.13}$$

As $\mathbf{b} \in \mathcal{P}(\mathcal{A}, \mathcal{G})$, we have $\sum_{j \notin S(i)} b_{ij} = 1 - \sum_{j \in S(i)} b_{ij}$ for every agent i. Substituting this in Equation (4.13) yields

$$\begin{aligned} f_{\mathrm{ncvx}}(\mathbf{b}, \mathbf{q}) \leq & \sum_{i \in \mathcal{A}} w_i \sum_{j \notin S(i)} b_{ij} \log v_{ij} + \log 2 \sum_{i \in \mathcal{A}} w_i \\ & + \sum_{i \in \mathcal{A}} w_i \left(\sum_{j \in S(i)} b_{ij} \log v_{ij} - b_{ij} \log b_{ij} - b_{ij} \log 2 \right) \\ = & \sum_{i \in \mathcal{A}} w_i \sum_{j \notin S(i)} (b_{ij} \log v_{ij}) + \log 2 \\ & + \sum_{i \in \mathcal{A}} w_i \left(\sum_{j \in S(i)} b_{ij} \log v_{ij} - b_{ij} \log b_{ij} - b_{ij} \log 2 \right), \end{aligned} \tag{4.14}$$

where the last equation follows from $\sum_i w_i = 1$.

For each agent $i \in \mathcal{A}$, Corollary 4.4.8 implies that

$$\sum_{j \in S(i)} b_{ij} \log v_{ij} - b_{ij} \log b_{ij} \\ \leq \sum_{j \in S(i)} b_{ij} \log \left(\sum_{j \in S(i)} v_{ij} \right) - \sum_{j \in S(i)} b_{ij} \log \left(\sum_{j \in S(i)} b_{ij} \right).$$

So, for any agent i,

$$\begin{aligned} &\sum_{j \in S(i)} b_{ij} \log v_{ij} - b_{ij} \log b_{ij} - b_{ij} \log 2 \\ &\quad \leq \sum_{j \in S(i)} b_{ij} \log \left(\sum_{j \in S(i)} v_{ij} \right) - \sum_{j \in S(i)} b_{ij} \log \left(\sum_{j \in S(i)} b_{ij} \right) - \sum_{j \in S(i)} b_{ij} \log 2 \\ &\quad \leq \sum_{j \in S(i)} b_{ij} \log \left(\sum_{j \in S(i)} v_{ij} \right) + \frac{1}{2e}, \end{aligned} \tag{4.15}$$

where the last inequality follows from $-x \log(x) - x \log 2 \leq 1/(2e)$ for all $x \geq 0$ applied to $x = \sum_{j \in S(i)} b_{ij}$.

Substituting Equation (4.15) in equation (4.14), we get

$$\begin{aligned} f_{\text{ncvx}}(\mathbf{b}, \mathbf{q}) &\leq \sum_{i \in \mathcal{A}} w_i \sum_{j \notin S(i)} b_{ij} \log v_{ij} + \log 2 \\ &\quad + \sum_{i \in \mathcal{A}} w_i \left(\sum_{j \in S(i)} b_{ij} \log \left(\sum_{j \in S(i)} v_{ij} \right) + \frac{1}{2e} \right) \\ &= \sum_{i \in \mathcal{A}} w_i \left(\sum_{j \notin S(i)} b_{ij} \log v_{ij} + \sum_{j \in S(i)} b_{ij} \log \left(\sum_{j \in S(i)} v_{ij} \right) \right) \\ &\quad + \log 2 + \frac{1}{2e}, \end{aligned}$$

where the last inequality again follows from $\sum_{i \in \mathcal{A}} w_i = 1$. □

Proof of Lemma 4.4.7. In this proof, we will analyze a matching that either assigns the bundle $S(i)$ to an agent or a single item $j \notin \cup_i \{S(i)\}$. Observe that the algorithm clearly finds an assignment with a larger objective as

$$\log \left(v_{iM(i)} + \sum_{j \in S(i)} v_{ij} \right) \geq \max \left\{ \log v_{iM(i)}, \log \left(\sum_{j \in S(i)} v_{ij} \right) \right\}.$$

So, for each agent $i \in \mathcal{A}$, we create a new leaf item ℓ_i with $v_{i\ell_i} = \sum_{j \in S(i)} v_{ij}$ corresponding to the set of items in $S(i)$. Define $S := \cup_i \{S(i)\}$ and $\widetilde{\mathcal{G}} := \{\mathcal{G} \backslash S\} \cup \{\ell_i\}_{i \in \mathcal{A}}$. We show that the maximum weight matching in the bipartite graph $(\mathcal{A}, \widetilde{\mathcal{G}})$ suffices to prove the lemma. As the matching polytope is integral, it is enough to demonstrate the existence of a fractional matching of a large value.

Using $\mathbf{b}$, we define fractional assignment variables $\mathbf{x}$ as follows:

$$x_{ij} := b_{ij} \quad \forall i \in \mathcal{A}, j \in \{\mathcal{G} \backslash L\}$$
$$x_{i\ell_i} := \sum_{j \in S(i)} b_{ij} \quad \forall i \in \mathcal{A}.$$

The L.H.S. of Equation (4.10) can be stated in terms of $\mathbf{x}$ as

$$\sum_{i \in \mathcal{A}} w_i \left(\sum_{j \notin S(i)} b_{ij} \log v_{ij} + \sum_{j \in S(i)} b_{ij} \log \left(\sum_{j \in S(i)} v_{ij} \right) \right) = \sum_{i \in \mathcal{A}} \sum_{j \in \widetilde{\mathcal{G}}} x_{ij}\, w_i \log v_{ij}. \tag{4.16}$$

Observe that $\mathbf{x}$ lies in the convex hull of matchings between agents $\mathcal{A}$ and items $\widetilde{\mathcal{G}}$ in which every agent is matched as $\mathbf{x}$ satisfies the following properties:

$$\sum_{j \in \widetilde{\mathcal{G}}} x_{ij} = \sum_{j \notin S(i)} b_{ij} + \sum_{j \in S(i)} b_{ij} = 1 \quad \forall i \in \mathcal{A}$$
$$\sum_{i \in \mathcal{A}} x_{ij} \leq 1 \quad \forall j \in \widetilde{\mathcal{G}}.$$

Here, for item $j \notin S$, the second inequality is inherited from the feasibility of $\mathbf{b}$. The constraint for $\ell_{i'}$ for some $i' \in \mathcal{A}$ is implied by the constraint $\sum_{i \in \mathcal{A}} x_{ij} = x_{i'j} = \sum_{j \in S(i)} b_{ij} \leq \sum_{j \in \mathcal{G}} b_{ij} \leq 1$, where the last constraint again follows from the feasibility of $\mathbf{b}$.

Using the integrality of the matching polytope, there exists a matching $\widetilde{M} : \mathcal{A} \to \widetilde{\mathcal{G}}$ such that

$$\sum_{i \in \mathcal{A}} \sum_{j \in \widetilde{\mathcal{G}}} x_{ij}\, w_i \log v_{ij} \leq \sum_{i \in \mathcal{A}} w_i\, \log v_{i\widetilde{M}(i)}. \tag{4.17}$$

Now consider a matching $M : \mathcal{A} \to \mathcal{G}$ with $M(i) = \emptyset$ if $\widetilde{M}(i) = \ell_i$, and $M(i) = \widetilde{M}(i)$ otherwise. Then

$$\sum_{i \in \mathcal{A}} w_i\, \log v_{i\widetilde{M}(i)} \leq \sum_{i \in \mathcal{A}} w_i\, \log \left(v_{iM(i)} + \sum_{j \in S(i)} v_{ij} \right). \tag{4.18}$$

Then Equations (4.16), (4.17), and (4.18) together imply

$$\sum_{i \in \mathcal{A}} w_i\, \log \left(v_{iM(i)} + \sum_{j \in S(i)} v_{ij} \right)$$

$$\geq \sum_{i \in \mathcal{A}} w_i \left(\sum_{j \notin S(i)} b_{ij} \log v_{ij} + \sum_{j \in S(i)} b_{ij} \log \left(\sum_{j \in S(i)} v_{ij} \right) \right).$$

□

4.5 Conclusion and Future directions

This chapter shows the equivalence of two previously introduced convex relaxations for the unweighted Nash Social Welfare problem. We then introduce a convex and a non-convex relaxation for the weighted (asymmetric) Nash Social Welfare problem to give an $O(\exp(2D_{\mathrm{KL}}(\mathbf{w} \,\|\, \mathbf{u})))$-approximation in polynomial time, and a $O(\exp(D_{\mathrm{KL}}(\mathbf{w} \,\|\, \mathbf{u})))$-approximation in pseudo-polynomial time. Both of these relaxations play a crucial role in obtaining the approximation algorithm for the problem.

After the initial release of this work there has been substantial progress improving the approximation factor. For the case of additive [77] and even the more general submodular valuations [78] there are now constant factor approximation algorithms. Their approach is based on rounding a fractional solution to the configuration formulation of the problem. This formulation is distinct from those discussed in this work, but it can be shown that it is no weaker than LogConcave-Weighted. It would be interesting to from a complete study of the relationship between this new formulation and the other relaxations discussed in this work. It is important to emphasize that we lose the $\exp(D_{\mathrm{KL}}(\mathbf{w} \,\|\, \mathbf{u}))$ when relating the objectives of the two relaxations; we only lose a constant factor when rounding the non-convex relaxation. A direct approach may exist to solve the non-convex formulation that gives an improved approximation guarantee.

Another question is whether the techniques introduced in this work can be expanded to more general valuation functions, particularly submodular valuations for the weighted Nash Social Welfare problem.

4.6 Appendix for Chapter 4

4.6.1 The proof of Lemma 4.4.5

The following is an application of Farkas' Lemma, which will be used in the proof of Lemma 4.4.5.

Lemma 4.6.1. *Let $\alpha > 0$ and $\beta_1, \ldots, \beta_k > 0$ with $\alpha + \sum_{j=1}^{k} \beta_i = 1$. For any $0 < \delta \leq \min\{\alpha, 1-\alpha\}$, there exist real numbers $\delta_1, \ldots, \delta_k$ such that*

$$\begin{aligned} \alpha - \delta + \sum_{j \in [k]} \beta_j (1+\delta_j) &= 1 \\ \beta_j (1+\delta_j) &\leq 1 \quad \forall j \in [k] \\ 0 \leq \delta_j &\leq 1 \quad \forall j \in [k]. \end{aligned} \tag{4.19}$$

Proof. As the above system contains only linear constraints in δ, we use Farkas' Lemma to show the existence of $\{\delta_j\}_{j=1}^k$. Re-arranging the constraints gives

$$\begin{aligned} \sum_{j\in[k]} \beta_j \delta_j &= \delta \\ \beta_j \delta_j &\leq 1-\beta_j \quad \forall j \in [k] \\ 0 \leq \delta_j &\leq 1 \quad \forall j \in [k] \end{aligned} \tag{4.20}$$

If there do not exist real numbers $\{\delta_j\}_{j=1}^k$ satisfying (4.20), then by Farkas' Lemma, there exist real numbers $\eta, \{\gamma_j\}_{j=1}^k, \{\lambda_j\}_{j=1}^k$ such that

$$\beta_j \eta + \beta_j \gamma_j + \lambda_j \geq 0 \quad \forall j \in [k] \tag{4.21}$$

$$\gamma_j, \lambda_j \geq 0$$

$$\delta\eta + \sum_{j\in[k]} (1-\beta_j)\gamma_j + \sum_{j\in[k]} \lambda_j < 0 \tag{4.22}$$

Adding equation (4.21) for all $j \in [k]$, we get

$$\eta \sum_{j\in[k]} \beta_j + \sum_{j\in[k]} \beta_j \gamma_j + \sum_{j\in[k]} \lambda_j \geq 0.$$

Since $\alpha + \sum_{j\in[k]} \beta_j = 1$, this implies $\eta(1-\alpha) + \sum_{j\in[k]} \beta_j\gamma_j + \sum_{j\in[k]} \lambda_j \geq 0$. In addition, since $\beta_i > 0$, we also have $\alpha < 1$. Therefore, dividing by $1-\alpha$ and re-arranging gives

$$\sum_{j\in[k]} \frac{\beta_j \gamma_j}{1-\alpha} + \sum_{j\in[k]} \frac{\lambda_j}{1-\alpha} \geq -\eta. \tag{4.23}$$

On the other hand, Equation (4.22) implies

$$-\eta > \sum_{j\in[k]} \frac{(1-\beta_j)\gamma_j}{\delta} + \sum_{j\in[k]} \frac{\lambda_j}{\delta}. \tag{4.24}$$

On comparing Equations (4.23) and (4.24), we obtain

$$\sum_{j\in[k]} \frac{\beta_j \gamma_j}{1-\alpha} + \sum_{j\in[k]} \frac{\lambda_j}{1-\alpha} > \sum_{j\in[k]} \frac{(1-\beta_j)\gamma_j}{\delta} + \sum_{j\in[k]} \frac{\lambda_j}{\delta}. \tag{4.25}$$

We will now derive a contradiction to (4.25).

As $\delta \leq 1-\alpha$, we have $1/(1-\alpha) \leq 1/\delta$, and therefore,

$$\sum_{j\in[k]} \frac{\lambda_j}{1-\alpha} \leq \sum_{j\in[k]} \frac{\lambda_j}{\delta}\,, \tag{4.26}$$

where we use the fact that $\lambda_j > 0$ for all $j \in [k]$.

In addition, for any $j \in [k]$

$$\frac{\beta_j}{1-\alpha} - \frac{(1-\beta_j)}{\delta} \leq \frac{\beta_j}{1-\alpha} - \frac{(1-\beta_j)}{\alpha} = \frac{\alpha + \beta_j - 1}{\alpha(1-\alpha)} \leq 0. \tag{4.27}$$

Here, the first inequality follows from $\delta \leq \alpha$, and the last inequality follows from the facts that $\alpha + \sum_{j\in[k]} \beta_j = 1$ and $\alpha, \beta_j > 0$.

On adding Equation (4.26) and equation (4.27) for all $j \in [k]$, we obtain

$$\sum_{j\in[k]} \frac{\beta_j \gamma_j}{1-\alpha} + \sum_{j\in[k]} \frac{\lambda_j}{1-\alpha} \leq \sum_{j\in[k]} \frac{(1-\beta_j)\gamma_j}{\delta} + \sum_{j\in[k]} \frac{\lambda_j}{\delta},$$

which contradicts Equation (4.25). Therefore, there exist real numbers $\{\delta_j\}_{j=1}^k$ satisfying (4.19). □

With this result, we are ready to finally complete the proof of Lemma 4.4.5.

Proof of Lemma 4.4.5. For $x \in \mathcal{A} \cup \mathcal{G}$, let $C(x)$ denote the children of node x in F and let $T(x)$ denote the sub-tree rooted at node x. We will prove this lemma by induction on the height of agent i, building $(\mathbf{b}^\delta, \mathbf{q}^\delta) \in \mathcal{P}(\mathcal{A}, \mathcal{G})$ in the process.

For the base case, assume agent i has height 1, i.e., $T(i)$ consists of only leaf item nodes that are the children of node i. We define a new vector $\mathbf{b}^\delta$ with $b^\delta_{i'j'} = b_{i'j'}$ for any $i' \neq i$ and $j' \in \mathcal{G}$. Note that setting $b^\delta_{ij} = b_{ij} - \delta$ and $q^\delta_j = q_j - \delta$ only violates the Agent constraint for agent i. So we will update the values of $\mathbf{b}$ in $T(i)$ to make the solution feasible.

By the feasibility of $\mathbf{b}$, $b_{ij} + \sum_{k\in C(i)} b_{ik} = 1$, and for every item node $k \in C(i)$, $q_k = b_{ik} < 1$. Using Lemma 4.6.1 with $\alpha = b_{ij}$ and $\beta_k = b_{ik}$, there exist δ_k for each $k \in C(i)$ such that

$$\begin{aligned} b_{ij} - \delta + \sum_{k\in C(i)} b_{ik}(1+\delta_k) &= 1 \\ b_{ik}(1+\delta_k) &\leq 1 \quad \forall k \in C(i) \\ 0 \leq \delta_k &\leq 1 \quad \forall k \in C(i). \end{aligned}$$

So, for each $k \in C(i)$, we set $b^\delta_{ik} = b_{ik}(1+\delta_k)$. Note that $b^\delta_{ik} \leq 1$, and as $\delta_k \leq 1$, we have

$$b^\delta_{ik} = b_{ik}(1+\delta_k) \leq 2b_{ik}.$$

As every item in $C(i)$ is a leaf, we also have

$$q_k \leq q^\delta_k = b^\delta_{ik} = b_{ik}(1+\delta_k) \leq 1$$

for each item $k \in C(i)$. The Agent constraint for agent i satisfies

$$\sum_{k\in C(i)} b^\delta_{ik} = b_{ij} - \delta + \sum_{k\in C(i)} b_{ik}(1+\delta_k) = 1.$$

Therefore, $\mathbf{b}^\delta \in \mathcal{P}(\mathcal{A}, \mathcal{G})$ and $b^\delta_{i'j'} \leq \min\{1, 2b_{ij'}\}$ for each $j' \in T(i)$.

For the induction hypothesis, assume that the lemma is true whenever the height of agent i is at most $\ell - 1$ for some integer $\ell > 1$. We now show that the statement also holds when the height of agent i is ℓ.

Again, setting $b^\delta_{ij} = b_{ij} - \delta$ and $q^\delta_j = q_j - \delta$ violates the Agent constraint for agent i. Similar to the base case, we can find $\delta_k \in (0, 1)$ for each $k \in C(i)$ such that $b_{ik}(1 + \delta_k) \leq 1$ and

$$b_{ij} - \delta + \sum_{k \in C(i)} b_{ik}(1 + \delta_k) = 1.$$

Setting $b^\delta_{ik} = b_{ik}(1 + \delta_k)$ for each $k \in C(i)$ will ensure that $\mathbf{b}^\delta$ satisfies the Agent constraint for agent i. However, this can violate the Item constraint for some item $k \in C(i)$, as $q^\delta_k = q_k + \delta_k b_{ik}$. So, we inductively update the values of $\mathbf{b}^\delta$ and $\mathbf{q}^\delta$ for the sub-tree rooted at item k for which such a violation occurs.

Consider an item $k \in C(i)$ such that $q^\delta_k = q_k + \delta_k b_{ik} > 1$. So we decrease $b_{i'k}$ for each $i' \in C(k)$ to ensure that q^δ_k is at most 1 as follows. Define $\gamma := q_k + \delta_k b_{ik} - 1$. Using the fact that $q_k = \sum_{i' \in C(k)} b_{i'k} + b_{ik}$, we bound γ as follows.

$$\begin{aligned}\gamma = q_k + \delta_k b_{ik} - 1 &= \sum_{i' \in C(k)} b_{i'k} + b_{ik} + \delta_k b_{ik} - 1 \\ &\leq \sum_{i' \in C(k)} b_{i'k},\end{aligned}$$

using $b_{ik}(1 + \delta_k) \leq 1$ Therefore, there exist numbers $\gamma_{i'} \geq 0$ for each $i' \in C(k)$ such that $\gamma_{i'} \leq b_{i'k}$ and $\sum_{i' \in C(k)} \gamma_{i'} = \gamma$.

We would like to update $b^\delta_{i'k} = b_{i'k} - \gamma_{i'}$ for each $i' \in C(k)$, but this violates the Agent constraint for agent i' when $\gamma_{i'} > 0$. We inductively update the solution for subtree $T(i')$ as follows.

First, note that $q^\delta_k = 1 \geq q_k$ after this update, as shown below.

$$\begin{aligned}q^\delta_k = b^\delta_{ik} + \sum_{i' \in C(k)} b^\delta_{i'k} &= b_{ik}(1 + \delta_k) + \sum_{i' \in C(k)} (b_{i'k} - \gamma_{i'}) = q_k + \delta_i b_{ik} - \sum_{i' \in C(k)} \gamma_{i'} \\ &= 1 - \gamma + \sum_{i' \in C(k)} \gamma_{i'} \\ &= 1,\end{aligned}$$

as $\sum_{i' \in C(k)} \gamma_{i'} = \gamma$ So now, $(\mathbf{b}^\delta, \mathbf{q}^\delta)$ only violates Agent constraints for agents in $C(k)$.

We claim that for each agent $i' \in C(k)$

$$\gamma_{i'} \leq \min\{b_{i'k},\ 1 - b_{i'k}\}. \tag{4.28}$$

Before proving this inequality, we use it to complete the proof.

Using the induction hypothesis, for each $i' \in C(k)$, there exists feasible $(\mathbf{b}^{\gamma_{i'}}, \mathbf{q}^{\gamma_{i'}})$ which differs from $(\mathbf{b}^{\delta}, \mathbf{q}^{\delta})$ only in the sub-tree rooted at i' such that for any $\hat{j} \in T(i')$,

$$q_{\hat{j}}^{\gamma_{i'}} \geq q_{\hat{j}}^{\delta} = q_{\hat{j}}.$$

and for any $\hat{i}, \hat{j} \in T(i')$,

$$b_{\hat{i}\hat{j}}^{\gamma_{i'}} \leq \min\{1, 2 \cdot b_{\hat{i},\hat{j}}^{\delta}\} = \min\{1, 2 \cdot b_{\hat{i},\hat{j}}\}.$$

So for each $i' \in C(k)$ with $\gamma_{i'} > 0$, we set $b_{\hat{i}\hat{j}}^{\delta} = b_{\hat{i}\hat{j}}^{\gamma_{i'}}$ for every $\hat{i}, \hat{j} \in T(i')$ to get the required solution.

We now only need to establish Equation (4.28). By definition, $\gamma_{i'} \leq b_{i'k}$ for each $i' \in C(k)$. Additionally, $\gamma_{i'} \leq \gamma$, so it suffices to show that $\gamma \leq 1 - b_{i'k}$ for every $i' \in C(k)$. Recall that

$$\begin{aligned}\gamma &= q_k + \delta_k b_{ik} - 1 \\ &\overset{(i)}{\leq} \delta_k b_{ik} \overset{(ii)}{\leq} b_{ik} \overset{(iii)}{\leq} q_k - b_{i'k} \overset{(iv)}{\leq} b_{i'k}.\end{aligned}$$

Here, (i) and (iv) follow from $q_k \leq 1$, (ii) follows from $\delta_k \leq 1$, and (iii) holds as $b_{ik} + \sum_{i' \in C(k)} b_{i'k} = q_k$. This completes the proof of (4.28). □

4.6.2 Relationships Between the Mathematical Programs

This section provides the proof of Theorem 4.1.2 by establishing a relationship between two natural convex programming relaxations for the unweighted Nash Social Welfare problem. We then build upon this relationship to derive (CVX-Weighted) for the weighted Nash Social Welfare problem.

To ensure that the optimum values of all the convex programs mentioned below are bounded, we assume that the instance of Nash Social Welfare $(\mathcal{A}, \mathcal{G}, \mathbf{v}, \mathbf{w})$ satisfies the following assumption.

Assumption Let $G[\mathcal{G}, \mathcal{A}, \mathbf{v}]$ denote the support graph of the valuation function. The support graph is the bipartite graph between agents and items with an edge between agent i and item j iff $v_{ij} > 0$. We assume that there exists a matching of size $|\mathcal{A}|$ in $G[\mathcal{G}, \mathcal{A}, \mathbf{v}]$. In other words, the objective of the Nash Social Welfare problem is not zero for $(\mathcal{A}, \mathcal{G}, \mathbf{v}, \mathbf{w})$. It is straightforward to verify this assumption given an instance of Nash Social Welfare.

The proof of Theorem 4.1.2 uses the following two results. The first result is the classical Sion's Minimax Theorem, which can be found as Corollary 3.3 from [175].

Theorem 4.6.2 (Sion's Minimax Theorem)**.** *Let M and N be convex spaces, one of which is compact, and $f(x, y)$ a function on $M \times N$ that is quasi-concave-convex and (upper semicontinuous) - (lower semicontinuous). Then*

$$\sup_{\mathbf{x} \in M} \inf_{y \in N} f(x, y) = \inf_{y \in N} \sup_{\mathbf{x} \in M} f(x, y).$$

The second result was proved in [7].

Lemma 4.6.3 (Lemma 4.3 in [7])**.** *Let $p : \mathbb{R}^m_{\geq 0} \to \mathbb{R}_{\geq 0}$ be a positive function satisfying the following properties:*

- $p(t \cdot \mathbf{y}) = t^n \cdot p(\mathbf{y})$ *for all* $\mathbf{y} \geq 0$ *and* $t \in \mathbb{R}$*,*
- $\log p(\mathbf{y})$ *is convex in* $\log \mathbf{y}$*.*

Then the following inequality holds

$$\inf_{\mathbf{y}>0: y^S \geq 1, \forall S \in \binom{[m]}{n}} \log p(\mathbf{y}) = \sup_{\boldsymbol{\alpha} \in [0,1]^m, \sum_j \alpha_j = n} \inf_{\mathbf{y}>0} \quad \log p(\mathbf{y}) - \sum_{j=1}^{m} \alpha_j \log(y_j).$$

While the original result in [7] assumed p to be a homogeneous polynomial with positive coefficients, their proof only relies on the two properties presented in Lemma 4.6.3.

Proof of Theorem 4.1.2

To prove Theorem 4.1.2, we start with the (LogConcave-Unweighted) and derive the convex program (CVX-Unweighted) via a sequence of duals presented in Lemmas 4.6.5, 4.6.6, and 4.6.7.

Let $\mathcal{P}$ and $\mathcal{Q}$ denote the feasible regions for $\mathbf{x}$ and $\mathbf{y}$ in (LogConcave-Unweighted), respectively.

$$\mathcal{P} := \left\{ \mathbf{x} \in \mathbb{R}^{\mathcal{A} \times \mathcal{G}}_{\geq 0} \; : \; \sum_{i \in \mathcal{A}} x_{ij} = 1 \quad \forall j \in \mathcal{G} \right\}$$

$$\mathcal{Q} := \left\{ \mathbf{y} \in \mathbb{R}^{\mathcal{G}}_{>0} \; : \; \prod_{j \in S} y_j \geq 1 \quad \forall S \in \binom{\mathcal{G}}{n} \right\}.$$

Note that the inner function in the objective

$$f(x) = \inf_{\mathbf{y} \in \mathcal{Q}} \sum_{i \in \mathcal{A}} \log \left(\sum_{j \in \mathcal{G}} x_{ij} \; v_{ij} \; y_j \right),$$

is bounded above ($\mathbf{y} = \mathbf{1}$ belongs to $\mathcal{Q}$), and the domain of x, $\mathcal{P}$, is compact (bounded and closed sets in Euclidean space are compact using Heine-Borel Theorem).

Lemma 4.6.4 shows that the inner infimum of (LogConcave-Unweighted) is $> -\infty$ for any integral allocation $\mathbf{x}$ that assigns at least one item to each agent in the support of $\mathbf{v}$. We know such an allocation exists by Assumption 4.6.2.

Lemma 4.6.4. *For any integral allocation $\mathbf{x} \in \mathcal{P} \cap \{0,1\}^{|\mathcal{A}| \times |\mathcal{G}|}$,*

$$\inf_{\mathbf{y} \in \mathcal{Q}} \quad \sum_{i \in \mathcal{A}} \log \left(\sum_{j \in \mathcal{G}} x_{ij} \, v_{ij} \, y_j \right) = \sum_{i \in \mathcal{A}} \log \left(\sum_{j \in \mathcal{G}} x_{ij} \, v_{ij} \right).$$

Proof. Let $\sigma : \mathcal{G} \to \mathcal{A}$ be the allocation corresponding to $\mathbf{x}$, i.e., $\sigma(j) = i$ iff $x_{ij} = 1$. Then this is a restatement of Lemma 4.2.2. □

Lemma 4.6.5. *The optimal value of* (LogConcave-Unweighted) *is the same as*

$$\inf_{\boldsymbol{\delta}} \max_{\mathbf{x}\in\mathcal{P}} \sum_{i\in\mathcal{A}} \log\left(\sum_{j\in\mathcal{G}} x_{ij}\, v_{ij}\, e^{-\delta_j}\right) + \sum_{j\in\mathcal{G}} \max(0,\, \delta_j). \tag{Unweighted-Primal}$$

Proof. For a fixed $\mathbf{x} \in \mathcal{P}$, using Lemma 4.6.3 with $p_{\mathbf{x}}(\mathbf{y}) = \prod_{i\in\mathcal{A}} \left(\sum_{j\in\mathcal{G}} x_{ij}\, v_{ij}\, y_j\right)$, we get

$$\begin{aligned}
&\inf_{\mathbf{y}>0: y^S \geq 0, \forall S \in \binom{\mathcal{G}}{n}} \log p_{\mathbf{x}}(\mathbf{y}) \\
&\qquad = \inf_{\mathbf{y}>0: y^S \geq 1, \forall S \in \binom{\mathcal{G}}{n}} \sum_{i\in\mathcal{A}} \log\left(\sum_{j\in\mathcal{G}} x_{ij}\, v_{ij}\, y_j\right) \\
&\qquad = \sup_{\boldsymbol{\alpha}\in[0,1]^{|\mathcal{G}|}, \sum_j \alpha_j = n} \inf_{\mathbf{y}>0} \sum_{i\in\mathcal{A}} \log\left(\sum_{j\in\mathcal{G}} x_{ij}\, v_{ij}\, y_j\right) - \sum_{j\in\mathcal{G}} \alpha_j \log(y_j).
\end{aligned}$$

Substituting $\delta_j = -\log(y_j)$ and taking a maximum over $\mathbf{x}$, we get

$$\begin{aligned}
&\max_{\mathbf{x}\in\mathcal{P}} \inf_{\mathbf{y}>0: y^S \geq 0, \forall S \in \binom{\mathcal{G}}{n}} \log p_{\mathbf{x}}(\mathbf{y}) \\
&\qquad = \sup_{\mathbf{x}\in\mathcal{P}, \boldsymbol{\alpha}\in[0,1]^{|\mathcal{G}|}, \sum_j \alpha_j = n} \inf_{\boldsymbol{\delta}} \sum_{i\in\mathcal{A}} \log\left(\sum_{j\in\mathcal{G}} x_{ij}\, v_{ij}\, e^{-\delta_j}\right) + \sum_{j\in\mathcal{G}} \alpha_j \delta_j.
\end{aligned}$$

As the domains of both $\mathbf{x}$ and $\boldsymbol{\alpha}$ are compact, and we have only added a linear function of α and γ to our original convex function, we can apply Theorem 4.6.2 to conclude that

$$\begin{aligned}
&\max_{\mathbf{x}\in\mathcal{P}} \inf_{\mathbf{y}>0: y^S \geq 0, \forall S \in \binom{\mathcal{G}}{n}} \log p_{\mathbf{x}}(\mathbf{y}) \\
&\qquad = \inf_{\boldsymbol{\delta}} \max_{\mathbf{x}\in\mathcal{P}} \max_{\boldsymbol{\alpha}\in[0,1]^{|\mathcal{G}|}, \sum_j \alpha_j = n} \sum_{i\in\mathcal{A}} \log\left(\sum_{j\in\mathcal{G}} x_{ij}\, v_{ij}\, e^{-\delta_j}\right) + \sum_{j\in\mathcal{G}} \alpha_j \delta_j.
\end{aligned}$$

Finally, the following claim completes the proof.

$$\begin{aligned}
&\inf_{\boldsymbol{\delta}} \max_{\mathbf{x}\in\mathcal{P}} \max_{\boldsymbol{\alpha}\in[0,1]^{|\mathcal{G}|}, \sum_j \alpha_j = n} \sum_{i\in\mathcal{A}} \log\left(\sum_{j\in\mathcal{G}} x_{ij}\, v_{ij}\, e^{-\delta_j}\right) + \sum_{j\in\mathcal{G}} \alpha_j \delta_j \\
&= \inf_{\boldsymbol{\delta}} \max_{\mathbf{x}\in\mathcal{P}} \sum_{i\in\mathcal{A}} \log\left(\sum_{j\in\mathcal{G}} x_{ij} v_{ij} e^{-\delta_j}\right) + \sum_{j\in\mathcal{G}} \max(0, \delta_j).
\end{aligned} \tag{4.29}$$

For proving the claim, we define functions

$$f_1(\boldsymbol{\delta}, \mathbf{x}, \boldsymbol{\alpha}) = \sum_{i \in \mathcal{A}} \log \left(\sum_{j \in \mathcal{G}} x_{ij}\, v_{ij}\, e^{-\delta_j} \right) + \sum_{j \in \mathcal{G}} \alpha_j \delta_j, \quad \text{and}$$

$$f_2(\boldsymbol{\delta}, \mathbf{x}) = \sum_{i \in \mathcal{A}} \log \left(\sum_{j \in \mathcal{G}} x_{ij} v_{ij} e^{-\delta_j} \right) + \sum_{j \in \mathcal{G}} \max(0, \delta_j).$$

Observe that for any $\boldsymbol{\delta}$ and $\boldsymbol{\alpha} \in [0,1]^{|G|}$, $\alpha_j \delta_j \leq \max(0, \delta_j)$. Therefore, for any $\boldsymbol{\delta}, \mathbf{x}$ and $\boldsymbol{\alpha} \in [0,1]^{|G|}$, we have $f_1(\boldsymbol{\delta}, \mathbf{x}, \boldsymbol{\alpha}) \leq f_2(\boldsymbol{\delta}, \mathbf{x})$. As a result,

$$\inf_{\boldsymbol{\delta}} \max_{\mathbf{x} \in \mathcal{P}} \max_{\boldsymbol{\alpha} \in [0,1]^{|\mathcal{G}|}, \sum_j \alpha_j = n} f_1(\boldsymbol{\delta}, \mathbf{x}, \boldsymbol{\alpha}) \leq \inf_{\boldsymbol{\delta}} \max_{\mathbf{x} \in \mathcal{P}} f_2(\boldsymbol{\delta}, \mathbf{x}). \tag{4.30}$$

To establish an inequality in the other direction, first note that $f_1(\boldsymbol{\delta}, \mathbf{x}, \boldsymbol{\alpha}) = f_1(\boldsymbol{\delta} + t \cdot \mathbf{1}, \mathbf{x}, \boldsymbol{\alpha})$ for any $t \in \mathbb{R}$. So, for a fixed $\boldsymbol{\delta}$, let t_δ denote a value of t for which the n largest values of $\boldsymbol{\delta} + t_\delta \cdot \mathbf{1}$ are non-negative and the $m - n$ smallest values of δ are non-positive. Then

$$\begin{aligned} \max_{\boldsymbol{\alpha} \in [0,1]^{|\mathcal{G}|}, \sum_j \alpha_j = n} f_1(\boldsymbol{\delta}, \mathbf{x}, \boldsymbol{\alpha}) &= \max_{\boldsymbol{\alpha} \in [0,1]^{|\mathcal{G}|}, \sum_j \alpha_j = n} f_1(\boldsymbol{\delta} + t_\delta \cdot \mathbf{1}, \mathbf{x}, \boldsymbol{\alpha}) \\ &= \sum_{i \in \mathcal{A}} \log \left(\sum_{j \in G} x_{ij} v_{ij} e^{-\delta_j - t_\delta} \right) \\ &\quad + \max_{\boldsymbol{\alpha} \in [0,1]^{|\mathcal{G}|}, \sum_j \alpha_j = n} \sum_{j \in \mathcal{G}} \alpha_j (\delta_j + t_\delta). \end{aligned} \tag{4.31}$$

The term $\sum_{j \in \mathcal{G}} \alpha_j (\delta_j + t_\delta)$ is maximized when $\alpha_j = 1$ for the largest n coordinates of $\boldsymbol{\delta} + t \cdot \mathbf{1}$. As a result, we get

$$\sum_{i \in \mathcal{A}} \log \left(\sum_{j \in G} x_{ij} v_{ij} e^{-\delta_j - t_\delta} \right) + \sum_{j \in \mathcal{G}} \max(0, \delta_j + t_\delta) = f_2(\boldsymbol{\delta} + t_\delta \cdot \mathbf{1}, \mathbf{x}). \tag{4.32}$$

Combining Equations (4.31) and (4.32), and taking max over $\mathbf{x}$, we have

$$\max_{\mathbf{x} \in \mathcal{P}} \max_{\boldsymbol{\alpha} \in [0,1]^{|\mathcal{G}|}, \sum_j \alpha_j = n} f_1(\boldsymbol{\delta}, \mathbf{x}, \boldsymbol{\alpha}) = \max_{\mathbf{x} \in \mathcal{P}} f_2(\boldsymbol{\delta} + t_\delta \cdot \mathbf{1}, \mathbf{x}) \geq \inf_{\boldsymbol{\gamma}} \max_{\mathbf{x} \in \mathcal{P}} f_2(\boldsymbol{\gamma}, \mathbf{x}).$$

Taking an infimum over $\boldsymbol{\delta}$, we obtain

$$\begin{aligned} \inf_{\boldsymbol{\delta}} \max_{\mathbf{x} \in \mathcal{P}} \max_{\boldsymbol{\alpha} \in [0,1]^{|\mathcal{G}|}, \sum_j \alpha_j = n} f_1(\boldsymbol{\delta}, \mathbf{x}, \boldsymbol{\alpha}) &\geq \inf_{\boldsymbol{\delta}} \inf_{\boldsymbol{\gamma}} \max_{\mathbf{x} \in \mathcal{P}} f_2(\boldsymbol{\gamma}, \mathbf{x}) \\ &= \inf_{\boldsymbol{\gamma}} \max_{\mathbf{x} \in \mathcal{P}} f_2(\boldsymbol{\gamma}, \mathbf{x}). \end{aligned} \tag{4.33}$$

Here, the last equality follows as the function being optimized does not depend on $\boldsymbol{\delta}$.

Combining Equations (4.30) and (4.33) completes the proof of equation (4.29). □

Lemma 4.6.6. *The optimal values of* (Unweighted-Primal) *is the same as that of the following program.*

$$\begin{aligned} \inf_{\boldsymbol{\delta},\mathbf{r},\gamma} \quad & \sum_{j\in\mathcal{G}} e^{r_j} + \sum_{i\in\mathcal{A}} \gamma_i + \sum_{j\in\mathcal{G}} \delta_j - n \\ & r_j + \gamma_i + \delta_j \geq \log v_{ij} \quad \forall (i,j) \in \mathcal{A}\times\mathcal{G} \\ & \boldsymbol{\delta} \geq \mathbf{0}. \end{aligned} \tag{Unweighted-Dual}$$

Proof. For a fixed $\boldsymbol{\delta}$, let us first re-write the internal maximum of (Unweighted-Primal) as

$$\begin{aligned} \max_{\mathbf{x},\mathbf{u}} \quad & \sum_{i\in\mathcal{A}} \log u_i + f(\boldsymbol{\delta}) \\ & u_i \leq \sum_{j\in\mathcal{G}} x_{ij}\, v_{ij}\, e^{-\delta_j} \quad \forall i \in \mathcal{A} \\ & \sum_{i\in\mathcal{A}} x_{ij} \leq 1 \quad \forall j \in \mathcal{G} \\ & \mathbf{x} \geq \mathbf{0}\,, \end{aligned} \tag{4.34}$$

where $f(\boldsymbol{\delta}) = \sum_{j\in\mathcal{G}} \max(0,\delta_j)$.

Let β_i, p_j, and θ_{ij} be the Lagrange dual variables associated with the constraints corresponding to agent i, item j, and agent-item pair(i,j), respectively. The Lagrangian of the above convex program is defined as follows

$$\begin{aligned} & L(\mathbf{x},\mathbf{u},\boldsymbol{\beta},\boldsymbol{\theta},\mathbf{p}) \\ & = f(\boldsymbol{\delta}) + \left[\sum_{i\in\mathcal{A}} \log u_i + \sum_{i\in\mathcal{A}} \beta_i \left(\sum_{j\in\mathcal{G}} x_{ij}\, v_{ij}\, e^{-\delta_j} - u_i\right) + \sum_{j\in\mathcal{G}} p_j (1 - \sum_{i\in\mathcal{A}} x_{ij}) + \sum_{i,j} \theta_{ij} x_{ij}\right] \\ & = f(\boldsymbol{\delta}) + \left[\sum_{i\in\mathcal{A}} (\log u_i - \beta_i u_i) + \sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{G}} x_{ij}\left(\beta_i\, v_{ij}\, e^{-\delta_j} + \theta_{ij} - p_j\right) + \sum_{j\in\mathcal{G}} p_j\right]. \end{aligned}$$

The Lagrange dual of (4.34) is given by

$$g(\boldsymbol{\beta},\boldsymbol{\theta},\boldsymbol{p}) = \max_{\mathbf{x}\in\mathcal{P},\mathbf{u}\geq 0} L(\mathbf{x},\mathbf{u},\boldsymbol{\beta},\boldsymbol{\theta},\mathbf{p}). \tag{4.35}$$

Observe that solution $x_{ij} = 1/n$ for each $(i,j) \in \mathcal{A}\times\mathcal{G}$ lies in the relative interior of $\mathcal{P}$. Since all the constraints are affine, Slater's condition is satisfied for (Unweighted-Primal). Thus, the optimal value of the infimum of Lagrange dual over $\boldsymbol{\beta},\boldsymbol{\theta},\boldsymbol{p} \geq 0$ is exactly equal to the optimum of (4.34).

The KKT conditions imply that the optimal solutions must satisfy

$$\begin{aligned} \frac{1}{u_i} - \beta_i = 0 \quad & \forall i \in \mathcal{A} \\ \beta_i\, v_{ij}\, e^{-\delta_j} - p_j + \theta_{ij} = 0 \quad & \forall (i,j) \in \mathcal{A}\times\mathcal{G}. \end{aligned}$$

The KKT conditions imply that $u_i = 1/\beta_i$ for each $i \in \mathcal{A}$ maximizes the Lagrangian. For the supremum over $\mathbf{x}, \mathbf{u}$ in (4.35) to stay finite, the second KKT condition is necessary and sufficient. Substituting these conditions in the Langrangian gives the following convex program.

$$\begin{aligned} \inf_{\mathbf{p},\boldsymbol{\beta},\boldsymbol{\theta}} \quad & f(\boldsymbol{\delta}) + \sum_{j\in\mathcal{G}} p_j - \sum_{i\in\mathcal{A}} \log \beta_i - n \\ & p_j = \beta_i \, v_{ij} \, e^{-\delta_j} + \theta_{ij} \quad \forall (i,j) \in \mathcal{A} \times \mathcal{G} \\ & \mathbf{p}, \boldsymbol{\beta}, \boldsymbol{\theta} \geq \mathbf{0}. \end{aligned}$$

Observe that we can remove $\boldsymbol{\theta}$ from the above program while making the first constraint an inequality. By substituting $r_j = \log p_j, \gamma_i = -\log \beta_i$, the above program is equivalent to

$$\begin{aligned} \inf_{\mathbf{r},\boldsymbol{\gamma}} \quad & f(\boldsymbol{\delta}) + \sum_{j\in\mathcal{G}} e^{r_j} + \sum_{i\in\mathcal{A}} \gamma_i + \sum_{j\in\mathcal{G}} -n \\ & r_j + \gamma_i + \delta_j \geq \log v_{ij} \quad \forall (i,j) \in \mathcal{A} \times \mathcal{G}. \end{aligned}$$

As (Unweighted-Primal) involves an infimum over $\boldsymbol{\delta}$, whenever $\delta_j < 0$, we can increase it to $\delta_j = 0$ without increasing the value of $f(\boldsymbol{\delta})$ and maintaining feasibility. Using this observation and taking an infimum over $\boldsymbol{\delta}$, the above program gives (Unweighted-Dual). □

Lemma 4.6.7. *The optimal values of* (Unweighted-Dual) *and* (CVX-Unweighted) *are the same.*

Proof. Let b_{ij} be the Lagrange dual variable associated with constraint $r_j + \gamma_i + \delta_j \geq \log v_{ij}$ of (Unweighted-Dual) and let τ_{ij} be the Lagrange dual variable associated with constraint $\delta_{ij} \geq 0$. The Lagrangian of (Unweighted-Dual) is defined as follows

$$\begin{aligned} L(\mathbf{r}, \boldsymbol{\gamma}, \boldsymbol{\delta}, \mathbf{b}, \boldsymbol{\tau}) &= \sum_{j\in\mathcal{G}} e^{r_j} + \sum_{i\in\mathcal{A}} \gamma_i + \sum_{j\in\mathcal{G}} \delta_j - n + \sum_{i,j} b_{ij}(\log v_{ij} - r_j - \gamma_i - \delta_j) - \sum_{j\in\mathcal{G}} \delta_j \tau_j \\ &= \sum_{j\in\mathcal{G}} (e^{r_j} - (\sum_{i\in\mathcal{A}} b_{ij}) r_j) + \sum_{i\in\mathcal{A}} \gamma_i (1 - \sum_{j\in\mathcal{G}} b_{ij}) \sum_{j\in\mathcal{G}} \delta_j (1 - \tau_j - \sum_{i\in\mathcal{A}} b_{ij}) \\ &\quad + \sum_{i,j} b_{ij} \log v_{ij} - n. \end{aligned}$$

The Lagrange dual of (Unweighted-Dual) is given by

$$g(\mathbf{b}, \boldsymbol{\tau}) = \inf_{\boldsymbol{\delta}\geq 0, \mathbf{r}, \boldsymbol{\gamma}} L(\mathbf{r}, \boldsymbol{\gamma}, \boldsymbol{\delta}, \mathbf{b}, \boldsymbol{\tau}). \tag{4.36}$$

One can verify that Slater's condition is satisfied by (Unweighted-Dual). So, the supremum of (4.36) with $\mathbf{b}, \boldsymbol{\tau} \geq 0$ is equal to the optimum of (Unweighted-Dual).

The KKT conditions for the Langrangian give

$$e^{r_j} - \sum_{i\in\mathcal{A}} b_{ij} = 0 \qquad 1 - \sum_{j\in\mathcal{G}} b_{ij} = 0 \qquad 1 - \tau_j - \sum_{j\in A} b_{ij} = 0.$$

The KKT conditions imply $r_j = \log\left(\sum_{i\in\mathcal{A}} b_{ij}\right)$ for each $j \in \mathcal{G}$ minimizes the Lagrangian. For the infimum over $\boldsymbol{\gamma}, \boldsymbol{\delta}$ in (4.36) to stay finite, the conditions $1 = \sum_{j\in\mathcal{G}} b_{ij}$ and $1 - \tau_j = \sum_{i\in\mathcal{A}} b_{ij}$ are necessary and sufficient. Substituting these conditions in the Lagrangian, we get

$$\begin{aligned}
\sup_{\mathbf{b},\boldsymbol{\tau}} \quad & \sum_{i,j} b_{ij} \log v_{ij} - \sum_{j\in\mathcal{G}}\sum_{i\in\mathcal{A}} b_{ij} \log\left(\sum_{i'\in\mathcal{A}} b_{i'j}\right) + \sum_{j\in\mathcal{G}}\sum_{i\in\mathcal{A}} b_{ij} - n \\
& \sum_{j\in\mathcal{G}} b_{ij} = 1 \\
& \sum_{i\in\mathcal{A}} b_{ij} = 1 - \tau_j \\
& \mathbf{b}, \boldsymbol{\tau} \geq \mathbf{0}.
\end{aligned}$$

Observe that the supremum in the above program can be switched to maximum as the feasible region is compact and the objective is bounded. Also note that $\sum_{i,j} b_{ij} = n$ for any $\mathbf{b}$ in the feasible region. As a result, the last two terms in the objective cancel each other. Finally, on substituting $q_j = \sum_{i\in\mathcal{A}} b_{ij}$ in the above program, we obtain (CVX-Unweighted). □

Generalization to Weighted Nash Social Welfare

Given an instance of weighted Nash Social Welfare $(\mathcal{A}, \mathcal{G}, \mathbf{v}, \mathbf{w})$ where $\sum_{i\in\mathcal{A}} w_i = 1$ and $\mathbf{w} \geq \mathbf{0}$, we introduce the following program as a generalization of (LogConcave-Unweighted) program.

$$\begin{aligned}
\max_{\mathbf{x}\geq 0} \min_{\mathbf{y}>0} \quad & \sum_{i\in\mathcal{A}} w_i \log\left(\sum_{j\in\mathcal{G}} x_{ij}\, v_{ij}\, y_j^{1/w_i}\right) && \text{(LogConcave-Weighted)} \\
\text{s.t.} \quad & \sum_{i\in\mathcal{A}} x_{ij} = 1 \quad \forall j \in \mathcal{G} \\
& \prod_{j\in S} y_j \geq 1 \quad \forall S \in \binom{\mathcal{G}}{n}.
\end{aligned}$$

Observe that the feasible region of (LogConcave-Weighted) is given by $\mathbf{x} \in \mathcal{P}$ and $\mathbf{y} \in \mathcal{Q}$, which is identical to that of (LogConcave-Unweighted).

The main result of this section is the following.

Theorem 4.6.8. *The optimal values of* (LogConcave-Weighted) *and* (CVX-Weighted) *are the same.*

We prove Theorem 4.6.8 analogously to Theorem 4.1.2, starting with (LogConcave-Weighted) and deriving (CVX-Weighted) via a sequence of duals presented in Lemmas 4.6.9, 4.6.11, and 4.6.12.

Lemma 4.6.9. *The optimal value of* (LogConcave-Weighted) *is the same as*

$$\inf_{\boldsymbol{\delta}} \max_{\mathbf{x}\in\mathcal{P}} \sum_{i\in\mathcal{A}} w_i \log\left(\sum_{j\in\mathcal{G}} x_{ij} v_{ij} e^{-\delta_j/w_i}\right) + \sum_{j\in\mathcal{G}} \max(0,\delta_j). \tag{Weighted-Primal}$$

The following fact is crucial to the proof of this lemma.

Claim 4.6.10. Let $p(\mathbf{y}) = w \log\left(\sum_{j=1}^{m} c_j\, y_j^{1/w}\right)$ with $w > 0$ and $c_j \geq 0$ for each j. Then $\log p(\mathbf{y})$ is a convex function in $\log(\mathbf{y})$.

Proof. For a fixed $\mathbf{x} \in \mathcal{P}$, the function

$$p_{\mathbf{x}}(\mathbf{y}) = \prod_{i\in\mathcal{A}} \left(\sum_{j\in\mathcal{G}} x_{ij}\, v_{ij}\, y_j^{1/w_i}\right)^{w_i}$$

satisfies all the prerequisites of Lemma 4.6.3. The first property is easy to verify and the second property follows from Fact 4.6.10. Therefore, by Lemma 4.6.3, we get

$$\begin{aligned}
\inf_{\mathbf{y}>0: y^S\geq 0, \forall S\in\binom{\mathcal{G}}{n}} \log p_{\mathbf{x}}(\mathbf{y}) &= \inf_{\mathbf{y}>0: y^S\geq 0, \forall S\in\binom{\mathcal{G}}{n}} \sum_{i\in\mathcal{A}} w_i \log\left(\sum_{j\in\mathcal{G}} x_{ij}\, v_{ij}\, y_j^{1/w_i}\right) \\
&= \sup_{\boldsymbol{\alpha}\in[0,1]^{|\mathcal{G}|}, \sum_j \alpha_j = n} \inf_{\mathbf{y}>0} \sum_{i\in\mathcal{A}} w_i \log\left(\sum_{j\in\mathcal{G}} x_{ij}\, v_{ij}\, y_j^{1/w_i}\right) \\
&\quad - \sum_{j\in\mathcal{G}} \alpha_j \log(y_j).
\end{aligned}$$

Substituting $\delta_j = -\log(y_j)$, and taking the supremum over $\mathbf{x}$, we get

$$\begin{aligned}
&\max_{\mathbf{x}\in\mathcal{P}} \inf_{\mathbf{y}>0: y^S\geq 0, \forall S\in\binom{\mathcal{G}}{n}} \log p_{\mathbf{x}}(\mathbf{y}) \\
&\quad = \sup_{\mathbf{x}\in\mathcal{P}, \boldsymbol{\alpha}\in[0,1]^{|\mathcal{G}|}, \sum_j \alpha_j = n} \inf_{\boldsymbol{\delta}} \sum_{i\in\mathcal{A}} w_i \log\left(\sum_{j\in\mathcal{G}} x_{ij}\, v_{ij}\, e^{-\delta_j/w_i}\right) + \sum_{j\in\mathcal{G}} \alpha_j \delta_j.
\end{aligned}$$

As the domains of both $\mathbf{x}$ and $\boldsymbol{\alpha}$ are compact, using Theorem 4.6.2, we get

$$\begin{aligned}
&\max_{\mathbf{x}\in\mathcal{P}} \inf_{\mathbf{y}>0: y^S\geq 0, \forall S\in\binom{\mathcal{G}}{n}} \sum_{i\in\mathcal{A}} w_i \log\left(\sum_{j\in\mathcal{G}} x_{ij}\, v_{ij}\, y_j^{1/w_i}\right) \\
&\quad = \inf_{\boldsymbol{\delta}} \max_{\mathbf{x}\in\mathcal{P}} \max_{\boldsymbol{\alpha}\in[0,1]^{|\mathcal{G}|}, \sum_j \alpha_j = n} \sum_{i\in\mathcal{A}} w_i \log\left(\sum_{j\in\mathcal{G}} x_{ij}\, v_{ij}\, e^{-\delta_j/w_i}\right) + \sum_{j\in\mathcal{G}} \alpha_j \delta_j.
\end{aligned}$$

Finally, we claim that

$$\inf_{\boldsymbol{\delta}} \max_{\mathbf{x}\in\mathcal{P}} \max_{\boldsymbol{\alpha}\in[0,1]^{|\mathcal{G}|}, \sum_j \alpha_j = n} \sum_{i\in\mathcal{A}} w_i \log\left(\sum_{j\in\mathcal{G}} x_{ij}\, v_{ij}\, e^{-\delta_j/w_i}\right) + \sum_{j\in\mathcal{G}} \alpha_j \delta_j$$

$$= \inf_{\boldsymbol{\delta}} \max_{\mathbf{x}\in\mathcal{P}} \quad \sum_{i\in\mathcal{A}} w_i \log\left(\sum_{j\in\mathcal{G}} x_{ij}\, v_{ij}\, e^{-\delta_j/w_i}\right) + \sum_{j\in\mathcal{G}} \max(0,\delta_j).$$

The proof of this claim is identical to the proof of the unweighted case in Equation (4.29). □

Lemma 4.6.11. *The optimal value of* (Weighted-Primal) *is the same as that of the following program.*

$$\begin{aligned} \inf_{\boldsymbol{\delta},\mathbf{r},\boldsymbol{\gamma}} \quad & \sum_{j\in\mathcal{G}} e^{r_j} + \sum_{i\in\mathcal{A}} w_i\gamma_i + \sum_{j\in\mathcal{G}} \delta_j + \sum_{i\in\mathcal{A}} (w_i \log w_i - w_i) \qquad \text{(Weighted-Dual)} \\ & r_j + \gamma_i + \frac{\delta_j}{w_i} \geq \log v_{ij} \quad \forall (i,j) \in \mathcal{A}\times\mathcal{G}. \end{aligned}$$

Proof. For a fixed $\boldsymbol{\delta}$, let us first re-write the internal maximum of (Weighted-Primal) as

$$\begin{aligned} \max_{\mathbf{x},\mathbf{u}} \quad & \sum_{i\in\mathcal{A}} w_i \log u_i + f(\boldsymbol{\delta}) \qquad (4.37) \\ & u_i \leq \sum_{j\in\mathcal{G}} x_{ij}\, v_{ij}\, e^{-\delta_j/w_i} \quad \forall i \in \mathcal{A} \\ & \sum_{i\in\mathcal{A}} x_{ij} \leq 1 \quad \forall j \in \mathcal{G} \\ & \mathbf{x} \geq \mathbf{0}\,, \end{aligned}$$

where $f(\boldsymbol{\delta}) = \sum_{j\in\mathcal{G}} \max(0,\delta_j)$.

Let β_i, p_j, and θ_{ij} be the Lagrange dual variables associated with the constraints corresponding to agent i, item j, and agent-item pair(i,j), respectively. The Lagrangian of the above convex program is defined as follows

$$\begin{aligned} L(\mathbf{x},\mathbf{u},\boldsymbol{\beta},\boldsymbol{\theta},\mathbf{p}) = f(\boldsymbol{\delta}) &+ \sum_{i\in\mathcal{A}} w_i \log u_i + \sum_{i\in\mathcal{A}} \beta_i \left(\sum_{j\in\mathcal{G}} x_{ij}\, v_{ij}\, e^{-\delta_j/w_i} - u_i\right) \\ &+ \sum_{j\in\mathcal{G}} p_j \left(1 - \sum_{i\in\mathcal{A}} x_{ij}\right) + \sum_{i,j} \theta_{ij} x_{ij} \\ = f(\boldsymbol{\delta}) &+ \Bigg[\sum_{i\in\mathcal{A}} (w_i \log u_i - \beta_i u_i) \\ &+ \sum_{i,j} x_{ij} \left(\beta_i\, v_{ij}\, e^{-\delta_j/w_i} + \theta_{ij} - p_j\right) + \sum_{j\in\mathcal{G}} p_j \Bigg]. \end{aligned}$$

The Lagrange dual of (4.37) is given by

$$g(\boldsymbol{\beta},\boldsymbol{\theta},\boldsymbol{p}) = \max_{\mathbf{x}\in\mathcal{P},\mathbf{u}\geq 0} L(\mathbf{x},\mathbf{u},\boldsymbol{\beta},\boldsymbol{\theta},\mathbf{p}).$$

Observe that solution $x_{ij} = 1/n$ is in the relative interior of $\mathcal{P}$. Since all the constraints are affine, Slater's condition is satisfied. Thus the optimum value of the infimum of Lagrange dual over $\boldsymbol{\beta}, \boldsymbol{\theta}, \boldsymbol{p} \geq 0$ is exactly equal to the optimum of (4.37).

The KKT conditions for the Lagrangian imply

$$\begin{aligned} \frac{w_i}{u_i} - \beta_i = 0 \quad \forall i \in \mathcal{A} \\ \beta_i \, v_{ij} \, e^{-\delta_j / w_i} - \sum_{j \in \mathcal{G}} p_j + \theta_{ij} = 0 \quad \forall (i,j) \in \mathcal{A} \times \mathcal{G}. \end{aligned}$$

The KKT conditions imply that $u_i = w_i/\beta_i$ for each $i \in \mathcal{A}$ maximizes the Lagrangian. For the supremum over $\mathbf{x}, \mathbf{u}$ in (4.36) to stay finite, the second KKT condition is necessary and sufficient. Substituting these conditions in the Langrangian gives the following convex program.

$$\begin{aligned} \inf_{\mathbf{p}, \boldsymbol{\beta}, \boldsymbol{\theta}} \quad & f(\boldsymbol{\delta}) + \sum_{j \in \mathcal{G}} p_j + \sum_{i \in \mathcal{A}} (w_i \log w_i - w_i) - \sum_{i \in \mathcal{A}} w_i \log \beta_i \\ & p_j = \beta_i \, v_{ij} \, e^{-\delta_j / w_i} + \theta_{ij} \quad \forall (i,j) \in \mathcal{A} \times \mathcal{G} \\ & \mathbf{p}, \boldsymbol{\beta}, \boldsymbol{\theta} \geq 0 \end{aligned}$$

Observe that we can remove $\boldsymbol{\theta}$ from the above program while making the first constraint an inequality. By substituting $r_j = \log p_j, \gamma_i = -\log \beta_i$, the above program is equivalently to

$$\begin{aligned} \inf_{\mathbf{r}, \boldsymbol{\gamma}} \quad & f(\boldsymbol{\delta}) + \sum_{j \in \mathcal{G}} e^{r_j} + \sum_{i \in \mathcal{A}} w_i \, \gamma_i + \sum_{i \in \mathcal{A}} (w_i \log w_i - w_i) \\ & r_j + \gamma_i + \frac{\delta_j}{w_i} \geq \log v_{ij} \quad \forall (i,j) \in \mathcal{A} \times \mathcal{G}. \end{aligned}$$

As (Weighted-Primal) involves an infimum over $\boldsymbol{\delta}$, whenever $\delta_j < 0$, we can increase it to $\delta_j = 0$ without increasing the value of $f(\boldsymbol{\delta})$ and maintaining feasibility in the above program. Using this observation and taking an infimum over $\boldsymbol{\delta}$ gives (Weighted-Dual). □

Lemma 4.6.12. *The optimal value of* (Weighted-Dual) *is the same as that of* (CVX-Weighted).

Proof. Let $\hat{b}_{ij}$ be the Lagrange dual variable associated with constraint $r_j + \gamma_i + \delta_j \geq \log v_{ij}$ of (Unweighted-Dual) and let $\mathbf{y}_{ij}$ be the Lagrange dual variable associated with constraint $\delta_{ij} \geq 0$. The Lagrangian of (Weighted-Dual) is defined as follows

$$\begin{aligned} L(\mathbf{r}, \boldsymbol{\gamma}, \boldsymbol{\delta}, \hat{\mathbf{b}}, \boldsymbol{\tau}) &= \sum_{j \in \mathcal{G}} e^{r_j} + \sum_{i \in \mathcal{A}} w_i \, \gamma_i + \sum_{j \in \mathcal{G}} \delta_j + \sum_{i,j} \hat{b}_{ij} (\log v_{ij} - r_j - \gamma_i - \frac{\delta_j}{w_i}) \\ &\quad - \sum_{j \in \mathcal{G}} \delta_j \tau_j + \sum_{i \in \mathcal{A}} (w_i \log w_i - w_i) \\ &= \sum_{j \in \mathcal{G}} (e^{r_j} - (\sum_{i \in \mathcal{A}} \hat{b}_{ij}) r_j) + \sum_{i \in \mathcal{A}} \gamma_i (w_i - \sum_{j \in \mathcal{G}} \hat{b}_{ij}) + \sum_{j \in \mathcal{G}} \delta_j (1 - \tau_j - \sum_{i \in \mathcal{A}} \frac{\hat{b}_{ij}}{w_i}) \end{aligned}$$

$$+ \sum_{i,j} \hat{b}_{ij} \log v_{ij} + \sum_{i \in \mathcal{A}} (w_i \log w_i - w_i).$$

The Lagrange dual of (Weighted-Dual) is given by

$$g(\hat{\mathbf{b}}, \boldsymbol{\tau}) = \inf_{\boldsymbol{\delta} \geq 0, \mathbf{r}, \boldsymbol{\gamma}} L(\mathbf{r}, \boldsymbol{\gamma}, \boldsymbol{\delta}, \hat{\mathbf{b}}, \boldsymbol{\tau}). \tag{4.38}$$

One can verify that Slater's condition is satisfied by (Weighted-Dual). So, the supremum of (4.38) with $\mathbf{b}, \boldsymbol{\tau} \geq 0$ is equal to the optimum of (Weighted-Dual).

The KKT conditions for the Langrangian imply

$$\begin{aligned} e^{r_j} - \sum_{i \in \mathcal{A}} \hat{b}_{ij} &= 0 \\ w_i - \sum_{j \in \mathcal{G}} \hat{b}_{ij} &= 0 \\ 1 - \tau_j - \sum_{j \in A} \frac{\hat{b}_{ij}}{w_i} &= 0. \end{aligned}$$

The KKT conditions imply that the minimizer for r_j is given by $r_j = \log \left(\sum_{i \in \mathcal{A}} \hat{b}_{ij} \right)$. For the infimum over $\boldsymbol{\gamma}, \boldsymbol{\delta}$ to stay finite, the conditions $w_i = \sum_{j \in \mathcal{G}} \hat{b}_{ij}$ for each $i \in \mathcal{A}$ and $1 - \tau_j = \sum_{i \in \mathcal{A}} \hat{b}_{ij}$ for each $j \in \mathcal{G}$ are necessary and sufficient. Substituting these conditions in the Lagrange dual, we get

$$\begin{aligned} \sup_{\hat{\mathbf{b}}, \boldsymbol{\tau}} \quad & \sum_{i,j} \hat{b}_{ij} \log v_{ij} - \sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} \hat{b}_{ij} \log \left(\sum_{i \in \mathcal{A}} \hat{b}_{ij} \right) + \sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} \hat{b}_{ij} + \sum_{i \in \mathcal{A}} (w_i \log w_i - w_i) \\ & \sum_{j \in \mathcal{G}} \hat{b}_{ij} = w_i \\ & \sum_{i \in \mathcal{A}} \frac{\hat{b}_{ij}}{w_i} = 1 - \tau_j \\ & \hat{\mathbf{b}}, \boldsymbol{\tau} \geq \mathbf{0}. \end{aligned}$$

Observe that the supremum in the above program can be switched to maximum because the feasible region is compact and the objective is bounded. Also, $\sum_{i,j} \hat{b}_{ij} = \sum_{i \in \mathcal{A}} w_i$ for any feasible $\hat{\mathbf{b}}$. Finally, substituting $b_{ij} = \frac{\hat{b}_{ij}}{w_i}$ and $q_j = \sum_{i \in \mathcal{A}} b_{ij}$, we obtain (CVX-Weighted). □

4.6.3 Improving the Approximation in Pseudo-polynomial Time.

It is possible to save the factor of $D_{\mathrm{KL}}(\mathbf{w} \,\|\, \mathbf{u})$ which is lost in Algorithm 6 if, instead of using an optimal solution to the support-restricted convex program, we use a locally-optimal solution to the non-convex program. The downside is that we are unable to find such a locally-optimal

solution in polynomial time in the inputs. Instead, we give a pseudo-polynomial time algorithm, where the running-time depends on the unary representation of the weights w_i and the valuations v_{ij}, which does give a polynomial-time algorithm in the case that the valuations are bounded.

Algorithm 7 Alternative Algorithm for Rounding an Acyclic Solution

Require: NSW instance $(\mathcal{A}, \mathcal{G}, \mathbf{v}, \mathbf{w})$, solution $(\mathbf{b}, \mathbf{q}) \in \mathcal{P}(\mathcal{A}, \mathcal{G})$

1: $(\mathbf{b}^\star, \mathbf{q}^\star) \leftarrow$ locally optimal acyclic solution of (NCVX-Weighted) starting from $(\mathbf{b}, \mathbf{q})$

2: $F^\star \leftarrow G_{\text{supp}}(\mathbf{b}^\star)$ with every tree rooted at an agent node

3: Remove edges between item j and its children in $F^\star$ whenever $q_j^\star < \frac{1}{2}$ to get forest $\widetilde{F}$ ▷ Pruning

4: $L_i^\star \leftarrow$ set of leaf children of agent i in $\widetilde{F}$; $\quad L^\star \leftarrow \bigcup_i L_i^\star$

5: $M^\star \leftarrow$ matching between $\mathcal{A} \to \mathcal{G} \setminus L^\star$ in $\widetilde{F}$ maximizing:

$$w_{\widetilde{F}}(M) := \sum_{i \in \mathcal{A}} w_i \log \left(v_{iM(i)} + \sum_{j \in L_i^\star} v_{ij} \right)$$

6: $\sigma^\star \leftarrow$ assignment with $\sigma^\star(j) = i$ if $j \in L_i^\star \cup M^\star(i)$ ▷ Matching

Ensure: $\sigma^\star$

The only difference with the previous algorithm is that the input solution need not be acyclic, and in step 2 we find a locally-optimal solution of the non-convex program. We will also need an alternative result to replace Theorem 4.1.6 in the analysis.

Theorem 4.6.13. *For a Nash Social Welfare instance $(\mathcal{A}, \mathcal{G}, \mathbf{v}, \mathbf{w})$, given a vector $\mathbf{b} \in \mathcal{P}(\mathcal{A}, \mathcal{G})$ such that the support of $\mathbf{b}$ is a forest, there exists a deterministic pseudo-polynomial time algorithm (Algorithm 6) which returns an assignment $\sigma : \mathcal{G} \to \mathcal{A}$ such that*

$$\text{NSW}(\sigma) \geq f_{\text{ncvx}}(\mathbf{b}) - 2 \log 2 - \frac{1}{2e}.$$

Note that, other than the running time, this result would imply Theorem 4.1.6 as well since Lemma 4.2.4 states that $f_{\text{ncvx}}(\mathbf{b}) \geq f_{\text{cvx}}(\mathbf{b}) - D_{\text{KL}}(\mathbf{w} \,\|\, \mathbf{u})$.

Looking into the proof of Theorem 4.1.6, the only thing which we need to change is to provide an alternative to Lemma 4.4.3 in terms of the non-convex objective. Lemma 4.4.3 gave us a way to bound the change in objective between an acyclic locally-optimal solution and another solutions with the same support. The proof used the optimality conditions for (CVX-Weighted), so we will use the local-optimality conditions for (NCVX-Weighted) and prove the following lemma.

Lemma 4.6.14. *Let $(\mathbf{b}^\star, \mathbf{q}^\star)$ be an acyclic locally-optimal solution to* (NCVX-Weighted)*. Let $(\mathbf{b}, \mathbf{q})$ be a feasible point in $\mathcal{P}(\mathcal{A}, \mathcal{G})$ such that the support of $\mathbf{b}$ is a subset of the support of $\mathbf{b}^\star$, and for any $j \in \mathcal{G}$, if $q_j^\star = 1$, then $q_j = 1$. Then*

$$f_{\text{ncvx}}(\mathbf{b}^\star, \mathbf{q}^\star) - f_{\text{ncvx}}(\mathbf{b}, \mathbf{q}) \leq \sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} w_i \, b_{ij} \log \left(\frac{\sum_{i \in \mathcal{A}} w_i \, b_{ij}}{\sum_{i \in \mathcal{A}} w_i \, b_{ij}^\star} \right).$$

Before providing a proof of this lemma, we will use it to prove Theorem 4.6.13. The proof is essentially identical to the proof of Theorem 4.1.6

Proof of Theorem 4.6.13. Let $(\mathbf{b}^\star, \mathbf{q}^\star)$ be a locally-optimal acyclic solution of (NCVX-Weighted), let $\widetilde{F}$ denote the forest obtained after pruning $G_{\text{supp}}(\mathbf{b}^\star)$. Let $L_i^\star$ denote the set of leaf children of agent i in $\widetilde{F}$.

Let $(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}})$ be a feasible solution guaranteed by Lemma 4.4.4 on input $(\mathbf{b}^\star, \mathbf{q}^\star)$. We are guaranteed that $q_j^\star \le q_j^{\text{pruned}}$ for each item j with $q_j^\star \ge 1/2$, each item with $q_j^\star$ is a leaf in the support of the pruned solution connected to its parent in F, and for any $(i,j) \in \mathcal{A} \times \mathcal{G}$, $b_{ij}^{\text{pruned}} \le \min\{1, 2 \cdot b_{ij}^\star\}$. Using Lemma 4.6.14 the difference in objective is bounded

$$\begin{aligned} f_{\text{ncvx}}(\mathbf{b}^\star, \mathbf{q}^\star) - f_{\text{ncvx}}(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}}) &\le \sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} w_i \, b_{ij}^{\text{pruned}} \log\left(\frac{\sum_{i \in \mathcal{A}} w_i \, b_{ij}^{\text{pruned}}}{\sum_{i \in \mathcal{A}} w_i \, b_{ij}^\star}\right) \\ &\le \sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} w_i \, b_{ij}^{\text{pruned}} \log 2 = \log 2 \end{aligned}$$

Using Lemma 4.4.2 on $(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}})$ with function $S(i) = L_i^\star$, we conclude that there exists a matching, M, in $G_{\text{supp}}(\mathbf{b}^{\text{pruned}})$ such that

$$\begin{aligned} \sum_{i \in \mathcal{A}} w_i \log\left(v_{iM(i)} + \sum_{j \in L_i^\star} v_{ij}\right) &= \sum_{i \in \mathcal{A}} w_i \log\left(v_{iM(i)} + \sum_{j \in S(i)} v_{ij}\right) \\ &\ge f_{\text{ncvx}}(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}}) - \log 2 - \frac{1}{2e}. \end{aligned}$$

Since $G_{\text{supp}}(\mathbf{b}^{\text{pruned}})$ is a subgraph of $\widetilde{F}$, the matching M is also present in $\widetilde{F}$. Therefore, the matching $M^\star$ (and corresponding assignment $\sigma^\star$) returned by Algorithm 7 satisfies

$$\begin{aligned} \text{NSW}(\sigma^\star) = \sum_{i \in \mathcal{A}} w_i \log\left(v_{iM^\star(i)} + \sum_{j \in L_i^\star} v_{ij}\right) &\ge \sum_{i \in \mathcal{A}} w_i \log\left(v_{iM(i)} + \sum_{j \in L_i^\star} v_{ij}\right) \\ &\ge f_{\text{ncvx}}(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}}) - \log 2 - \frac{1}{2e} \\ &\ge f_{\text{ncvx}}(\mathbf{b}^\star, \mathbf{q}^\star) - 2\log 2 - \frac{1}{2e}, \end{aligned}$$

where the second inequality follows from Lemma 4.4.2, and the last inequality follows from our earlier bound on the difference in objective between the two solutions. □

Technical Lemmas

In this section we provide a proof of Lemma 4.6.14. We will break the proof into a couple intermediate steps to highlight how we are using the local optimality conditions, and the relationship between the natural conclusions for the convex and non-convex objectives. The first lemma summarizes the first-order optimality conditions.

Lemma 4.6.15. *If $(\mathbf{b}^\star, \mathbf{q}^\star)$ is a first-order stationary solution of NCVX-Weighted then there exists real numbers $\{\lambda_i\}_{i\in\mathcal{A}}$ and $\{\eta_j\}_{j\in\mathcal{G}} \geq 0$ such that $\eta_j(1-q_j^\star) = 0$ for all $j \in \mathcal{G}$ and if $b_{ij}^\star > 0$ then*

$$\log v_{ij} = \log q_j^\star + \frac{\lambda_i}{w_i} + \frac{\sum_{i\in\mathcal{A}} w_i b_{ij}}{w_i q_j^\star} + \frac{\eta_j}{w_i}.$$

Proof. If $(\mathbf{b}^\star, \mathbf{q}^\star)$ is a first-order stationary solution of (NCVX-Weighted), then using the KKT conditions, there exist $\lambda_i, \gamma_j \in \mathbb{R}$ and $\alpha_{ij}, \eta_j \geq 0$ such that

$$\frac{\partial L}{\partial b_{ij}^\star} = w_i \log v_{ij} - w_i \log q_j^\star - \lambda + \gamma + \alpha_{ij} = 0$$

$$\frac{\partial L}{\partial q_j^\star} = -\frac{\sum_{i\in\mathcal{A}} w_i b_{ij}^\star}{q_j^\star} + \gamma_j - \eta_j = 0,$$

and which satisfy the complementary slackness conditions:

$$\eta_j(1-q_j^\star) = 0$$
$$\alpha_{ij} b_{ij}^\star = 0.$$

Substituting the value of γ from the second Equation into the first and using the complementary slackness condition, if $b_{ij} > 0$ we see

$$\log v_{ij} = \log q_j^\star + \frac{\lambda_i}{w_i} + \frac{\sum_{i\in\mathcal{A} w_i b_{ij}}}{w_i q_j^\star} + \frac{\eta_j}{w_i}.$$

□

The next lemma uses the first-order optimality conditions to derive a bound on the change in objective for certain types of solutions.

Lemma 4.6.16. *Let $(\mathbf{b}^\star, \mathbf{q}^\star)$ be any acyclic first order stationary point of NCVX-Weighted. Let $(\mathbf{b}, \mathbf{q})$ be a feasible point in $\mathcal{P}(\mathcal{A}, \mathcal{G})$ such that the support of $\mathbf{b}$ is a subset of the support of $\mathbf{b}^\star$, and for any $j \in \mathcal{G}$, if $q_j^\star = 1$, then $q_j = 1$. Then*

$$f_{\text{ncvx}}(\mathbf{b}^\star, \mathbf{q}^\star) - f_{\text{ncvx}}(\mathbf{b}, \mathbf{q}) = 1 + \sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{G}} w_i b_{ij} \log\left(\frac{q_j}{q_j^\star}\right) - \sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{G}} w_i b_{ij}^\star \left(\frac{q_j}{q_j^\star}\right).$$

Proof. Expanding the difference between the two objective values, we get

$$f_{\text{ncvx}}(\mathbf{b}^\star, \mathbf{q}^\star) - f_{\text{ncvx}}(\mathbf{b}, \mathbf{q}) = \sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{G}} w_i(b_{ij}^\star - b_{ij}) \log v_{ij} - w_i b_{ij}^\star \log q_j^\star + w_i b_{ij} \log q_j.$$

Since $(\mathbf{b}^\star, \mathbf{q}^\star)$ is locally optimal, Lemma 4.6.15 implies that there exist real numbers $\{\lambda_i\}_{i\in\mathcal{A}}$ and $\{\eta_j\}_{j\in\mathcal{G}} \geq 0$ such that $\eta_j(1-q_j^\star) = 0$ for all $j \in \mathcal{G}$ and if $b_{ij}^\star > 0$ then

$$\log v_{ij} = \log q_j^\star + \frac{\lambda_i}{w_i} + \frac{\sum_{i\in\mathcal{A}} w_i b_{ij}}{w_i q_j^\star} + \frac{\eta_j}{w_i}.$$

Substituting this into our expression for the change in objective gives

$$\begin{aligned} f_{\text{ncvx}}(\mathbf{b}^\star, \mathbf{q}^\star) - f_{\text{ncvx}}(\mathbf{b}, \mathbf{q}) &= \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{G}} w_i (b_{ij}^\star - b_{ij}) \left(\log q_j^\star + \frac{\lambda_i}{w_i} + \frac{\sum_{i \in \mathcal{A}} w_i b_{ij}}{w_i q_j^\star} + \frac{\eta_j}{w_i} \right) \\ &\quad + \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{G}} w_i b_{ij} \log q_j - w_i b_{ij}^\star \log q_j^\star \\ &= \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{G}} w_i b_{ij} \left(\log q_j - \log q_j^\star \right) + \sum_{i \in \mathcal{A}} \lambda_i \left(\sum_{j \in \mathcal{G}} b_{ij}^\star - \sum_{j \in \mathcal{G}} b_{ij} \right) \\ &\quad + \sum_{j \in \mathcal{G}} \left(\eta_j + \frac{\sum_{i \in \mathcal{A}} w_i b_{ij}^\star}{q_j^\star} \right) \left(\sum_{i \in \mathcal{A}} b_{ij}^\star - \sum_{i \in A} b_{ij} \right). \end{aligned}$$

Using the fact that $\sum_{j \in \mathcal{G}} b_{ij}^\star = \sum_{j \in \mathcal{G}} b_{ij} = 1$ we can eliminate a term, and use $\sum_{i \in \mathcal{A}} b_{ij}^\star = q_j^\star$ and $\sum_{i \in \mathcal{A}} b_{ij} = q_j$ to simplify and see that

$$\begin{aligned} f_{\text{ncvx}}(\mathbf{b}^\star, \mathbf{q}^\star) - f_{\text{ncvx}}(\mathbf{b}, \mathbf{q}) &= \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{G}} w_i b_{ij} \left(\log q_j - \log q_j^\star \right) \\ &\quad + \sum_{j \in \mathcal{G}} \left(\eta_j + \frac{\sum_{i \in \mathcal{A}} w_i b_{ij}^\star}{q_j^\star} \right) \left(q_j^\star - q_j \right). \end{aligned}$$

Now, using the complementary slackness conditions, we know that if $q_j^\star < 1$ then $\eta_j = 0$ and therefore $\eta_j(q_j^\star - q_j) = 0$. Alternatively, if $q_j^\star = 1$, then by assumption $q_j = 1$ and still $\eta_j(q_j^\star - q_j) = 0$. Substituting this in the above

$$\begin{aligned} f_{\text{ncvx}}(\mathbf{b}^\star, \mathbf{q}^\star) - f_{\text{ncvx}}(\mathbf{b}, \mathbf{q}) &= \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{G}} w_i b_{ij} \left(\log q_j - \log q_j^\star \right) + \sum_{j \in \mathcal{G}} \frac{\sum_{i \in \mathcal{A}} w_i b_{ij}^\star}{q_j^\star} \left(q_j^\star - q_j \right) \\ &= 1 + \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{G}} w_i b_{ij} \log \left(\frac{q_j}{q_j^\star} \right) - \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{G}} w_i b_{ij}^\star \left(\frac{q_j}{q_j^\star} \right). \end{aligned}$$

□

Now we are finally ready to prove the main lemma.

Proof of Lemma 4.6.14. Using Lemma 4.6.16 we know that

$$f_{\text{ncvx}}(\mathbf{b}^\star, \mathbf{q}^\star) - f_{\text{ncvx}}(\mathbf{b}, \mathbf{q}) = 1 + \sum_{j \in \mathcal{G}} \log \left(\frac{q_j}{q_j^\star} \right) \sum_{i \in \mathcal{A}} w_i b_{ij} - \sum_{j \in \mathcal{G}} \left(\frac{q_j}{q_j^\star} \right) \sum_{i \in \mathcal{A}} w_i b_{ij}^\star.$$

Note that for any item j

$$\log \left(\frac{q_j}{q_j^\star} \right) \sum_{i \in \mathcal{A}} w_i b_{ij} - \left(\frac{q_j}{q_j^\star} \right) \sum_{i \in \mathcal{A}} w_i b_{ij}^\star \leq \left(\sum_{i \in \mathcal{A}} w_i b_{ij} \right) \log \left(\frac{\sum_{i \in \mathcal{A}} w_i b_{ij}}{\sum_{i \in \mathcal{A}} w_i b_{ij}^\star} \right) - \left(\sum_{i \in \mathcal{A}} w_i b_{ij} \right).$$

This inequality follows from the fact that for any $\alpha, \beta \geq 0$

$$\max_{x \geq 0} \alpha \log(x) - \beta x = \alpha \log\left(\frac{\alpha}{\beta}\right) - \alpha.$$

Since $\sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} w_i b_{ij} = 1$ we conclude that

$$f_{\text{ncvx}}(\mathbf{b}^\star, \mathbf{q}^\star) - f_{\text{ncvx}}(\mathbf{b}, \mathbf{q}) \leq \sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} w_i \, b_{ij} \log\left(\frac{\sum_{i \in \mathcal{A}} w_i \, b_{ij}}{\sum_{i \in \mathcal{A}} w_i \, b^\star_{ij}}\right).$$

□

Finding a locally-optimal solution

Our specific goal is to find a locally optimal solution to the problem $\max_{\mathbf{b} \in \mathcal{P}(\mathcal{A}, \mathcal{G})} f_{\text{ncvx}}(\mathbf{b})$. Our objective is not concave, but recall that

$$f_{\text{cvx}}(\mathbf{b}) - f_{\text{ncvx}}(\mathbf{b}) = D_{\text{KL}}(\mathbf{w} \,\|\, \mathbf{u}) - D_{\text{KL}}(\mu \,\|\, \theta)\,.$$

where

$$\mu_j(b) := \sum_{i \in A} w_i b_{ij} \quad \text{and} \quad \theta_j(b) := \frac{\sum_{i \in A} b_{ij}}{n}.$$

The function f_{cvx} is concave, and the first term in the gap is a constant. Therefore, we can define

$$g(b) := D_{\text{KL}}(\mu(b) \,\|\, \theta(b))\,,$$

so that $-f_{\text{ncvx}}(b) + g(b)$ is convex.

With this setup we can fit our problem into a more general framework. Suppose we want to find a locally optimal solution to the problem $\min_{x \in K} f(x)$, where $f(x)$ is not necessarily convex, but we do have the additional assumption that there is a convex function $g(x)$ such that $f(x) + g(x)$ is convex. The following approach is a simplified version of standard techniques from first-order methods [122], and is also similar to the approach used for weakly convex functions [64]. To get an approximate locally optimal solution we use gradient descent with the Bregman divergence of g:

$$D_g(x, y) := g(x) - g(y) - \langle \nabla g(y), x - y \rangle.$$

Algorithm 8 Gradient Descent with Bregman Divergence

Require: Functions f, g such that g and $f + g$ are convex; $\delta > 0$; initial $x_0 \in K$

1: $k \leftarrow 0$
2: **while** $\|\nabla f(x_k)\| > \delta$ **do**
3: $\quad x_{k+1} \leftarrow \arg\min_{x \in K} f(x) + D_g(x, x_k)$
4: $\quad k \leftarrow k + 1$
5: **end while**

Ensure: x_k

We can solve the internal minimization problem, since the objective is convex.

Lemma 4.6.17. *If f and g are functions such that g and $f+g$ are convex, then for any fixed choice of $y \in \mathbb{R}^n$, the function $f(x) + D_g(x,y)$ is convex.*

Proof. By the definition of Bregman divergence $f(x) + D_g(x,y)$ is equal to $f(x) + g(x)$ with some additional linear term that does not affect convextity. □

After sufficiently many iteration, we are guaranteed to find a point x where $\nabla f(x)$ is close to zero.

Lemma 4.6.18. *Let f be a function such that there exists a convex g such that $f+g$ is convex and where $D_g(x,x') < \epsilon$ implies $\|\nabla g(x) - \nabla g(x')\| < \delta$, and let K be a convex set. Then for $N \geq \frac{f(x_0) - \max_{x^* \in K} f(x^*)}{\epsilon}$ there is $i \leq N$ such that the distance between $\nabla f(x_i)$ and the normal cone of K is at most δ.*

Proof. We will first show that there is $i \leq N$ such that

$$D_g(x_{i+1}, x_i) \leq \epsilon$$

as follows. By the optimality of x_{k+1} we know that

$$f(x_{k+1}) + D_g(x_{k+1}, x_k) \leq f(x_k) + D_g(x_k, x_k) = f(x_k).$$

Summing, we get

$$f(x_N) + \sum_{i=1}^{N-1} D_g(x_{i+1}, x_i) \leq f(x_0).$$

Now we pick $N = \frac{f(x_0) - \max_{x^* \in K} f(x^*)}{\epsilon}$, so that by averaging, there is $i \leq N$ such that $D_g(x_{i+1}, x_i) \leq \epsilon$. By the assumption on the function g we know that $\|\nabla g(x_{i+1}) - \nabla g(x_i)\| < \delta$,. By the optimality of x_{i+1} we know that the gradient of $f(x) + D_g(x, x_i)$ is inside the normal cone of K at x_{i+1}. Thus we see that

$$\nabla f(x_{i+1}) + \nabla g(x_{i+1}) - \nabla g(x_i) = \nabla_x f(x_{i+1}) + \nabla_x D_g(x_{i+1}, x_i),$$

is inside the normal cone of K at x_{i+1}. Thus the distance from $\nabla f(x_{i+1})$ to the normal cone of K at x_{i+1} is at most $\|\nabla g(x_i) - \nabla g(x_{i+1})\|$, which we know is small by our choice of i. □

The rest of this section is devoted to showing that $D_g(b,b') < \epsilon$ implies $\|\nabla g(b) - \nabla g(b')\| < \delta$ for our particular choice of g, and where δ depends polynomially on n, the weights w_i and the valuations v_{ij}.

To ensure the gradient descent algorithm runs in pseudo-polynomial time, we need lower bounds on μ and θ which will be used later to upper bound the eigenvalues of the hessian of g^*. To do this, we give a slightly stronger relaxation. For each $i \in \mathcal{A}$ let $\rho_i = \frac{\min_{j: v_{ij} > 0} v_{ij}}{\max_{j: v_{ij} > 0} v_{ij}}$ and let $\rho^* = \min_i \rho_i$.

$$\max_{\mathbf{b}} \quad f_{\text{ncvx}}(\mathbf{b}) := \sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{G}} w_i\, b_{ij} \log v_{ij} - \sum_{j\in\mathcal{G}}\sum_{i\in\mathcal{A}} w_i\, b_{ij} \log\left(\sum_{i\in\mathcal{A}} b_{ij}\right)$$

$$\begin{aligned}
\text{s.t.} \quad & \sum_{j\in\mathcal{G}} b_{ij} = 1 \quad \forall i \in \mathcal{A} \\
& \sum_{i\in\mathcal{A}} b_{ij} \le 1 \quad \forall j \in \mathcal{G} \\
& b_{ij} \ge 0 \quad \forall (i,j) \in \mathcal{A}\times\mathcal{G} \\
& \frac{1}{n}\sum_{i\in\mathcal{A}} b_{ij} \ge \frac{1}{n^2}\rho^* \quad \forall (i,j) \in \mathcal{A}\times\mathcal{G} \\
& \sum_{i\in\mathcal{A}} w_i b_{ij} \ge \frac{\min_i w_i}{n}\rho^* \quad \forall (i,j) \in \mathcal{A}\times\mathcal{G}.
\end{aligned}$$

This is still a relaxation for the weighted Nash Social Welfare problem, because for any assignment $\sigma : \mathcal{G} \to \mathcal{A}$ we can again set

$$b_{ij} = \begin{cases} \frac{v_{ij}}{V_i} & i = \sigma(j) \\ 0 & \text{otherwise} \end{cases}$$

and we see that

$$\begin{aligned}
\frac{1}{n}\sum_{i\in\mathcal{A}} b_{ij} &= \frac{1}{n}\frac{v_{\sigma(j)j}}{V_{\sigma(j)}} \ge \frac{1}{n^2}\rho^*, \text{ and} \\
\sum_{i\in\mathcal{A}} w_i b_{ij} &= w_i \frac{v_{\sigma(j)j}}{V_{\sigma(j)}} \ge \frac{\min_i w_i}{n}\rho^*.
\end{aligned}$$

Let $\mathcal{P}(\mathcal{A},\mathcal{G};\rho^*)$ denote the feasible region. Note that the lower bounds in the relaxation are not polynomial in the input size for the valuations v_{ij}, but depend on the unary representation of the valuations. This will still give a polynomial time algorithm when the valuations are bounded.

Recall that

$$D_g(b,b') = g(b) - g(b') - \langle \nabla g(b'), b - b' \rangle.$$

and that the Fenchel conjugate of g is defined as

$$g^*(x) = \sup_y \{\langle y, x\rangle - g(y)\},$$

and that it is always convex.

We will use the following two well-known facts about Bregman divergence and Fenchel conjugates. It allows us to relate a small Bregman divergence to a small Bregman divergence for the Fenchel conjugate between the swapped gradients.

Fact 4.6.19.

$$D_g(b, b') = D_{g^*}(\nabla g(b'), \nabla g(b)).$$

We want to show that if $D_g(b, b') < \epsilon$ then $\|\nabla g(b) - \nabla g(b')\| < \delta$, where ϵ and δ are only polynomially small in terms of the input. Using the above lemma, this becomes equivalent to showing a quadratic lower bound (with polynomial-sized coefficient) on D_{g^*}. To get there, we will also need the following fact about Fenchel conjugate.

Fact 4.6.20.

$$\nabla g^*(x) = (\nabla g)^{-1}(x).$$

We will use this fact as follows: the optimal choice of y is $\nabla g^{-1}(x) = \nabla g^*(x)$, so if we can calculate the Jacobian of y, this will also be the Hessian of g^* at x. Note that since g^* is always convex, we know that the Hessian of g^* will be positive semidefinite.

Lemma 4.6.21. *If $\mathbf{b} \in \mathcal{P}(\mathcal{A}, \mathcal{G}; \rho^*)$ then $D_g(b, b') < \epsilon$ implies $\|\nabla g(b) - \nabla g(b')\| < \delta$, where ϵ depends linearly on δ and polynomially on n, the weights w_i and the valuations v_{ij}.*

Proof. In what follows, we will keep everything in terms of θ_j and μ_j as much as possible for the sake of clarity. Just note that $\mu_j(b)$ and $\theta_j(b)$ depend only on $\{b_{ij}\}_{i \in A}$, and that for all $i \in A$

$$\frac{\partial}{\partial b_{ij}} \mu_j(b) = w_i \quad \text{and} \quad \frac{\partial}{\partial b_{ij}} \theta_j(b) = \frac{1}{n}.$$

Thus,

$$\begin{aligned}\frac{\partial}{\partial b_{ij}} D_{\mathrm{KL}}(\mu \,\|\, \theta) &= w_i \log \mu_j(b) + w_i - w_i \log \theta_j(b) - \frac{1}{n} \frac{\mu_j(b)}{\theta_j(b)} \\ &= w_i \log \frac{\mu_j(b)}{\theta_j(b)} + w_i - \frac{1}{n} \frac{\mu_j(b)}{\theta_j(b)}.\end{aligned}$$

This gives us the negative gradient of g. Since the sum on the KL-divergence is separable over j, we will suppress the j index from now on for simplicity. The Hessian of g will be block diagonal with a block for each j, so it suffices to prove a lower bound on the eigenvalues of each block. For fixed x, we know that the optimal choice of y in the definition of $g^*(x)$ will satisfy

$$x_i = w_i \log \frac{\mu(y)}{\theta(y)} + w_i - \frac{1}{n} \frac{\mu(y)}{\theta(y)}.$$

Differentiating this with respect to x_i and x_k for $k \neq i$ we get the following system of equations:

$$\begin{aligned}1 &= \sum_{i' \in A} \frac{\partial y_{i'}}{\partial x_i} \cdot \frac{\partial}{\partial y_{i'}} \left(w_i \log \frac{\mu(y)}{\theta(y)} + w_i - \frac{1}{n} \frac{\mu(y)}{\theta(y)} \right) \\ &= \sum_{i' \in A} \frac{\partial y_{i'}}{\partial x_i} \cdot \left(\frac{w_i w_{i'}}{\mu(y)} - \frac{w_i}{n\theta(y)} - \frac{w_{i'}\theta(y) - \frac{1}{n}\mu(y)}{n\theta(y)^2} \right)\end{aligned}$$

$$= \sum_{i' \in A} \frac{\partial y_{i'}}{\partial x_i} \cdot \left(\frac{w_i w_{i'}}{\mu(y)} - \frac{w_i + w_{i'}}{n\theta(y)} + \frac{\mu(y)}{n^2\theta(y)^2} \right),$$

and, using similar simplifications,

$$0 = \sum_{i' \in A} \frac{\partial y_{i'}}{\partial x_k} \cdot \left(\frac{w_i w_{i'}}{\mu(y)} - \frac{w_i + w_{i'}}{n\theta(y)} + \frac{\mu(y)}{n^2\theta(y)^2} \right).$$

Note that the terms

$$\frac{w_i w_{i'}}{\mu_j(y)} - \frac{w_i + w_{i'}}{n\theta(y)} + \frac{\mu(y)}{n^2\theta(y)^2},$$

depend only on i and i', and they are symmetric so we can put them into a matrix

$$A = [a_{ii'}] = \left[\frac{w_i w_{i'}}{\mu(y)} - \frac{w_i + w_{i'}}{n\theta(y)} + \frac{\mu(y)}{n^2\theta(y)^2} \right].$$

Rewriting our system of equations, and noting that $D_x y = \nabla^2 g^*$, we get the matrix equation

$$A \cdot \nabla^2 g^* = I,$$

which implies that $\nabla^2 g^* = A^{-1}$. From here, it suffices to prove a polynomial-sized upper bound on the eigenvalues of A, from which we can conclude a lower bound on the eigenvalues of $\nabla^2 g^*$. Since we only care about a upper bound that is polynomial in n, it suffices to consider the trace of A. This will always give an upper bound on the maximum eigenvalue, and dividing the trace by n gives a lower bound on the maximum eigenvalue, so this is the correct term up to polynomial factors. The diagonal entries of A are

$$a_{ii} = \frac{w_i^2}{\mu(y)} - \frac{2w_i}{n\theta(y)} + \frac{\mu(y)}{n^2\theta(y)^2},$$

so the trace is

$$\sum_i a_{ii} = \frac{\sum_i w_i^2}{\mu(y)} - \frac{2}{n\theta(y)} + \frac{\mu(y)}{n\theta(y)^2}.$$

Note also, that when $w_i = \frac{1}{n}$, the matrix A has all entries equal to zero. This is consistent, since the function g is constant in this case, and so g^* will be $+\infty$ everywhere. Because we have enforced lower bounds on $\theta(y)$ and $\mu(y)$ this gives an upper bound on the eigenvalues of $\nabla^2 g^*$ which is polynomial in n, the weights w_i and the valuations v_{ij}. □

This is enough to prove the following theorem, which allows us to find locally optimal solutions to the non-convex program, and explains the loss in runtime.

Theorem 4.6.22. *Algorithm 8 finds an approximately locally-optimal solution maximizing f_{ncvx} over the restricted feasible region $\mathcal{P}(\mathcal{A}, \mathcal{G}; \rho^*)$ in time polynomial in n, $1/\epsilon$, the weights w_i, and the valuations v_{ij}. In particular, when the valuations v_{ij} and the weights w_i are polynomially bounded by in terms of n it is polynomial time.*

Chapter 5

Fair Subspace Approximation

5.1 Introduction

Large data sets, often represented as collections of high-dimensional points, naturally arise in fields such as machine learning, data mining, and computational geometry. Despite their high-dimensional nature, these points typically exhibit low intrinsic dimensionality. Identifying (or summarizing) this underlying low-dimensional structure is a fundamental algorithmic question with numerous applications to data analysis. We study a general formulation, that we call the *subspace approximation problem*.

In subspace approximation, given a set of n points $\{a_1, \ldots, a_n\} \in \mathbb{R}^d$ and a rank parameter k, we consider the problem of "best approximating" the input points with a k-dimensional subspace in a high-dimensional space. Here the goal is to find a rank k projection $\boldsymbol{P}$ that minimizes the projection costs $\|a_i - \boldsymbol{P}a_i\|$, aggregated over $i \in [n]$. The choice of aggregation leads to different well-studied formulations. In the ℓ_p subspace approximation problem, the objective is $(\sum_i \|a_i - \boldsymbol{P}a_i\|_2^p)$. Formally, denoting by A the $d \times n$ matrix whose ith column is a_i, the ℓ_p-subspace approximation problem asks to find a rank k projection matrix $\boldsymbol{P} \in \mathbb{R}^{d\times d}$ that minimizes $\|\boldsymbol{A}-\boldsymbol{P}\boldsymbol{A}\|_{2,p}^p := \sum_{i=1}^n \|a_i-\boldsymbol{P}a_i\|_2^p$. For different choices of p, ℓ_p-subspace approximation captures some well-studied problems, notably the *median hyperplane problem* (when $p = 1$), the *principal component analysis (PCA) problem* (when $p = 2$), and the *center hyperplane problem* (when $p = \infty$).

Subspace approximation for general p turns out to be NP-hard for all $p \neq 2$. For $p > 2$, semidefinite programming helps achieve a constant factor approximation (for fixed p) for the problem [67]. Matching hardness results were also shown for the case $p > 2$, first assuming the Unique Games Conjecture [67], and then based only on P $\neq$ NP [95]. For $p < 2$, hardness results were first shown in the work of [51].

Due to the ubiquitous applications of subspace approximation in various domains, several "constrained" versions of the problem have been extensively studied as well [13, 29, 53, 70, 152, 192]. In the most general setting of the *constrained ℓ_p-subspace approximation* problem, we are addi-

tionally given a collection $\mathcal{S}$ of rank-k projection matrices (specified either explicitly or implicitly) and the goal is to find a projection matrix $\boldsymbol{P} \in \mathcal{S}$ minimizing the objective. I.e.,

$$\min_{\boldsymbol{P} \in \mathcal{S}} \|\boldsymbol{A} - \boldsymbol{P}\boldsymbol{A}\|_{2,p}^p. \tag{5.1}$$

Some examples of problems in constrained subspace approximation include the well-studied *column subset selection* [4, 26, 28, 49, 66, 94, 183] where the projection matrices are constrained to project onto the span of k of the original vectors, (k, z)*-means clustering* in which the set of projection matrices can be specified by the partitioning of the points into k clusters (see [53] for a reference), and many more which we will describe in this chapter.

5.1.1 Our Contributions and Applications

In this chapter, we provide a general algorithmic framework for constrained ℓ_p-subspace approximation that yields either $(1+\varepsilon)$-multiplicative or ε-additive error approximations to the objective (depending on the setting), with running time exponential in k. We apply the framework to several classes of constrained subspace approximation, leading to new results or results matching the state-of-the-art for these problems. Note that since the problems we consider are typically APX-hard (including k-means, and even the *unconstrained* version of ℓ_p-subspace approximation for $p > 2$), a running time exponential in k is necessary for our results, assuming the Exponential Time Hypothesis; a discussion in Section 5.2. Before presenting our results, we start with an informal description of the framework.

Overview of Approach. Our approach is based on coresets [76] (also [55, 59, 107] and references therein), but turns out to be different from the standard approach in a subtle yet important way. Recall that a (strong) coreset for an optimization problem $\mathcal{O}$ on set of points $\boldsymbol{A}$ is a subset $\boldsymbol{B}$ such that for any solution for $\mathcal{O}$, the cost on $\boldsymbol{B}$ is approximately the same as the cost on $\boldsymbol{A}$, up to an appropriate scaling. In the formulation of ℓ_p-subspace approximation above, a coreset for a dataset $\boldsymbol{A}$ is a subset $\boldsymbol{B}$ of its columns with $k' \ll n$ columns, such that for all k-dimensional subspaces, each defined by some $\boldsymbol{P}$, $\|\boldsymbol{B} - \boldsymbol{P}\boldsymbol{B}\|_{2,p}^p \approx \|\boldsymbol{A} - \boldsymbol{P}\boldsymbol{A}\|_{2,p}^p$, up to scaling. Thus the goal becomes to minimize the former quantity.

In the standard coreset approach, first a coreset is obtained, and then a problem-specific enumeration procedure is used to find a near optimal solution $\boldsymbol{P}$. For example, for the k-means clustering objective, one can consider all the k-partitions of the points in the coreset $\boldsymbol{B}$; each partition leads to a set of candidate centers, and the best of these candidate solutions will be an approximate solution to the full instance. Similarly for (unconstrained) ℓ_p-subspace approximation, one observes that for an optimal solution, the columns of $\boldsymbol{P}$ must lie in the span of the vectors of $\boldsymbol{B}$, and thus one can enumerate over the combinations of the vectors of $\boldsymbol{B}$. Each combination gives a candidate $\boldsymbol{P}$, and the best of these candidate solutions is an approximate solution to the full instance.

However, this approach does not work in general for constrained subspace approximation. To see this, consider the very simple constraint of having the columns of $\boldsymbol{P}$ coming from some given subspace S. Here, the coreset for ℓ_p-subspace approximation on $\boldsymbol{A}$ will be some set $\boldsymbol{B}$ that is

"oblivious" of the subspace S. Thus, enumerating over combinations of $\boldsymbol{B}$ may not yield any vectors in S!

Our main idea is to avoid enumeration over candidate solutions, but instead, we view the solution (the matrix $\boldsymbol{P} \in \mathbb{R}^{d \times k}$) as simply a set of variables. We then note that since the goal is to use $\boldsymbol{P}$ to approximate $\boldsymbol{B}$, there must be some combination of the vectors of $\boldsymbol{P}$ (equivalently, a set of k coefficients) that approximates each vector a_i in $\boldsymbol{B}$. If the coreset size is k', there are only $k \cdot k'$ coefficients in total, and we can thus hope to enumerate these coefficients in time $\exp(k \cdot k')$. For every given choice of coefficients, we can then solve an optimization problem to find the optimal $\boldsymbol{P}$. For the constraints we consider (including the simple example above), this problem turns out to be convex, and can thus be solved efficiently!

This simple idea yields ε-additive approximation guarantees for a range of problems. We then observe that in specific settings of interest, we can obtain $(1 + \varepsilon)$-multiplicative approximations by avoiding guessing of the coefficients. In these settings, once the coefficients have been guessed, there is a *closed form* for the optimal basis vectors, in the form of low degree polynomials of the coefficients. We can then use the literature on solving polynomial systems of equations (viewing the coefficients as variables) to obtain algorithms that are more efficient than guessing. The framework is described more formally in Section 5.3.

We believe our general technique of using coresets to reduce the number of *coefficients* needed in order to turn a constrained non-convex optimization problem into a convex one, may be of broader applicability. We note it is fundamentally different than the "guess a sketch" technique for variable reduction in [16, 17, 132, 157] and the techniques for reducing variables in non-negative matrix factorization [141]. To support this statement, the guess a sketch technique requires the existence of a small sketch, and consequently has only been applied to approximation with entrywise p-norms for $p \leq 2$ and weighted variants [16, 132, 157], whereas our technique applies to a much wider family of norms.

Relation to Prior Work. We briefly discuss the connection to prior work on binary matrix factorization using coresets. The work of [185] addresses binary matrix factorization by constructing a strong coreset that reduces the number of distinct rows via importance sampling, leveraging the discrete structure of binary inputs. Our framework generalizes these ideas to continuous settings: we use strong coresets not merely to reduce distinct rows, but to reduce the number of variables in a polynomial system for solving continuous constrained optimization problems. This enables us to extend the approach to real-valued matrices and to more general loss functions. Overall, our framework can be seen as a generalization and unification of prior coreset-based "guessing" strategies, adapting them to significantly broader settings.

Applications. We apply our framework to the following applications. Each of these settings can be viewed as subspace approximation with a constraint on the subspace (i.e., on the projection matrix), or on properties of the associated basis vectors. Below we describe these applications, mention how they can be formulated as Constrained Subspace Approximation, and state our results for them. See Table 5.1 for a summary.

Table 5.1: Summary of the upper bound results we get using our framework. In the approximation column, we use superscripts $*, +, \dagger$ to represent multiplicative, additive, or multiplicative-additive approximation respectively. In the prior work column, we use tilde ($\sim$) to indicate no known theoretical guarantees (only heuristics), and hyphen ($-$) to specify that the problem is new.

Problem	Running Time	Approx.	Prior Work
PC-ℓ_p-Subspace Approx.	$(\frac{\kappa}{\varepsilon})^{\mathrm{poly}(\frac{k}{\varepsilon})} \cdot \mathrm{poly}(n)$ (5.4.6)	$\big(O(\varepsilon p) \cdot \|\boldsymbol{A}\|_{p,2}^p\big)^+$	-
	$n^{O(\frac{k^2}{\varepsilon})} \cdot \mathrm{poly}(H)$ (5.4.7)	$(1+\varepsilon)^*$	-
Constrained Subspace Est.	$\mathrm{poly}(n) \cdot (\frac{1}{\delta})^{O(\frac{k^2}{\varepsilon})}$ (5.4.3)	$(1+\varepsilon, O(\delta \cdot \|\boldsymbol{A}\|_F^2))^\dagger$	$\sim$
	$O(\frac{nd\gamma}{\varepsilon})^{O(\frac{k^3}{\varepsilon})}$ (5.4.4)	$(1+\varepsilon)^*$	$\sim$
PNMF	$O(\frac{dk^2}{\varepsilon}) \cdot (\frac{1}{\delta})^{O(\frac{k^2}{\varepsilon})}$ (1.4.1)	$(1+\varepsilon, O(\delta \cdot \|\boldsymbol{A}\|_F^2))^\dagger$	$\sim$
	$(\frac{nd\gamma}{\varepsilon})^{O(\frac{k^3}{\varepsilon})}$ (1.4.2)	$(1+\varepsilon)^*$	$\sim$
k-Means Clustering	$O(nnz(\boldsymbol{A}) + 2^{\widetilde{O}(\frac{k}{\varepsilon})} + n^{o(1)})$ (5.4.17)	$(1+\varepsilon)^*$	[76]
Sparse PCA	$d^{O(\frac{k^3}{\varepsilon^2})} \cdot \frac{k^3}{\varepsilon}$ (5.4.19)	$(\varepsilon \|\boldsymbol{A} - \boldsymbol{A}_k\|_F^2)^+$	[65]

Subspace Approximation with Partition Constraints

First, we study a generalization of ℓ_p-subspace approximation, where we have *partition constraints* on the subspace. More specifically, we consider *PC-ℓ_p-subspace approximation*, where besides the point set $\{a_1, \cdots, a_n\} \in \mathbb{R}^d$, we are given ℓ subspaces $S_1, \cdots, S_\ell$ along with capacities $k_1, \cdots, k_\ell$ such that $\sum_{i=1}^{\ell} k_i = k$. Now the set of valid projections $\mathcal{S}$ is implicitly defined to be the set of projections onto the subspaces that are obtained by selecting k_i vectors from S_i for each $i \in [\ell]$, taking their span.

PC-ℓ_p-subspace approximation can be viewed as a variant of data summarization with "fair representation". Specifically, when S_i is the span of the vectors (or points) in group i, then by setting k_i values properly (depending on the application or the choice of policy makers), PC-ℓ-subspace approximation captures the problem of finding a summary of the input data in which groups are fairly represented. This corresponds to the equitable representation criterion, a popular notion studied extensively in the fairness of algorithms, e.g., clustering [47, 105, 111, 117].[1] We show the following results for PC-subspace approximation:

- First, in Theorem 5.4.6, we show for any $p \geq 1$, an algorithm for PC-ℓ_p-subspace approximation with runtime $(\frac{\kappa}{\varepsilon})^{\mathrm{poly}(k/\varepsilon)} \cdot \mathrm{poly}(n)$ that returns a solution with additive error at most $O(\varepsilon p) \cdot \|\boldsymbol{A}\|_{p,2}^p$, where κ is the condition number of the optimal choice of vectors from the given subspaces.
- For $p = 2$, which is one of the most common loss functions for PC-ℓ_p-subspace approximation, we also present a multiplicative approximation guarantee. There exists a $(1+\varepsilon)$-approximation algorithm running in time $s^{O(k^2/\varepsilon)} \cdot \mathrm{poly}(H)$ where H is the bit complexity

[1] We note that the fair representation definitions differ from those in the line of work on fair PCA and column subset selection [138, 164, 177, 182], where the *objective contributions* (i.e., projection costs) of different groups must either be equal (if possible) or ensure that the maximum incurred cost is minimized. We focus on the question of groups having equal, or appropriately bounded, *representation* among the chosen low-dimensional subspace (i.e., directions). This distinction is also found in algorithmic fairness studies of other problems, such as clustering.

of each element in the input and s is the sum of the dimensions of the input subspaces $S_1, \cdots, S_\ell$, i.e., $s = \sum_{j=1}^{\ell} \dim(S_j)$. The formal statement is in Theorem 5.4.7.

Constrained Subspace Estimation

The *Constrained Subspace Estimation* problem originates from the signal processing community [166], and aims to find a subspace V of dimension k, that best approximates a collection of experimentally measured subspaces $T_1, \cdots, T_m$, with the constraint that it intersects a model-based subspace W in at least a predetermined number of dimensions ℓ, i.e., $\dim(V \cap W) \geq \ell$. This problem arises in applications such as beamforming, where the model-based subspace is used to encode the available prior information about the problem. The paper of [166] formulates and motivates that problem, and further present an algorithm based on a semidefinite relaxation of this non-convex problem, where its performance is only demonstrated via numerical simulation.

We show in Section 5.4.1, that this problem can be reduced to at most k instances of PC-ℓ_2-subspace approximation, in which the number of parts is 2. This will give us the following result for the constrained subspace estimation problem.

- In Corollary 5.4.3, we show a $(1+\varepsilon, \delta\|A\|_F^2)$-multiplicative-additive approximation in time $\mathrm{poly}(n) \cdot (1/\delta)^{O(k^2/\varepsilon)}$.
- In Theorem 5.4.4, we show a $(1+\varepsilon)$ multiplicative approximation in time $O(nd\gamma/\varepsilon)^{O(k^3/\varepsilon)}$ where we assume A has integer entries of absolute value at most γ. We assume that $\gamma = \mathrm{poly}(n)$.

Projective Non-Negative Matrix Factorization

Projective Non-Negative Matrix Factorization (PNMF) [193] (see also [189, 194]) is a variant of Non-Negative Matrix Factorization (NMF), used for dimensionality reduction and data analysis, particularly for datasets with non-negative values such as images and texts. In NMF, a non-negative matrix $\boldsymbol{X}$ is factorized into the product of two non-negative matrices $\boldsymbol{W}$ and $\boldsymbol{H}$ such that $\boldsymbol{X} \approx \boldsymbol{W}\boldsymbol{H}$ where $\boldsymbol{W}$ contains basis vectors, and $\boldsymbol{H}$ represents coefficients. In PNMF, the aim is to approximate the data matrix by projecting it onto a subspace spanned by non-negative vectors, similar to NMF. However, in PNMF, the factorization is constrained to be *projective*.

Formally, PNMF can be formulated as a constrained ℓ_2-subspace approximation as follows: the set of feasible projection matrices $\mathcal{S}$, consists of all matrices that can be written as $\boldsymbol{P} = UU^\top$, where U is a $d \times k$ orthonormal matrix with all non-negative entries.

We show the following results:

- In Theorem 1.4.1, we show a $(1+\varepsilon, \delta\|A\|_F^2)$-multiplicative-additive approximation in time $O(dk^2/\varepsilon) \cdot (1/\delta)^{O(k^2/\varepsilon)}$.
- In Theorem 1.4.2, we show a $(1+\varepsilon)$ multiplicative approximation in time $(nd\gamma)^{O(k^3/\varepsilon)}$, where we assume A has integer entries of absolute value at most γ.

k-Means Clustering

k-means is a popular clustering algorithm widely used in data analysis and machine learning. Given a set of n vectors $a_1, \cdots, a_n$ and a parameter k, the goal of k-means clustering is to partition these vectors into k clusters $\{C_1, \cdots, C_k\}$ such that the sum of the squared distances of all points to their corresponding cluster center $\sum_{i=1}^{n} \|a_i - \mu_{C(a_i)}\|_2^2$ is minimized, where $C(a_i)$ denotes the cluster that a_i belongs to and $\mu_{C(a_i)}$ denotes its center. It is an easy observation that once the clustering is determined, the cluster centers need to be the centroid of the points in each cluster. It is shown in [53] that this problem is an instance of constrained subspace approximation. More precisely, the set of valid projection matrices are all those that can be written as $\boldsymbol{P} = X_C X_C^\top$, where X_C is a $n \times k$ matrix where $X_C(i,j)$ is $1/\sqrt{|C_j|}$ if $C(a_i) = j$ and 0 otherwise. Note that this is an orthonormal matrix and thus $X_C X_C^\top$ is an orthogonal projection matrix. Further, note that using our language we need to apply the constrained subspace approximation on the matrix $A^\top$, i.e., $\min_{\boldsymbol{P} \in \mathcal{S}} \|A^\top - \boldsymbol{P} A^\top\|_F^2$.

In Theorem 5.4.17, we show a $(1+\varepsilon)$ approximation algorithm for k-means that runs in $O(nnz(\boldsymbol{A}) + 2^{\widetilde{O}(k/\varepsilon)} + n^{o(1)})$ time, whose dependency on k and ε matches that of [76].

Sparse PCA

The goal of Principal Component Analysis (PCA) is to find k linear combinations of the d features (dimensions), which are called principal components, that captures most of the mass of the data. As mentioned earlier, PCA is the subspace approximation problem with $p = 2$. However, typically the obtained principal components are linear combinations of all vectors which makes interpretability of the components more difficult. As such, *Sparse PCA* which is the optimization problem obtained from PCA by adding a sparsity constraint on the principal components have been defined which provides higher data interpretability [27, 40, 65, 102, 195].

Sparse PCA can be formulated as a constrained subspace approximation problem in which the set of projection matrices are constrained to those that can be written as $P = UU^\top$ where U is a $d \times k$ orthonormal matrix such that the total number of non-zero entries in the U is at most s, for a given parameter s.

We give an algorithm that runs in time $d^{O(k^3/\varepsilon^2)}\,(dk^3/\varepsilon + d \log d)$ that computes a $\varepsilon\|\boldsymbol{A} - \boldsymbol{A}_k\|_F^2$ additive approximate solution, which translates to a $(1+\varepsilon)$-multiplicative approximate solution to one formulation the problem (see Theorem 5.4.19 for the exact statement).

Column Subset Selection with Partition Constraint

Column subset selection (CSS) is a popular data summarization technique [4, 29, 53], where given a matrix $\boldsymbol{A}$, the goal is to find k columns in $\boldsymbol{A}$ that best approximates all columns of $\boldsymbol{A}$. Since in CSS, a subset of columns in the matrix are picked as the summary of the matrix $\boldsymbol{A}$, enforcing partition constraints naturally captures the problem of column subset selection with fair representation. More formally, in *column subset selection with partition constraints* (*PC-column subset selection*), given a partitioning of the columns of $\boldsymbol{A}$ into ℓ groups, $\boldsymbol{A}^{(1)}, \cdots, \boldsymbol{A}^{(\ell)}$,

along with capacities $k_1, \cdots, k_\ell$, where $\sum_i k_i = k$, the set of valid subspaces are obtained by picking k_i vectors from $\boldsymbol{A}^{(i)}$, and projecting onto the span of these k columns of $\boldsymbol{A}$.

In Section 5.5, we show that PC-column subset selection is hard to approximate to any factor f in polynomial time, even if there are only two groups, or even when we allow for violating the capacity constraint by a factor of $O(\log n)$ (see Theorem 5.5.4 for the formal statement). This is in sharp contrast with the standard column subset selection problem for which efficient algorithms with tight guarantees are known.

5.2 Preliminaries

We will heavily use standard notations for vector and matrix quantities. For a matrix $\boldsymbol{M}$, we denote by $\boldsymbol{M}_{\cdot,i}$ the ith column of $\boldsymbol{M}$ and by $\boldsymbol{M}_{i,\cdot}$ the ith row. We denote by $\|\boldsymbol{M}\|_F$ the Frobenius norm, which is simply $\sqrt{\sum_{i,j} m_{ij}^2}$, where m_{ij} is the entry in the ith row and jth column of $\boldsymbol{M}$. We also use mixed norms, where $\|\boldsymbol{M}\|_{2,p} = \left(\sum_i \|\boldsymbol{M}_{\cdot,i}\|_2^p\right)^{1/p}$. I.e., it is the ℓ_p norm of the vector whose entries are the ℓ_2 norm of the columns of $\boldsymbol{M}$.

We also use $\sigma_{\min}(\boldsymbol{M})$ to denote the least singular value of a matrix, and $\sigma_{\max}(\boldsymbol{M})$ to denote the largest singular value. The value $\kappa(\boldsymbol{M})$ is used to denote the condition number, which is the ratio of the largest to the smallest singular value.

In analyzing the running times of our algorithms, we will use the following basic primitives, the running times of which we denote as T_0 and T_1 respectively. These are standard results from numerical linear algebra; while there are several improvements using randomization, these bounds will not be the dominant ones in our running time, so we do not optimize them.

Lemma 5.2.1 (SVD Computation; see [89])**.** *Given $\boldsymbol{A} \in \mathbb{R}^{d\times n}$, computing the reduced matrix $\boldsymbol{B}$ as in Lemma 5.3.5 takes time $T_0 := H \cdot \min\{O(nd^2), O(nd \cdot \frac{k}{\varepsilon})\}$, where H is the maximum bit complexity of any element of $\boldsymbol{A}$.*

Lemma 5.2.2 (Least Squares Regression; see [89])**.** *Given $\boldsymbol{A} \in \mathbb{R}^{d\times n}$ and given a target matrix $\boldsymbol{B}$ with r columns, the optimization problem $\min_{\boldsymbol{C}} \|\boldsymbol{B} - \boldsymbol{A}\boldsymbol{C}\|_F^2$ can be solved in time $T_1 := O(nrd^2 \cdot H)$, where H is the maximum bit length of any entry in A, B.*

Remark on the Exponential in k Running Times. In all of our results, it is natural to ask if the exponential dependence on k is necessary. We note that many of the problems we study are APX hard, and thus obtaining *multiplicative* $(1+\varepsilon)$ factors will necessarily require exponential time in the worst case. For problems that generalize ℓ_p-subspace approximation (e.g., the PC-ℓ_p-subspace approximation problem, Section 5.4.2), the works of [95] and [51] showed APX hardness. In these reductions, we in fact have the stronger property that the YES and NO instances differ in objective value by $\frac{1}{\text{poly}(k)} \cdot \|\boldsymbol{A}\|_{2,p}^p$, where $\boldsymbol{A}$ is the matrix used in the reduction. Thus, assuming the Exponential Time Hypothesis, even the additive error guarantee in general requires an exponential dependence on either k or $1/\varepsilon$.

5.3 Framework for Constrained Subspace Approximation

Given a $d \times n$ matrix $\boldsymbol{A}$ and a special collection $\mathcal{S}$ of rank k projection matrices, we are interested in selecting the projection matrix $\boldsymbol{P} \in \mathcal{S}$ that minimizes the sum of projection costs (raised to the p^{th} power) of the columns of $\boldsymbol{A}$ onto $\boldsymbol{P}$. More compactly, the optimization problem is

$$\min_{\boldsymbol{P} \in \mathcal{S}} : \|\boldsymbol{A} - \boldsymbol{P}\boldsymbol{A}\|_{2,p}^p. \tag{CSA}$$

A more geometric and equivalent interpretation is that we have a collection of n data-points $\{a_1, a_2, \ldots, a_n\} \subseteq \mathbb{R}^d$ and we would like to approximate these data points by a subspace while satisfying certain constraints on the subspace:

$$\begin{aligned} \min : & \sum_{i=1}^{n} \|a_i - \widehat{a}_i\|_2^p \\ & \widehat{a}_i \in \text{ColumnSpan}(\boldsymbol{P}) \\ & \boldsymbol{P} \in \mathcal{S}. \end{aligned} \tag{CSA-geo}$$

See Lemma 5.3.2 for a proof of the equivalence. We provide a unified framework to obtain approximately optimal solutions for various special collections of $\mathcal{S}$. In our framework, there are three steps to obtaining an approximate solution to any instance of CSA.

1. **Build a coreset:** Reduce the size of the problem by replacing $\boldsymbol{A}$ with a different matrix $\boldsymbol{B} \in \mathbb{R}^{d \times r}$ with fewer number of columns typically $\text{poly}(k, 1/\varepsilon)$. The property we need to guarantee is that the projection cost is approximately preserved possibly with an additive error $c \geq 0$ independent of $\boldsymbol{P}$:

$$\|\boldsymbol{B} - \boldsymbol{P}\boldsymbol{B}\|_{2,p}^p \in (1, 1+\varepsilon) \cdot \|\boldsymbol{A} - \boldsymbol{P}\boldsymbol{A}\|_{2,p}^p - c \quad \forall \boldsymbol{P} \text{ with rank at most } k. \tag{5.2}$$

   Such a $\boldsymbol{P}$ (for $p = 2$) has been referred to as a *Projection-Cost-Preserving Sketch with one sided error* in [53] . See Definition 5.3.3, Theorem 5.3.4, and Lemma 5.3.5 for results obtaining such a $\boldsymbol{B}$ for various $1 \leq p < \infty$. Lemma 5.3.7 shows that approximate solutions to reduced instances $(\boldsymbol{B}, \mathcal{S})$ satisfying Equation (5.2) are also approximate solutions to the original instance $(\boldsymbol{A}, \mathcal{S})$.

2. **Guess Coefficients:** Since the projection matrix $\boldsymbol{P}$ is of rank k, it can be represented as $\boldsymbol{U}\boldsymbol{U}^\top$ such that $\boldsymbol{U}^\top\boldsymbol{U} = \boldsymbol{I}_k$. Using this, observe that the residual matrix

$$\boldsymbol{B} - \boldsymbol{P}\boldsymbol{B} = \boldsymbol{B} - \boldsymbol{U}(\boldsymbol{U}^\top\boldsymbol{B})$$

   can be represented as $\boldsymbol{B} - \boldsymbol{U}\boldsymbol{C}$ where $\boldsymbol{C} = \boldsymbol{U}^\top\boldsymbol{B}$ is a $\mathbb{R}^{k \times r}$ matrix. The norm of the i^{th} column of $\boldsymbol{C}$ can be bounded by $\|b_i\|_2$ the norm of the i^{th} column of $\boldsymbol{B}$. This allows us to guess every column of $\boldsymbol{C}$ inside a k dimensional ball of radius at most the norm of the corresponding column in $\boldsymbol{B}$. Using a net with appropriate granularity, we guess the optimal $\boldsymbol{C}$ up to an additive error.

3. **Solve:** For every fixed $\boldsymbol{C}$ in the search space above, we solve the constrained regression problem

$$\min_{\boldsymbol{U}\in\mathbb{R}^{d\times k}:\boldsymbol{U}\boldsymbol{U}^\top\in\mathcal{S}} \|\boldsymbol{B}-\boldsymbol{U}\boldsymbol{C}\|_{2,p}^p$$

exactly. If $\widehat{\boldsymbol{C}}$ is the $\boldsymbol{C}$ matrix that induces the minimum cost, and $\widehat{\boldsymbol{U}}$ is the minimizer to the constrained regression problem, we return the projection matrix $\widehat{\boldsymbol{U}}\widehat{\boldsymbol{U}}^\top$.

The following lemma formalizes the framework above and can be used as a black box application for several specific instances of CSA.

Lemma 5.3.1. *Given an instance $(\boldsymbol{A},\mathcal{S})$ of CSA, for $1\le p<\infty$,*

1. *Let T_s be the time taken to obtain a smaller instance $(\boldsymbol{B},\mathcal{S})$ such that the approximate cost property in Equation* (5.2) *is satisfied and the number of columns in $\boldsymbol{B}$ is r.*
2. *Let T_r be the time taken to solve the constrained regression problem for any fixed $\boldsymbol{B}\in\mathbb{R}^{d\times r}$ and $\boldsymbol{C}\in\mathbb{R}^{k\times r}$*

$$\min_{\boldsymbol{U}\in\mathbb{R}^{d\times k}:\boldsymbol{U}\boldsymbol{U}^\top\in\mathcal{S}} \|\boldsymbol{U}\boldsymbol{C}-\boldsymbol{B}\|_{2,p}^p. \tag{5.3}$$

Then for any granularity parameter $0<\delta<1$, we obtain a solution $\boldsymbol{P}\in\mathcal{S}$ such that

$$\|\boldsymbol{A}-\boldsymbol{P}\boldsymbol{A}\|_{2,p}^p \le (1+\varepsilon)\mathrm{OPT}+\Delta \tag{5.4}$$

in time $T_s+T_r\cdot O((1/\delta)^{kr})$.

Here, $\Delta=(1+\varepsilon)\|\boldsymbol{A}\|_{2,p}^p\cdot((1+\delta)^p-1)$ and $\mathrm{OPT}=\min_{\boldsymbol{P}'\in\mathcal{S}}\|\boldsymbol{A}-\boldsymbol{P}'\boldsymbol{A}\|_{2,p}^p$.

Proof. Let the optimal solution to the instance $(\boldsymbol{A},\mathcal{S})$ be $\boldsymbol{P}^*=\boldsymbol{U}^*\boldsymbol{U}^{*\top}$ and let $\boldsymbol{C}^*=\boldsymbol{U}^{*\top}\boldsymbol{B}$. Since the columns of $\boldsymbol{U}^*$ are unit vectors, the norm of the i^{th} column of $\boldsymbol{C}^*$ is at most $\|b_i\|_2$ the norm of the i^{th} column of $\boldsymbol{B}$. We will try to approximately guess the columns of $\boldsymbol{C}^*$ using epsilon nets. For each i, we search for the i^{th} column of $\boldsymbol{C}$ using a $(\|b_i\|_2\cdot\delta)$-net inside a k dimensional ball of radius $\|b_i\|_2$ centered at origin. The size of the net for each column of $\boldsymbol{C}$ is $O((1/\delta)^k)$ and hence the total search space over matrices $\boldsymbol{C}$ has $O((1/\delta)^{kr})$ possibilities.

For each $\boldsymbol{C}$, we solve the constrained regression problem in Equation (5.3). Let $\widehat{\boldsymbol{C}}$ be the matrix for which the cost is minimized and $\widehat{\boldsymbol{U}}$ be the corresponding minimizer to the constrained regression problem respectively. Consider the solution $\widehat{\boldsymbol{P}}=\widehat{\boldsymbol{U}}\widehat{\boldsymbol{U}}^\top$. The cost of this solution on reduced instance $(\boldsymbol{B},\mathcal{S})$ is

$$\|\boldsymbol{B}-\widehat{\boldsymbol{U}}\widehat{\boldsymbol{U}}^\top\boldsymbol{B}\|_{2,p}^p \le \|\boldsymbol{B}-\widehat{\boldsymbol{U}}\widehat{\boldsymbol{C}}\|_{2,p}^p. \tag{5.5}$$

Let $\overline{\boldsymbol{C}}$ be the matrix in the search space such that $\|\overline{\boldsymbol{C}}_{\cdot,i}-\boldsymbol{C}^*_{\cdot,i}\|_2\le\|b_i\|_2\cdot\delta$ for every $i\in[r]$. Using the cost minimality of $\widehat{\boldsymbol{C}}$, we can imply that the above cost is

$$\le \min_{\boldsymbol{U}\in\mathbb{R}^{d\times k}:\boldsymbol{U}\boldsymbol{U}^\top\in\mathcal{S}} \|\boldsymbol{B}-\boldsymbol{U}\overline{\boldsymbol{C}}\|_{2,p}^p \tag{5.6}$$

$$\leq \|\boldsymbol{B} - \boldsymbol{U}^*\overline{\boldsymbol{C}}\|_{2,p}^p. \tag{5.7}$$

It remains to upper bound the difference $\Delta = \|\boldsymbol{B} - \boldsymbol{U}^*\overline{\boldsymbol{C}}\|_{2,p}^p - \|\boldsymbol{B} - \boldsymbol{U}^*\boldsymbol{C}^*\|_{2,p}^p$. If we let $b_i^* := (\boldsymbol{U}^*\boldsymbol{C}^*)_{\cdot,i}$ and $\overline{b}_i := (\boldsymbol{U}^*\overline{\boldsymbol{C}})_{\cdot,i}$ for $i \in [r]$, then

$$\Delta = \sum_{i=1}^{r} \left(\|b_i - \overline{b}_i\|_2^p - \|b_i - b_i^*\|_2^p \right). \tag{5.8}$$

Using the fact that $\|\overline{\boldsymbol{C}}_{\cdot,i} - \boldsymbol{C}^*_{\cdot,i}\|_2 \leq \|b_i\|_2 \cdot \delta$, we know that

$$\|\overline{b}_i - b_i^*\|_2 = \|\boldsymbol{U}^*(\overline{\boldsymbol{C}}_{\cdot,i} - \boldsymbol{C}^*_{\cdot,i})\|_2 \leq \|\overline{\boldsymbol{C}}_{\cdot,i} - \boldsymbol{C}^*_{\cdot,i}\|_2 \leq \|b_i\|_2 \cdot \delta. \tag{5.9}$$

This implies that each error term

$$\begin{aligned}
\Delta_i &:= \|b_i - \overline{b}_i\|_2^p - \|b_i - b_i^*\|_2^p && (5.10)\\
&\leq (\|b_i - b_i^*\|_2 + \|b_i^* - \overline{b}_i\|_2)^p - \|b_i - b_i^*\|_2^p && \text{(Triangle inequality)}\\
&\leq (\|b_i - b_i^*\|_2 + \|b_i\| \cdot \delta)^p - \|b_i - b_i^*\|_2^p && (\|\overline{b}_i - b_i^*\|_2 \leq \|b_i\|_2 \cdot \delta)\\
&\leq \|b_i\|_2^p \cdot ((1+\delta)^p - 1). && ((x+\delta)^p - x^p \text{ is increasing in } [0,1],\ \|b_i - b_i^*\|_2 \leq \|b_i\|_2)
\end{aligned}$$

Summing up, the total error Δ is at most $\|\boldsymbol{B}\|_{2,p}^p \cdot ((1+\delta)^p - 1) = O(\delta p) \cdot \|\boldsymbol{B}\|_{2,p}^p$ for $\delta \leq 1/p$. This implies that

$$\|\boldsymbol{B} - \widehat{\boldsymbol{P}}\boldsymbol{B}\|_{2,p}^p \leq \|\boldsymbol{B} - \boldsymbol{P}^*\boldsymbol{B}\|_{2,p}^p + \|\boldsymbol{B}\|_{2,p}^p \cdot ((1+\delta)^p - 1) \tag{5.11}$$

Using the property of $\boldsymbol{B}$ from Equation (5.2), we can imply that

$$\|\boldsymbol{A} - \widehat{\boldsymbol{P}}\boldsymbol{A}\|_{2,p}^p \leq (1+\varepsilon)\|\boldsymbol{A} - \boldsymbol{P}^*\boldsymbol{A}\| + \|\boldsymbol{B}\|_{2,p}^p \cdot ((1+\delta)^p - 1). \tag{5.12}$$

setting $\boldsymbol{P} = 0$ in Equation (5.2) and using the fact that $c \geq 0$ gives $\|\boldsymbol{B}\|_{2,p}^p \leq (1+\varepsilon)\|\boldsymbol{A}\|_{2,p}^p$. Plugging this in the equation above gives

$$\|\boldsymbol{A} - \widehat{\boldsymbol{P}}\boldsymbol{A}\|_{2,p}^p \leq (1+\varepsilon)\|\boldsymbol{A} - \boldsymbol{P}^*\boldsymbol{A}\| + (1+\varepsilon)\|\boldsymbol{A}\|_{2,p}^p \cdot ((1+\delta)^p - 1) \tag{5.13}$$

The total time taken by the algorithm is $T_s + T_r \cdot O((1/\delta)^{kr})$. □

Lemma 5.3.2. *The mathematical programs CSA and CSA-geo equivalent to the following "constrained factorization" problem:*

$$\min_{\boldsymbol{U}\boldsymbol{U}^\top \in \mathcal{S},\, \boldsymbol{H} \in \mathbb{R}^{d \times n}} \|\boldsymbol{A} - \boldsymbol{U}\boldsymbol{H}\|_{2,p}^p. \tag{CSA-fac}$$

Proof. First, we will prove the equivalence between CSA and CSA-fac.

1. The easier direction to see is $\min_{\boldsymbol{U}\boldsymbol{U}^\top \in \mathcal{S},\, \boldsymbol{H} \in \mathbb{R}^{d \times n}} \|\boldsymbol{A} - \boldsymbol{U}\boldsymbol{H}\|_{2,p}^p \leq \min_{\boldsymbol{U}\boldsymbol{U}^\top \in \mathcal{S}} \|\boldsymbol{A} - \boldsymbol{U}\boldsymbol{U}^\top\boldsymbol{A}\|_{2,p}^p$ because setting $\boldsymbol{H} = \boldsymbol{U}^\top\boldsymbol{A}$ in CSA-fac gives CSA.

2. For the other direction, it suffices to show that for any fixed choice of $\boldsymbol{U}$ such that $\boldsymbol{U}\boldsymbol{U}^\top \in \mathcal{S}$, an optimal choice of $\boldsymbol{H}$ is $\boldsymbol{U}^\top \boldsymbol{A}$. In order to see this, observe that the problem

$$\min_{\boldsymbol{H}} \|\boldsymbol{A} - \boldsymbol{U}\boldsymbol{H}\|_{2,p}^p = \min_{\boldsymbol{H}} \sum_{i=1}^{n} \|a_i - \boldsymbol{U}h_i\|_2^p \tag{5.14}$$

where a_i and h_i are the i^{th} columns of $\boldsymbol{A}$ and $\boldsymbol{H}$ respectively. Since the cost function decomposes into separate problems for each column, we can push the minimization inside.

$$= \sum_{i=1}^{n} \left(\min_{h_i} \|a_i - \boldsymbol{U}h_i\|_2 \right)^p. \tag{5.15}$$

Using normal equation, the optimal choice for h_i satisfies $\boldsymbol{U}^\top \boldsymbol{U} h_i = \boldsymbol{U}^\top a_i$. Since the columns of $\boldsymbol{U}$ are orthonormal, this implies that $h_i = \boldsymbol{U}^\top a_i$ for each $i \in [n]$ and hence $\boldsymbol{H} = \boldsymbol{U}^\top \boldsymbol{A}$.

Now we show the equivalence between CSA-fac and CSA-geo. Observe that CSA-geo can be re-written as

$$\begin{aligned} \min \sum_{i=1}^{n} \|a_i - \widehat{a}_i\|_2^p \\ \widehat{a}_i \in \text{ColumnSpan}(\boldsymbol{U}) \\ \boldsymbol{U}\boldsymbol{U}^\top \in \mathcal{S}. \end{aligned}$$

Because the column span of $\boldsymbol{P} = \boldsymbol{U}\boldsymbol{U}^\top$ is identical to the column span of $\boldsymbol{U}$. Replacing $\widehat{a}_i \in \text{ColumnSpan}(\boldsymbol{U})$ by $\widehat{a}_i = \boldsymbol{U}h_i$ gives CSA-fac. □

Definition 5.3.3 (Strong coresets; as defined in [186])**.** Let $1 \leq p < \infty$ and $0 < \varepsilon < 1$. Let $\boldsymbol{A} \in \mathbb{R}^{d \times n}$. Then, a diagonal matrix $\boldsymbol{S} \in \mathbb{R}^{n \times n}$ is a $(1 \pm \varepsilon)$ strong coreset for ℓ_p subspace approximation if for all rank k projection matrices $\boldsymbol{P}_F$, we have

$$\|(\boldsymbol{I} - \boldsymbol{P}_F)\boldsymbol{A}\boldsymbol{S}\|_{2,p}^p \in (1 \pm \varepsilon)\|(\boldsymbol{I} - \boldsymbol{P}_F)\boldsymbol{A}\|_{2,p}^p. \tag{5.16}$$

The number of non-zero entries $\text{nnz}(\boldsymbol{S})$ of $\boldsymbol{S}$ will be referred to as the size of the coreset.

Theorem 5.3.4 (Theorems 1.3 and 1.4 of [187])**.** *Let $p \in [1, 2) \cup (2, \infty)$ and $\varepsilon > 0$ be given, and let $\boldsymbol{A} \in \mathbb{R}^{d \times n}$. There is an algorithm running in $\widetilde{O}(\text{nnz}(\boldsymbol{A}) + d^\omega)$ time which, with probability at least $1 - \delta$, constructs a strong coreset $\boldsymbol{S}$ that satisfies Definition 5.3.3 and has size:*

$$\text{nnz}(\boldsymbol{S}) = \begin{cases} \frac{k}{\varepsilon^{4/p}} (\log(k/\varepsilon\delta))^{O(1)} & \text{if } p \in [1, 2), \\ \frac{k^{p/2}}{\varepsilon^p} (\log(k/\varepsilon\delta))^{O(p^2)} & \text{if } p \in (2, \infty). \end{cases} \tag{5.17}$$

Remark. *Note that for any $\boldsymbol{S}$ that satisfies the property in Definition 5.3.3, we can scale it up to satisfy $\|(\boldsymbol{I} - \boldsymbol{P}_F)\boldsymbol{A}\boldsymbol{S}\|_{p,2}^p \in (1, 1 + \varepsilon)\|(\boldsymbol{I} - \boldsymbol{P}_F)\boldsymbol{A}\|_{p,2}^p$ matching the condition in Equation* (5.2)*.*

For many of the applications, we have $p = 2$. For this case, the choice of the reduced matrix $\boldsymbol{B}$ that replaces $\boldsymbol{A}$ is simply the matrix of scaled left singular vectors of $\boldsymbol{A}$. More formally,

Lemma 5.3.5. *When $p = 2$, if $\boldsymbol{A} = \sum_{i=1}^{n} \sigma_i p_i q_i^\top$ be the singular value decomposition of $\boldsymbol{A}$ (where σ_i is the i^{th} largest singular value and $p_i \in \mathbb{R}^d, q_i \in \mathbb{R}^n$ are the left singular vector and right singular vector corresponding to σ_i), then $\boldsymbol{B} = \sum_{i=1}^{r} \sigma_i p_i q_i^\top$ satisfies Equation* (5.2) *for $r = k + k/\varepsilon$.*

Proof. For any two arbitrary projection matrices $\boldsymbol{P}$ and $\boldsymbol{P}'$ of rank $\leq k$, consider the difference

$$
\begin{aligned}
&\left(\|\boldsymbol{A} - \boldsymbol{P}\boldsymbol{A}\|_F^2 - \|\boldsymbol{B} - \boldsymbol{P}\boldsymbol{B}\|_F^2\right) - \left(\|\boldsymbol{A} - \boldsymbol{P}'\boldsymbol{A}\|_F^2 - \|\boldsymbol{B} - \boldsymbol{P}'\boldsymbol{B}\|_F^2\right) && (5.18)\\
&= \langle \boldsymbol{A}\boldsymbol{A}^\top, \boldsymbol{I} - \boldsymbol{P}\rangle - \langle \boldsymbol{B}\boldsymbol{B}^\top, \boldsymbol{I} - \boldsymbol{P}\rangle - \langle \boldsymbol{A}\boldsymbol{A}^\top, \boldsymbol{I} - \boldsymbol{P}'\rangle + \langle \boldsymbol{B}\boldsymbol{B}^\top, \boldsymbol{I} - \boldsymbol{P}'\rangle && (5.19)\\
&= \langle \boldsymbol{A}\boldsymbol{A}^\top - \boldsymbol{B}\boldsymbol{B}^\top, \boldsymbol{P}'\rangle - \langle \boldsymbol{A}\boldsymbol{A}^\top - \boldsymbol{B}\boldsymbol{B}^\top, \boldsymbol{P}\rangle && (5.20)\\
&\leq \langle \boldsymbol{A}\boldsymbol{A}^\top - \boldsymbol{B}\boldsymbol{B}^\top, \boldsymbol{P}'\rangle && (\boldsymbol{A}\boldsymbol{A}^\top - \boldsymbol{B}\boldsymbol{B}^\top \succeq 0,\ \boldsymbol{P} \succeq 0)\\
&\leq \sum_{i=r+1}^{r+k} \sigma_i && (\text{rank of } \boldsymbol{P}' \leq k)\\
&\leq k \cdot \sigma_r && (\sigma_r \geq \sigma_{r'},\ r' \geq r)\\
&\leq \frac{k}{r-k} \cdot \left(\sum_{i=k+1}^{r} \sigma_i\right) && (\sigma_r \leq \sigma_{r'},\ r' \leq r)\\
&\leq \frac{k}{r-k}\|\boldsymbol{A} - \boldsymbol{A}_k\|_F^2 = \varepsilon\|\boldsymbol{A} - \boldsymbol{A}_k\|_F^2. && (\|\boldsymbol{A} - \boldsymbol{A}_k\|_F^2 = \sum_{i=k+1}^{d} \sigma_i)
\end{aligned}
$$

If we let $c := \max_{\text{rank}(\boldsymbol{P}) \leq k} \left(\|\boldsymbol{A} - \boldsymbol{P}\boldsymbol{A}\|_F^2 - \|\boldsymbol{B} - \boldsymbol{P}\boldsymbol{B}\|_F^2\right)$, then we have

$$c - \varepsilon\|\boldsymbol{A} - \boldsymbol{A}_k\|_F^2 \leq \|\boldsymbol{A} - \boldsymbol{P}\boldsymbol{A}\|_F^2 - \|\boldsymbol{B} - \boldsymbol{P}\boldsymbol{B}\|_F^2 \leq c$$

for any projection matrix $\boldsymbol{P}$ of rank at most k. This can we re written as

$$\|\boldsymbol{B} - \boldsymbol{P}\boldsymbol{B}\|_F^2 \in (0, \varepsilon) \cdot \|\boldsymbol{A} - \boldsymbol{A}_k\|_F^2 + \|\boldsymbol{A} - \boldsymbol{P}\boldsymbol{A}\|_F^2 - c. \tag{5.21}$$

Using the fact that $\|\boldsymbol{A} - \boldsymbol{A}_k\|_F^2 \leq \|\boldsymbol{A} - \boldsymbol{P}\boldsymbol{A}\|_F^2$, we get

$$\|\boldsymbol{B} - \boldsymbol{P}\boldsymbol{B}\|_F^2 \in (1, 1 + \varepsilon) \cdot \|\boldsymbol{A} - \boldsymbol{P}\boldsymbol{A}\|_F^2 - c.$$

The fact that $c \geq 0$ follows from the fact that

$$
\begin{aligned}
\|\boldsymbol{A} - \boldsymbol{P}\boldsymbol{A}\|_F^2 - \|\boldsymbol{B} - \boldsymbol{P}\boldsymbol{B}\|_F^2 &= \langle \boldsymbol{A}\boldsymbol{A}^\top - \boldsymbol{B}\boldsymbol{B}^\top, \boldsymbol{I} - \boldsymbol{P}\rangle && (5.22)\\
&\geq 0. && (\boldsymbol{A}\boldsymbol{A}^\top - \boldsymbol{B}\boldsymbol{B}^\top \succeq 0,\ \boldsymbol{I} - \boldsymbol{P} \succeq 0)
\end{aligned}
$$

□

Remark 5.3.6. Notice that when $p = 2$, Lemma 5.3.5 proves the condition in Equation (5.21):

$$\|\boldsymbol{B} - \boldsymbol{P}\boldsymbol{B}\|_F^2 \in (0, \varepsilon) \cdot \|\boldsymbol{A} - \boldsymbol{A}_k\|_F^2 + \|\boldsymbol{A} - \boldsymbol{P}\boldsymbol{A}\|_F^2 - c$$

which is stronger than the condition in Equation (5.2).

Lemma 5.3.7. *If $(\boldsymbol{A}, \mathcal{S})$ is an instance of CSA and $\boldsymbol{B} \in \mathbb{R}^{d \times r}$ is a matrix that satisfies Equation* (5.2)*, and*

$$\widehat{\boldsymbol{P}} := \arg\min_{\boldsymbol{P} \in \mathcal{S}} \|\boldsymbol{B} - \boldsymbol{P}\boldsymbol{B}\|_{2,p}^p, \quad \boldsymbol{P}^* := \arg\min_{\boldsymbol{P} \in \mathcal{S}} \|\boldsymbol{A} - \boldsymbol{P}\boldsymbol{A}\|_{2,p}^p, \tag{5.23}$$

then $\widehat{\boldsymbol{P}}$ is an $(1 + \varepsilon)$-approximate solution to the instance $(\boldsymbol{A}, \mathcal{S})$ i.e.,

$$\|\boldsymbol{A} - \widehat{\boldsymbol{P}}\boldsymbol{A}\|_{2,p}^p \leq (1 + \varepsilon)\|\boldsymbol{A} - \boldsymbol{P}^*\boldsymbol{A}\|_{2,p}^p. \tag{5.24}$$

1. *More generally, if $\widehat{\boldsymbol{P}}$ is an approximate solution to $(\boldsymbol{B}, \mathcal{S})$ such that*

   $$\|\boldsymbol{B} - \widehat{\boldsymbol{P}}\boldsymbol{B}\|_{2,p}^p \leq \alpha\|\boldsymbol{B} - \boldsymbol{P}\boldsymbol{B}\|_{2,p}^p + \beta \quad \forall \boldsymbol{P} \in \mathcal{S},$$

   for some $\alpha \geq 1$, $\beta \geq 0$, then we have

   $$\|\boldsymbol{A} - \widehat{\boldsymbol{P}}\boldsymbol{A}\|_{2,p}^p \leq \alpha(1 + \varepsilon)\|\boldsymbol{A} - \boldsymbol{P}^*\boldsymbol{A}\|_{2,p}^p + \beta.$$

2. *For the specific case when $p = 2$, if $\widehat{\boldsymbol{P}}$ is an exact solution to $(\boldsymbol{B}, \mathcal{S})$, then we have*

   $$\|\boldsymbol{A} - \widehat{\boldsymbol{P}}\boldsymbol{A}\|_F^2 \leq \|\boldsymbol{A} - \boldsymbol{P}^*\boldsymbol{A}\|_F^2 + \varepsilon\|\boldsymbol{A} - \boldsymbol{A}_k\|_F^2.$$

Proof. 1. Using the approximate optimality of $\widehat{\boldsymbol{P}}$ for the instance $(\boldsymbol{B}, \mathcal{S})$, we have

$$\|\boldsymbol{B} - \widehat{\boldsymbol{P}}\boldsymbol{B}\|_{2,p}^p \leq \alpha\|\boldsymbol{B} - \boldsymbol{P}^*\boldsymbol{B}\|_{2,p}^p + \beta. \tag{5.25}$$

Using the lower-bound and upper-bound from Equation (5.2) for the LHS and RHS, we get

$$\|\boldsymbol{A} - \widehat{\boldsymbol{P}}\boldsymbol{A}\|_{2,p}^p - c \leq \alpha(1 + \varepsilon)\|\boldsymbol{A} - \boldsymbol{P}^*\boldsymbol{A}\|_{2,p}^p - \alpha c + \beta. \tag{5.26}$$

Since $\alpha \geq 1$ and $c \geq 0$, we get

$$\|\boldsymbol{A} - \widehat{\boldsymbol{P}}\boldsymbol{A}\|_{2,p}^p \leq \alpha(1 + \varepsilon)\|\boldsymbol{A} - \boldsymbol{P}^*\boldsymbol{A}\|_{2,p}^p + \beta. \tag{5.27}$$

2. Using the optimality of $\widehat{\boldsymbol{P}}$ for the instance $(\boldsymbol{B}, \mathcal{S})$ for with $p = 2$, we have

$$\|\boldsymbol{B} - \widehat{\boldsymbol{P}}\boldsymbol{B}\|_F^2 \leq \|\boldsymbol{B} - \boldsymbol{P}^*\boldsymbol{B}\|_F^2. \tag{5.28}$$

Using Remark 5.3.6, we know that $\|\boldsymbol{B} - \boldsymbol{P}\boldsymbol{B}\|_F^2 \in (0, \varepsilon) \cdot \|\boldsymbol{A} - \boldsymbol{A}_k\|_F^2 + \|\boldsymbol{A} - \boldsymbol{P}\boldsymbol{A}\|_F^2 - c$ for any rank k projection matrix $\boldsymbol{P}$ for some $c \geq 0$ independent of $\boldsymbol{P}$ (see Equation (5.21)). Using this, we get

$$\|\boldsymbol{A} - \widehat{\boldsymbol{P}}\boldsymbol{A}\|_F^2 - c \leq \|\boldsymbol{B} - \widehat{\boldsymbol{P}}\boldsymbol{B}\|_F^2 \leq \|\boldsymbol{B} - \boldsymbol{P}^*\boldsymbol{B}\|_F^2 \leq \|\boldsymbol{A} - \boldsymbol{P}^*\boldsymbol{A}\|_F^2 + \varepsilon\|\boldsymbol{A} - \boldsymbol{A}_k\|_F^2 - c.$$

Canceling out the $-c$ gives the inequality we claimed.

□

Lemma 5.3.8 (Lemma 4.1 in [50]). *If $n \times d$ matrix $\boldsymbol{A}$ has integer entries bounded in magnitude by γ, and has rank $\rho \geq k$, then the k^{th} singular value σ_k of $\boldsymbol{A}$ has $|\log \sigma_k| = O(\log(nd\gamma))$ as $nd \to \infty$. This implies that $\|\boldsymbol{A}\|_F/\Delta_k \leq (nd\gamma)^{O(k/(\rho-k))}$ as $nd \to \infty$. Here $\Delta_k := \|\boldsymbol{A} - \boldsymbol{A}_k\|_F$*

5.4 Applications

In this section, we present several applications to illustrate our framework.

5.4.1 Constrained Subspace Estimation [166]

In constrained subspace estimation, we are given a collection of target subspaces $T_1, T_2, \ldots, T_m$ and a model subspace W. The goal is to find a subspace V of dimension k such that $\dim(V \cap W) \geq \ell$ that maximizes the average overlap between the subspace V and $T_1, \ldots, T_m$. More formally, the problem can be formulated as mathematical program:

$$\max : \langle \overline{\boldsymbol{P}}_T, \boldsymbol{P}_V \rangle \tag{CSE-max}$$

$$\dim(V) = k,\ \dim(V \cap W) \geq \ell, \tag{5.29}$$

$$\overline{\boldsymbol{P}}_T = \frac{1}{m} \sum_{i=1}^{m} \boldsymbol{P}_{T_i}, \tag{5.30}$$

$$\boldsymbol{P}_{T_i} \text{ and } \boldsymbol{P}_V \text{ are the projection matrices onto the subspaces } T_i \text{ and } V \text{ respectively.} \tag{5.31}$$

Let us assume that the constraint $\dim(V \cap W) \geq \ell$ is actually an exact constraint $\dim(V \cap W) = \ell$ because we can solve for $k - \ell + 1$ different cases $\dim(V \cap W) = i$ for each $\ell \leq i \leq k$. Since $\overline{\boldsymbol{P}}_T$ is a PSD matrix, let it be $\boldsymbol{A}\boldsymbol{A}^\top$ for some $\boldsymbol{A} \in \mathbb{R}^{d \times d}$. Changing the optimization problem from a maximization problem to a minimization problem, we get

$$\min : \langle \boldsymbol{A}\boldsymbol{A}^\top, \boldsymbol{I} - \boldsymbol{P}_V \rangle = \|\boldsymbol{A} - \boldsymbol{P}_V \boldsymbol{A}\|_F^2 \tag{CSE-min}$$

$$\boldsymbol{P}_V \text{ is the projection matrix onto } V \tag{5.32}$$

$$\dim(V) = k,\ \dim(V \cap W) = \ell. \tag{5.33}$$

Lemma 5.4.1. *The CSE-min problem is a special case of CSA.*

Proof. Setting $p = 2$ and $\mathcal{S}$ as the set of k dimensional projection matrices P_V such that $\dim(V \cap W) = \ell$ in CSA gives CSE-min. □

Let $\boldsymbol{B} \in \mathbb{R}^{d \times r}$, $r = k + k/\varepsilon$ be the reduced matrix obtained as in Lemma 5.3.5. Using Lemma 5.3.7, it is sufficient to focus on the reduced instance with $\boldsymbol{A}$ replaced instead of $\boldsymbol{B}$.

Any subspace V such that $\dim(V) = k,\ \dim(V \cap W) = \ell$ can be represented equivalently as

$$V = \text{Span}(u_1, u_2, \ldots, u_\ell, v_1, v_2, \ldots, v_{k-\ell})$$
$$u_i \in W,\ v_j \in W^\perp \quad \forall i \in [\ell],\ j \in [k - \ell].$$

Using these observations and Lemma 5.3.2, we can focus on the following subspace estimation program

$$\min : \|\boldsymbol{B} - \boldsymbol{U}\boldsymbol{C}\|_F^2 \tag{5.34}$$
$$\boldsymbol{U} \text{ is a orthogonal basis for Span}(u_1, \ldots, u_\ell, v_1, \ldots, v_{k-\ell}) \tag{5.35}$$
$$u_i \in W,\ v_j \in W^\perp \quad \forall i \in [\ell],\ j \in [k-\ell]. \tag{5.36}$$

Since $\boldsymbol{C}$ is unconstrained, we can replace the condition in Equation (5.35) with the much simpler condition $\boldsymbol{U} = [u_1, \ldots, u_\ell, v_1, \ldots, v_\ell]$. This gives

$$\min : \|\boldsymbol{B} - \boldsymbol{U}\boldsymbol{C}\|_F^2 \tag{CSE-min-reduced}$$
$$\boldsymbol{U} = [u_1, \ldots, u_\ell, v_1, \ldots, v_\ell] \tag{5.37}$$
$$u_i \in W,\ v_j \in W^\perp \quad \forall i \in [\ell],\ j \in [k-\ell]. \tag{5.38}$$

Lemma 5.4.2. *For any fixed $\boldsymbol{B} \in \mathbb{R}^{d\times r}$ and $\boldsymbol{C} \in \mathbb{R}^{k\times r}$, the Equation* (CSE-min-reduced) *can be solved exactly in* $\mathrm{poly}(n)$ *time.*

Proof. For fixed $\boldsymbol{B}$ and $\boldsymbol{C}$, the objective is convex quadratic in $\boldsymbol{U}$ and the constraints are linear on $\boldsymbol{U}$. Linear constrained convex quadratic program can be efficiently solved. □

Corollary 5.4.3 (Additive approximation for CSE)**.** *Using Lemma 5.3.1, we can get a subspace V such that* $\dim(V) = k,\ \dim(V \cap W) = \ell$ *and*

$$\|\boldsymbol{A} - \boldsymbol{P}_V \boldsymbol{A}\|_F^2 \le (1+\varepsilon)\mathrm{OPT} + O(\delta\|\boldsymbol{A}\|_F^2)$$

for any choice of $0 < \delta < 1$ *in time* $\mathrm{poly}(n) \cdot (1/\delta)^{O(k^2/\varepsilon)}$.

Lemma 5.3.8 gives a lower bound for OPT when the entries of the input matrix $\boldsymbol{A}$ are integers bounded in magnitude by γ.

Theorem 5.4.4 (Multiplicative approximation for CSE)**.** *Given an instance $(\boldsymbol{A} \in \mathbb{R}^{d\times n}, k, W)$ of constrained subspace estimation with integer entries of absolute value at most γ in $\boldsymbol{A}$, there is an algorithm that obtains a subspace V such that* $\dim(V) = k,\ \dim(V \cap W) = \ell$ *and*

$$\|\boldsymbol{A} - \boldsymbol{P}_V \boldsymbol{A}\|_F^2 \le (1+\varepsilon)\mathrm{OPT}$$

in $O(nd\gamma/\varepsilon)^{O(k^3/\varepsilon)}$ time.

Proof. Using Lemma 5.3.8, we know that $\|\boldsymbol{A}\|_F^2/\|\boldsymbol{A} - \boldsymbol{A}_k\|_F^2 \le (nd\gamma)^{O(k)}$. Setting $\delta = \varepsilon\|\boldsymbol{A} - \boldsymbol{A}_k\|_F^2/\|\boldsymbol{A}\|_F^2 \ge \varepsilon(nd\gamma)^{-O(k)}$ in Corollary 5.4.3 gives the desired time complexity. □

5.4.2 Partition Constrained ℓ_p-Subspace Approximation

We now consider the PC-ℓ_p-subspace approximation problem, which generalizes the subspace approximation and subspace estimation problems.

Definition 5.4.5 (Partition Constrained ℓ_p-Subspace Approximation)**.** In the PC-ℓ_p-subspace approximation problem, we are given a set of target vectors $\{a_1, a_2, \ldots, a_n\} \subseteq \mathbb{R}^d$ as columns of a matrix $\boldsymbol{A} \in \mathbb{R}^{d \times n}$, a set of ℓ subspaces $S_1, \ldots, S_\ell \subseteq \mathbb{R}^d$, and a sequence of capacity constraints $k_1, \cdots, k_\ell$ where $k_1 + \cdots + k_\ell = k$. The goal is to select k vectors in total, k_i from subspace S_i, such that their span captures as much of $\boldsymbol{A}$ as possible. Formally, the goal is to select vectors $\{v_{i,t_i}\}_{i \le \ell, t_i \le k_i}$, such that for every $i \le \ell$, $v_{i,1}, \ldots, v_{i,k_i} \in S_i$, so as to minimize $\sum_{i \in [n]} \| \operatorname{proj}^{\perp}_{\text{span}(\{v_{i,t_i}\}_{i \le \ell, t_i \le k_i})}(a_i)\|_2^p$.

Our results will give algorithms with running times exponential in $\operatorname{poly}(k)$ for PC-ℓ-subspace approximation. Given this goal, we can focus on the setting where $k_i = 1$, since we can replace each S_i in the original formulation with k_i copies of S_i, with a budget of 1 for each copy.

PC-ℓ-subspace approximation with Unit Capacity. Given a set of vectors $\{a_1, a_2, \ldots, a_n\} \subseteq \mathbb{R}^d$ as columns of a matrix $\boldsymbol{A} \in \mathbb{R}^{d \times n}$ and subspaces $S_1, \ldots, S_k \subseteq \mathbb{R}^d$, select a vector $v_i \in S_i$ for $i \in [k]$ in order to minimize $\sum_{i \in [n]} \| \operatorname{proj}^{\perp}_{\text{span}(v_1, \ldots, v_k)}(a_i)\|_2^p$, where $p \ge 1$ is a given parameter. A more compact formulation is

$$\min : \sum_{i=1}^{n} \|a_i - \widehat{a}_i\|_2^p \qquad \text{(PC-}\ell_p\text{-SA-geo)}$$

$$\widehat{a}_i \in \text{Span}(v_1, \ldots, v_k) \quad \forall i \in [n] \qquad (5.39)$$

$$v_j \in S_j \quad \forall j \in [k]. \qquad (5.40)$$

Using Lemma 5.3.2, the two other equivalent formulations are

$$\min : \|\boldsymbol{A} - \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{A}\|_{2,p}^p \qquad \text{(PC-}\ell_p\text{-SA)}$$

$$\boldsymbol{U} \text{ is an orthogonal basis for Span}(v_1, v_2, \ldots, v_k) \qquad (5.41)$$

$$v_i \in S_i \quad \forall i \in [k]. \qquad (5.42)$$

$$\min : \|\boldsymbol{A} - \boldsymbol{V}\boldsymbol{C}\|_{2,p}^p \qquad \text{(PC-}\ell_p\text{-SA-fac)}$$

$$\boldsymbol{V} = [v_1, \ldots, v_k] \qquad (5.43)$$

$$v_i \in S_i \quad \forall i \in [k]. \qquad (5.44)$$

In what follows, we thus focus on the unit capacity version. We can use our general framework to derive an additive error approximation, for any p.

Additive Error Approximation

Theorem 5.4.6. *There exists an algorithm for PC-ℓ_p-subspace approximation with runtime $(\kappa/\varepsilon)^{\operatorname{poly}(k/\varepsilon)} \cdot \operatorname{poly}(n)$ which returns a solution with additive error at most $O(\varepsilon p) \cdot \|A\|_{p,2}^p$, where κ is the condition number of an optimal solution $\boldsymbol{V}^* = [v_1^*, v_2^*, \ldots, v_k^*]$ for the PC-subspace approximation problem PC-ℓ_p-SA-fac.*

The algorithm we present below assumes a given bound on κ, the condition number. In practice, we can search for it via doubling and stop when a sufficiently small approximation error is reached or a certain time complexity is reached. We also note that it may happen that the *optimal* solution uses a large κ, but there is an approximately optimal solution with small κ. In this case, our result can be applied with the smaller κ, and it gives a guarantee relative to the latter solution.

Proof. As a first step, we will find an additive approximation to the smaller instance obtained by replacing the $\boldsymbol{A}$ matrix with the smaller $\boldsymbol{B}$ matrix as in Lemma 5.3.5. Our proof mimics the argument from Lemma 5.3.1, but we need a slight change in the analysis because $\{v_j\}$ are not orthogonal. Note that we can assume without loss of generality that the columns of $\boldsymbol{V}^*$ are unit vectors, $\sigma_{\max}(\boldsymbol{V}^*) \geq 1$ and $\sigma_{\min}(\boldsymbol{V}^*) \leq 1$, and thus $\kappa \geq 1$. Given a bound on κ, the algorithm is simply the following: we first create a δ-net for the Ball of radius κ in $\mathbb{R}^k$, with $\delta = \varepsilon/\kappa$, and for each i, we form a guess for the coefficient vector $\boldsymbol{C}_{\cdot,i}$ as $\|b_i\|_2 \cdot u$, where u is a vector from the net and b_i is the i^{th} column of $\boldsymbol{B}$. For each guess $\widehat{\boldsymbol{C}}$, we solve for $\boldsymbol{V}$ that minimizes $\left\|\boldsymbol{B} - \boldsymbol{V}\widehat{\boldsymbol{C}}\right\|_{2,p}$ subject to $v_j \in S_j$. Note that we can drop the unit vector constraints at this point; this makes the above optimization problem convex (specifically, it is the well-studied problem of ℓ_p regression [2]), which can be solved in polynomial time.

To bound the error, we first note that the optimum coefficients $\boldsymbol{C}^*$ satisfy the condition that for each i,

$$\left\|\boldsymbol{C}^*_{\cdot,i}\right\| \leq \frac{\|b_i\|_2}{\sigma_{\min}(\boldsymbol{V}^*)} \leq \|b_i\|_2 \cdot \kappa.$$

Now suppose we focus on one target vector b_i. By choice, in one of our guessed solutions, say $\overline{\boldsymbol{C}}$, we will have $\left\|\overline{\boldsymbol{C}}_{\cdot,i} - \boldsymbol{C}^*_{\cdot,i}\right\| \leq \|b_i\|_2 \cdot \delta$. Thus, we have

$$\left\|\bar{b}_i - b^*_i\right\|_2 \leq \left\|\boldsymbol{V}^*(\overline{\boldsymbol{C}}_{\cdot,i} - \boldsymbol{C}^*_{\cdot,i})\right\|_2 \leq \sigma_{\max}(\boldsymbol{V}^*) \left\|\overline{\boldsymbol{C}}_{\cdot,i} - \boldsymbol{C}^*_{\cdot,i}\right\|_2 \leq \kappa \cdot \|b_i\|_2 \cdot \delta.$$

Thus, analyzing the error Δ_i as in the proof of Lemma 5.3.1, we obtain

$$\Delta_i \leq \|b_i\|_2^p \cdot \left((1 + \varepsilon/p)^p - 1\right).$$

This yields the desired additive guarantee to the reduced instance. Using the coreset property from Equation (5.2), we know that the cost of the solution we find is at most $(1+\varepsilon)\mathrm{OPT} + O(\varepsilon p) \cdot \|\boldsymbol{A}\|_{2,p}^p$. Using the fact that $\mathrm{OPT} \leq \|\boldsymbol{A}\|_{2,p}^p$ completes the proof. □

Multiplicative Approximation Using Polynomial System Solving

For the special case of $p = 2$, it turns out that we can obtain a $(1+\varepsilon)$-multiplicative approximation, using a novel idea.

As described in our framework, we start by constructing the reduced instance $\boldsymbol{B}, \mathcal{S}$, where $\boldsymbol{B} = \{b_1, b_2, \ldots, b_r\} \subset \mathbb{R}^d$ is a set of target vectors and $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$ is the given collection of subspaces of $\mathbb{R}^d$. We define $\boldsymbol{P}_j$ to be some fixed orthonormal basis for the space S_j. Recall that any solution to PC-ℓ_2-subspace approximation is defined by (a) the vector x_j that expresses the chosen v_j as $v_j = \boldsymbol{P}_j x_j$ (we have one x_j for each $j \in [k]$), and (b) a set of combination

coefficients c_{ij} used to represent the vectors b_i using the vectors $\{v_j\}_{j=1}^k$. We collect the vectors x_j into one long vector $\boldsymbol{x}$ and the coefficients c_{ij} into a matrix $\boldsymbol{C}$.

Theorem 5.4.7. *Let $\boldsymbol{B}, \mathcal{S}$ be an instance of PC-ℓ_2-subspace approximation, where $\boldsymbol{B} = \{b_1, b_2, \ldots b_r\}$, and suppose that the bit complexity of each element in the input is bounded by H. Suppose there exists an (approximately) optimal solution is defined by the pair $(\boldsymbol{x}^*, \boldsymbol{C}^*)$ with bit complexity poly(n, H). There exists an algorithm that runs in time $n^{O(k^2/\varepsilon)} \cdot poly(H)$ and outputs a solution whose objective value is within a $(1+\varepsilon)$ factor of the optimum objective value. We denote $s = \sum_{j=1}^k s_j$ and $s_j = dim(S_j)$; n for this result can be set to $\max(s, d, k/\varepsilon)$.*

Algorithm Overview. Recall that $\boldsymbol{P}_j$ specifies an orthonormal basis for S_j. Let $\boldsymbol{P}_{ij} := c_{ij}\boldsymbol{P}_j$, where c_{ij} are variables. Define $\boldsymbol{P}$ to be the $\mathbb{R}^{rd \times s}$ matrix consisting of $r \times k$ blocks; the $(i,j)^{\text{th}}$ block is $\boldsymbol{P}_{ij}$ and we let $\boldsymbol{x}, \boldsymbol{b}$ be the vectors representing all the x_j, b_i stacked vertically respectively as shown below:

$$\boldsymbol{P} = \left[\begin{array}{c|c|c|c} \boldsymbol{P}_{1,1} & \boldsymbol{P}_{1,2} & \cdots & \boldsymbol{P}_{1,k} \\ \hline \boldsymbol{P}_{2,1} & \boldsymbol{P}_{2,2} & \cdots & \boldsymbol{P}_{2,k} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline \boldsymbol{P}_{r,1} & \boldsymbol{P}_{r,2} & \cdots & \boldsymbol{P}_{r,k} \end{array}\right], \quad \boldsymbol{x} = \left[\begin{array}{c} x_1 \\ \hline x_2 \\ \hline \vdots \\ \hline x_k \end{array}\right], \quad \boldsymbol{b} = \left[\begin{array}{c} b_1 \\ \hline b_2 \\ \hline \vdots \\ \hline b_r \end{array}\right].$$

The problem PC-ℓ_2-subspace approximation can now be expressed as the regression problem:

$$\min_{\boldsymbol{C}, \boldsymbol{x}} : \|\boldsymbol{P}\boldsymbol{x} - \boldsymbol{b}\|_2^2. \tag{5.45}$$

Written this way, it is clear that for any $\boldsymbol{C}$, the optimization problem with respect to $\boldsymbol{x}$ is simply a regression problem. For the sake of exposition, suppose that for the optimal solution $(\boldsymbol{C}^*, \boldsymbol{x}^*)$, the matrix $\boldsymbol{P}$ turns out to have a full column rank (i.e., $\boldsymbol{P}^\top \boldsymbol{P}$ is invertible). In this case, the we can write down the normal equation $\boldsymbol{P}^\top \boldsymbol{P}\boldsymbol{x} = \boldsymbol{P}^\top \boldsymbol{b}$ and solve it using Cramer's rule! More specifically, let $\boldsymbol{D} = \boldsymbol{P}^\top \boldsymbol{P}$ and $\boldsymbol{D}_j^{(i)}$ be the matrix obtained by replacing the i^{th} column in the j^{th} column block of $\boldsymbol{D}$ with the column $\boldsymbol{P}^\top \boldsymbol{b}$ for $j \in [k], i \in [s_j]$. Using Cramer's rule, we have $x_j^{(i)} = \det(\boldsymbol{D}_j^{(i)}) / \det(\boldsymbol{D})$.

The key observation now is that substituting this back into the objective yields an optimization problem over (the variables) $\boldsymbol{C}$. First, observe that using the normal equation, the objective can be simplified as

$$\|\boldsymbol{P}\boldsymbol{x} - \boldsymbol{b}\|_2^2 = \boldsymbol{x}^\top \boldsymbol{P}^\top \boldsymbol{P}\boldsymbol{x} - \boldsymbol{x}^\top \boldsymbol{P}^\top \boldsymbol{b} - \boldsymbol{b}^\top \boldsymbol{P}\boldsymbol{x} + \|\boldsymbol{b}\|^2 = \|\boldsymbol{b}\|^2 - \boldsymbol{b}^\top \boldsymbol{P}\boldsymbol{x}.$$

Suppose t is a real valued parameter that is a guess for the objective value. We then consider the following feasibility problem:

$$\|\boldsymbol{P}\boldsymbol{x} - \boldsymbol{b}\|_2^2 = \|\boldsymbol{b}\|_2^2 - \boldsymbol{b}^\top \boldsymbol{P}\boldsymbol{x} \leq t \tag{5.46}$$

$$\iff \|\boldsymbol{b}\|_2^2 - t \leq \sum_{j \in [k], i \in [s_j]} (\boldsymbol{b}^\top \boldsymbol{P})_j^{(i)} \det(\boldsymbol{D}_j^{(i)}), \quad \det(\boldsymbol{D}) = 1. \tag{5.47}$$

The idea is to solve this feasibility problem using the literature on solving polynomial systems. This leaves two main gaps: guessing t, and handling the case of $\boldsymbol{P}$ not having a full column rank in the optimal solution. We handle the former issue using known quantitative bounds on the solution value to polynomial systems, and the latter using a pre-multiplication with random matrices of different sizes.

Core Solver. We begin by describing the details of solving (5.47), assuming a feasible guess for t, and assuming that $\boldsymbol{P}$ has full column rank. Note that this is an optimization problem in the variables c_{ij}, and so the number of variables is $rk = O(k^2/\varepsilon)$. Furthermore, the degree of the polynomials is $O(s)$, and the bit sizes of all the coefficients is $\text{poly}(n, H)$.

We can thus use a well-known result on solving polynomial systems over the reals:

Theorem 5.4.8 ([23, 158, 159])**.** *Given a real polynomial system $P(x_1, \ldots, x_v)$ having v variables and m polynomial constraints $f_i(x_1, \ldots, x_v)\Delta_i 0$, where $\Delta_i \in \{\geq, =, \leq\}$, where d is the maximum degree of all polynomials, and H is the maximum bit-size of the coefficients of the polynomials, one can find a solution to P (or declare infeasibility) in time $(md)^{O(v)}\text{poly}(H)$.*

Applying this Theorem to our setting, we obtain a running time of $s^{O(rk)} \cdot \text{poly}(H) = n^{O(k^2/\varepsilon)} \cdot \text{poly}(H)$, where H is the maximum bit-size of the entries in the matrices $\boldsymbol{P}_j$ and the target vectors $\boldsymbol{b}$.

Guessing t. The solver step assumes that we are able to guess a feasible value of t. In order to perform a binary search, we need some guarantees on the range of the objective value. First, we note that the value of the objective is in the range $[0, \|\boldsymbol{b}\|_2^2]$, so we have an obvious upper bound. We can also test if the problem is feasible for $t = 0$. If $t = 0$ is infeasible, we need a non-trivial lower bound on t. Fortunately, this problem has been well-studied in the literature on polynomial systems. We will use the following Theorem:

Theorem 5.4.9 ([110])**.** *Let $T = \{x \in \mathbb{R}^v | f_1(x) \geq 0, \ldots, f_\ell(x) \geq 0, f_{\ell+1}(x) = 0, \ldots, f_m(x) = 0\}$ be the feasibility set for a polynomial system, where $f_1, \ldots, f_m \in \mathbb{Z}[x_1, \ldots, x_v]$ are polynomials with degrees bounded by an even integer d and coefficients of absolute value at most G, and let C be a compact connected component of T. Let $g \in \mathbb{Z}[x_1, \ldots, x_v]$ be a polynomial of degree at most d and coefficients of absolute value bounded by H. Then the minimum value that g takes over C if not zero, has absolute value at least*

$$(2^{4-v/2}\tilde{G}d^v)^{-v2^v d^v},$$

where $\tilde{G} = \max\{G, 2v + 2m\}$.

We can use Theorem 5.4.9 to obtain a lower bound on the minimum non-zero value attainable for the polynomial

$$\min_{\boldsymbol{C}} : \|\boldsymbol{b}\|_2^2 - \sum_{j\in[k], i\in[s_j]} (\boldsymbol{b}^\top \boldsymbol{P})_j^{(i)} \det(\boldsymbol{D}_j^{(i)}), \quad \text{over the set defined by } \det(\boldsymbol{D}) = 1. \tag{5.48}$$

Using our assumption that the bit complexity of the (near-) optimal solution $\boldsymbol{C}^*$ is $\Delta = \text{poly}(n, H)$, we can add box constraints for each of the variables, making the feasible set compact. Thus, we have polynomials of degree $O(s)$ defining the constraints and the objective, and the number of variables is $rk = O(k^2/\varepsilon)$. Using Theorem 5.4.9, we have that if the minimum value is non-zero, it must be at least

$$\left(2^{\text{poly}(n,H)} s^v\right)^{-v 2^v s^v}, \text{ where } v = \frac{k^2}{\varepsilon}.$$

This implies that a binary search takes time $O\left((2s)^{k^2/\varepsilon} \text{poly}(n, H) \frac{k^4}{\varepsilon^2} \log s\right)$. Note that this time roughly matches the complexity of the core Solver procedure.

Column Rank of $\boldsymbol{P}$. The algorithm above requires the matrix $\boldsymbol{D} = \boldsymbol{P}^\top \boldsymbol{P}$ to have full rank, for a (near) optimum $\boldsymbol{C}^*$. If this does not hold, the idea will be to search over all the possible values of the rank. One natural idea is to solve the problem for all subsets of the columns $[s]$, but note that this takes time $\exp(s)$, which is much larger than our target running time $\exp(\text{poly}(k))$. We thus perform a randomized procedure: for every guess j for the column rank of $\boldsymbol{P}$, we take a random matrix $R \in \mathbb{R}^{s \times j}$, and consider the regression problem

$$\min_{\boldsymbol{C}, \boldsymbol{x}} \|(\boldsymbol{P}R)\boldsymbol{x} - \boldsymbol{b}\|^2 \leq t. \tag{5.49}$$

Let $\boldsymbol{M}$ be the $\boldsymbol{P}$ matrix corresponding to an optimal solution $\boldsymbol{C}^*$, and suppose j is its rank. In Lemma 5.4.10, we show that if the entries of R are drawn IID from $\mathcal{N}(0, 1)$, then with probability at least $3/4$, the matrix $\boldsymbol{M}R$ has full column rank. Thus, if we were to solve (5.49) with the optimal value of t as our guess using the Solver discussed above, we would obtain an approximately optimal solution.

This leads to the following overall algorithm:

- For $j = 1, 2, \ldots, s$:
  - Sample a random matrix $R \in \mathbb{R}^{s \times j}$ with entries drawn i.i.d. from $\mathcal{N}(0, 1)$.
  - Guess value of t for the formulation (5.49) as discussed above.
  - If the determinant is not identically zero, call the core Solve subroutine and check obtained solution for feasibility.
- Return the solution found as above with the least t.

Note that we can boost the success probability by sampling multiple R for each guess of j. This completes the proof of our result, Theorem 5.4.7, modulo Lemma 5.4.10 which we prove below.

Lemma 5.4.10. *Let $M \in \mathbb{R}^{d' \times s}$ be a matrix of rank j, and $R \in \mathbb{R}^{s \times j}$ be a random matrix with entries drawn i.i.d. from $\mathcal{N}(0, 1)$. Then the rank of MR is equal to j with probability $\geq 3/4$.*

Note that since the ranks are equal, the column spans of M and MR must be the same.

Proof. We will prove a quantitative version of the statement. Suppose we denote the jth largest singular value of M as $\tau = \sigma_j(M)$. We will show that $\sigma_j(MR) \geq \frac{\sigma_j}{4j^{3/2}}$ with probability $\geq 3/4$.

Let $\mathcal{S}$ be the span of the columns of M. For a random x whose entries are drawn from $\mathcal{N}(0,1)$, note that the distribution of Mx is a (non-spherical) Gaussian on $\mathcal{S}$. Moreover the covariance matrix has all eigenvalues $\geq \tau^2$. Let us now consider the matrix MR. We use a leave-one-out argument to show linear independence of its columns. I.e., we claim that every column of MR has a projection of length at least $\frac{\tau}{4j}$ orthogonal to the span of the other columns, with probability at least $3/4$. This implies (e.g., see [163]) that $\sigma_j(MR) \geq \frac{\sigma_j}{4j^{3/2}}$ with probability at least $3/4$.

To see the claim, suppose we condition on all but the ith column of the matrix R, for some $1 \leq i \leq j$. Then by the earlier observation, the ith column of MR (denoted by V_i) is distributed as a non-spherical Gaussian on $\mathcal{S}$ whose covariance matrix has all eigenvalues $\geq \tau^2$. Thus its projection to the space orthogonal to the span of the remaining columns of MR (which are all fixed after conditioning) behaves as a Gaussian with at least one dimension and standard deviation $\geq \tau$. Thus by using the anti-concentration bound for a Gaussian (which states that the probability mass in any $\delta\tau$ sized interval is $\leq \delta$), V_i's projection orthogonal to the span of the other columns is at least $\frac{1}{4j} \cdot \tau$, with probability $1 - \frac{1}{4j}$. We can now take a union bound over all $1 \leq i \leq j$ to obtain a success probability of $3/4$. This completes the proof. □

Remark about precision. The result above assumes that we use infinite precision for R. We now show how to avoid this assumption, with a slight loss in the parameters. Note that the key step in the above argument is showing that conditioned on the randomness in all but the ith column of R, the vector $V_i = \sum_{l=1}^{s} R_{il} M_l$ has a sufficiently large norm in the direction orthogonal to the span of the other columns in MR (that are fixed due to the conditioning). For convenience, let Π be the projector orthogonal to the span of the other columns, and denote $v_l = \Pi M_l$, and $X_l := R_{il}$. By the assumption on the least singular value, for any Π, we have that $\sum_l \|v_l\|^2 \geq \tau^2$. This implies that there exists a coordinate $t \in [d']$ such that $\sum_l v_{lt}^2 \geq \frac{\tau^2}{d'}$. Just focusing on this coordinate, we could note that $\sum_l v_{lt} X_l$ is distributed as a Gaussian and thus conclude that the probability of the coordinate being $< \frac{\tau}{4j\sqrt{d'}}$ is $< 1/4j$. This leads to a slightly weaker (by a $\sqrt{d'}$ factor) bound on the least singular value, with the same success probability. However, this argument is more flexible, it lets us use "discretized" Gaussians.

Lemma 5.4.11. *For any $\delta, \eta \in (0, 1/2)$, there exists a centered distribution $\mathcal{Y}$ with the following properties:*

1. *The support of $\mathcal{Y}$ and the probability masses at each point in the support are all rational numbers of bit length $b = O(\log \frac{s}{\delta\eta})$.*
2. *For all $\{a_i\}_{i=1}^{s}$ with $\sum_i a_i^2 = 1$, if $Y_1, Y_2, \ldots, Y_s$ are independent random variables drawn from $\mathcal{Y}$,*

$$\Pr[|\sum_i a_i Y_i| < \delta] \leq 2\delta + \eta.$$

Proof. The proof goes by a discretization of the Gaussian distribution and a sequence of reductions. First, let X_i be independent random variables distributed as $\mathcal{N}(0,1)$. Let X_i' be independent random variables distributed as a *truncated normal*, the distribution obtained from $\mathcal{N}(0,1)$

by removing the mass at points $> \sqrt{\log(s/\eta)}$ and rescaling. We begin by noting that

$$\Pr[|\sum_i a_i X_i'| < \delta] \leq (1+\eta)\Pr[|\sum_i a_i X_i| < \delta]. \tag{5.50}$$

To see this, let us write $\boldsymbol{X}$ to be the vector $(X_1, X_2, \ldots, X_s)$. If $f(\boldsymbol{X})$ and $f'(\boldsymbol{X})$ denote the probability density functions of the multi-dimensional normal and the truncation version respectively, by the choice of the truncation, we have, for all $\boldsymbol{X}$,

$$f'(\boldsymbol{X}) \leq \left(1 + \frac{\eta}{s}\right)^s f(\boldsymbol{X}) \leq (1+\eta) f(\boldsymbol{X}).$$

Furthermore, if any of the coordinates of $\boldsymbol{X}$ is $> \sqrt{\log(s/\eta)}$, we have $f'(\boldsymbol{X}) = 0$. Next, note that the LHS of (5.50) can be written out as an integral, and every term also appears on the probability on the right, albeit with the measure f' replaced with f. Thus, using the bound above, (5.50) follows.

Next, we discretize the interval $[-\lceil\sqrt{\log(s/\eta)}\rceil, \lceil\sqrt{\log(s/\eta)}\rceil]$ into integral multiples of $1/M$, where M is a parameter we will choose later. To the point i/M, we assign the mass that the truncated Gaussian assigns to the interval $[i - \frac{1}{2M}, i + \frac{1}{2M}]$. We call this discrete distribution $\mathcal{D}$. Let $Y_1, Y_2, \ldots, Y_s$ be IID samples from $\mathcal{D}$. We can write $Y_i = X_i' + Z_i$, where X_i' is drawn from the truncated Gaussian, and $|Z_i| \leq \frac{1}{2M}$. Thus, for $M > \frac{s}{\delta}$, we claim that

$$\Pr[|\sum_i a_i Y_i| < \delta] \leq \Pr[|\sum_i a_i X_i'| < 2\delta].$$

This follows because $|\sum_{i\in[s]} a_i Z_i| \leq \frac{s}{2M} < \delta$. Using the earlier bounds, this implies that

$$\Pr[|\sum_i a_i Y_i| < \delta] \leq (1+\eta)\cdot 2\delta.$$

This almost completes the proof, because we have obtained a discrete distribution with the desired anti-concentration bound. But as such, note that the probability values can require very high precision. This turns out to be easy to correct: we can take $\mathcal{Y}$ to be any distribution on the same support as $\mathcal{D}$ with $d_{TV}(\mathcal{Y}, \mathcal{D}) < \epsilon$, and we can conclude (using a coupling argument and a union bound), that if $W_1, W_2, \ldots, W_s$ are drawn IID from $\mathcal{Y}$,

$$\Pr[|\sum_i a_i W_i| < \delta] \leq \Pr[|\sum_i a_i Y_i| < \delta] + \epsilon s.$$

To complete the argument, we need to ensure that $\epsilon < \frac{\eta}{s}$; this can be achieved using probability values with bit complexity only $O(\log(s/\eta))$, thus completing the proof. □

5.4.3 Projective Non-negative Matrix Factorization

In projective non-negative matrix factorization, the basis matrix $\boldsymbol{U} \in \mathbb{R}^{d\times k}$ is constrained to have non-negative entries. More formally, the mathematical program formulation for Projective Non-negative Matrix Factorization (NMF) is

$$\min : \|\boldsymbol{A} - \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{A}\|_F^2 \tag{NMF}$$

$$\boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}_k,\ \boldsymbol{U} \in \mathbb{R}^{d\times k}_{\geq 0}. \tag{5.51}$$

There is a alternate formulation of NMF that better aligns with the name of the problem and is well suited to apply our framework:

$$\begin{aligned} &\min : \|\boldsymbol{A} - \boldsymbol{W}\boldsymbol{H}\|_F^2 && \text{(NMF-alternate)}\\ &\boldsymbol{W} \in \mathbb{R}^{d\times k}_{\geq 0} \text{ has orthogonal columns .} && (5.52)\end{aligned}$$

Lemma 5.4.12. *The programs NMF and NMF-alternate are equivalent.*

Proof. Using Lemma 5.3.2, we know that NMF is equivalent to

$$\begin{aligned} &\min : \|\boldsymbol{A} - \boldsymbol{U}\boldsymbol{H}\|_F^2\\ &\boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}_k,\ \boldsymbol{U} \in \mathbb{R}^{d\times k}_{\geq 0}.\end{aligned}$$

Since $\boldsymbol{H}$ is unconstrained, it can absorb the normalization of the columns of $\boldsymbol{U}$. This gives

$$\begin{aligned} &\min : \|\boldsymbol{A} - \boldsymbol{W}\boldsymbol{H}\|_F^2\\ &\boldsymbol{W} \in \mathbb{R}^{d\times k}_{\geq 0} \text{ has orthogonal columns}\end{aligned}$$

which is exactly Equation (NMF-alternate). □

Lemma 5.4.13. *The set of matrices*

$$\mathcal{W} := \{\boldsymbol{W} \in \mathbb{R}^{d\times k}_{\geq 0} : \boldsymbol{W} \text{ has orthogonal columns }\} \tag{5.53}$$

is equal to the set

$$\overline{\mathcal{W}} := \{\boldsymbol{W} \in \mathbb{R}^{d\times k}_{\geq 0} : \|\boldsymbol{W}_{i,.}\|_0 \leq 1 \quad \forall i \in [d]\}. \tag{5.54}$$

Proof. For any $\boldsymbol{W} \in \mathcal{W}$, if there exists a row $i \in [d]$ and two distinct indices $j, j' \in [k]$ such that $\boldsymbol{W}_{i,j}, \boldsymbol{W}_{i,j'} \neq 0$, then by the non-negativity constraint, these non zero values are in fact strictly positive. The dot product of the columns $\boldsymbol{W}_{.,j}$ and $\boldsymbol{W}_{.,j'}$ is at least $\boldsymbol{W}_{i,j} \cdot \boldsymbol{W}_{i,j'} > 0$ which contradicts the orthogonality of the columns of $\boldsymbol{W}$. This implies that there is at most one non-zero entry in every row of $\boldsymbol{W}$ which further implies that $\boldsymbol{W} \in \overline{\mathcal{W}}$.

For any $\boldsymbol{W} \in \overline{\mathcal{W}}$, the orthogonality of the columns is straight forward because for any two distinct indices $j, j' \in [k]$, the dot product of the columns $\boldsymbol{W}_{.,j}$ and $\boldsymbol{W}_{.,j'}$ is zero because either of $\boldsymbol{W}_{i,j}, \boldsymbol{W}_{i,j'}$ is equal to zero for every $i \in [d]$. □

Lemma 5.4.14. *For any given $\boldsymbol{B} \in \mathbb{R}^{d\times r}$ and $\boldsymbol{H} \in \mathbb{R}^{k\times r}$, we can solve the program*

$$\begin{aligned} &\min : \|\boldsymbol{B} - \boldsymbol{W}\boldsymbol{H}\|_F^2\\ &\boldsymbol{W} \in \mathbb{R}^{d\times k}_{\geq 0} \text{ has orthogonal columns}\end{aligned}$$

exactly in time $O(dkr)$.

Proof. Using Lemma 5.4.13, we can re-write the optimization problem as

$$\min : \|\boldsymbol{B} - \boldsymbol{W}\boldsymbol{H}\|_F^2$$
$$\|\boldsymbol{W}\|_0 \leq 1 \quad \forall i \in [d],\ \boldsymbol{W} \in \mathbb{R}_{\geq 0}^{d \times k}.$$

Since the rows of $\boldsymbol{W}$ are independent variables, we can decompose the problem into

$$\sum_{i=1}^{d} \min_{w_i \in \mathbb{R}_{\geq 0}^k, \|w_i\|_0 \leq 1} : \|b_i - \boldsymbol{H}^\top w_i\|_2^2$$

where b_i, w_i are the i^{th} columns of $\boldsymbol{B}^\top$ and $\boldsymbol{W}^\top$ respectively. Each problem $\min_{w_i \in \mathbb{R}_{\geq 0}^k, \|w_i\|_0 \leq 1} : \|b_i - \boldsymbol{H}^\top w_i\|_2^2$ can be solved by looking at the k cases for the non-zero entry if w_i. For every choice of non-zero entry, say $j \in [k]$, the resulting minimization problem is $min_{\lambda \geq 0} : \|b_i - \lambda h_j\|_2^2$ where h_j is the j^{th} column of $\boldsymbol{H}^\top$. The optimal choice of λ is $\max(0, \langle b_i, h_j \rangle / \|h_j\|_2^2)$. The only computation we had to do is to evaluate the dot products between b_i, h_j for $i \in [d], j \in [k]$ which takes $O(dkr)$ time. □

Theorem 1.4.1 (Additive approximation for NMF)**.** *Given an instance $\boldsymbol{A} \in \mathbb{R}^{d \times n}$ of Non-negative matrix factorization, there is an algorithm that computes a $\boldsymbol{U} \in \mathbb{R}_{\geq 0}^{d \times k},\ \boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}_k$ such that*

$$\|\boldsymbol{A} - \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{A}\|_F^2 \leq (1 + \varepsilon) \cdot \mathrm{OPT} + O(\delta \cdot \|\boldsymbol{A}\|_F^2)$$

in time $O(dk^2/\varepsilon) \cdot (1/\delta)^{O(k^2/\varepsilon)}$. For any $0 < \delta < 1$.

Proof. Let $\boldsymbol{U}^*$ be the optimal solution to NMF. This implies that $\boldsymbol{H} = \boldsymbol{U}^{\top *} \boldsymbol{A}$ is an optimal solution to NMF-alternate. We first replace the instance with a smaller instance $\boldsymbol{B}$. Then we search for every row of $\boldsymbol{H}$ exactly as in the proof of Lemma 5.3.1 to obtain a solution $\boldsymbol{U} \in \mathbb{R}_{\geq 0}^{d \times k},\ \boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}_k$ such that

$$\|\boldsymbol{A} - \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{A}\|_F^2 \leq (1 + \varepsilon) \cdot \mathrm{OPT} + O(\delta \cdot \|\boldsymbol{A}\|_F^2)$$

in time $T_0 + T_1 \cdot (1/\delta)^{O(rk)}$. Where T_0 is the time required to obtain $\boldsymbol{B}$ and T_1 is the time required to solve for the optimal $\boldsymbol{W}$ in the program

$$\min : \|\boldsymbol{B} - \boldsymbol{W}\boldsymbol{H}\|_F^2$$
$$\boldsymbol{W} \in \mathbb{R}_{\geq 0}^{d \times k} \text{ has orthogonal columns}$$

We know that $T_1 = O(dkr)$ using Lemma 5.4.14 and $T_0 = O(nrd^2 \cdot H)$ from Lemma 5.2.1. We hide T_0 as it is negligible. □

Theorem 1.4.2 (Multiplicative approximation for NMF)**.** *Given an instance $\boldsymbol{A} \in \mathbb{R}^{d \times n}$ of Non-negative matrix factorization with integer entries of absolute value at most γ in $\boldsymbol{A}$, there is an algorithm that computes a $\boldsymbol{U} \in \mathbb{R}_{\geq 0}^{d \times k},\ \boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}_k$ such that*

$$\|\boldsymbol{A} - \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{A}\|_F^2 \leq (1 + \varepsilon) \cdot \mathrm{OPT}$$

in time $(nd\gamma/\varepsilon)^{O(k^3/\varepsilon)}$.

Proof. Using Lemma 5.3.8, we know that $\|\boldsymbol{A}\|_F^2 / \|\boldsymbol{A} - \boldsymbol{A}_k\|_F^2 \leq (nd\gamma)^{O(k)}$. Setting $\delta = \varepsilon \|\boldsymbol{A} - \boldsymbol{A}_k\|_F^2 / \|\boldsymbol{A}\|_F^2 \geq \varepsilon (nd\gamma)^{-O(k)}$ in Corollary 5.4.3 gives the desired time complexity. □

5.4.4 k-means clustering [53]

In the k-means problem, we are given a collection of data points $a_1, \ldots, a_n \in \mathbb{R}^d$. The objective is to find k-centers $c_1, \ldots, c_k \in \mathbb{R}^d$ and an assignment $\pi : [n] \to [k]$ that minimizes:

$$\sum_{i=1}^{n} \|a_i - c_{\pi(i)}\|_2^2. \tag{k-means}$$

Let $\boldsymbol{A} \in \mathbb{R}^{n \times d}$ and $\boldsymbol{C} \in \mathbb{R}^{k \times d}$ matrices with a_i and c_i as their i^{th} rows respectively (note that this differs from the notation we used for previous applications). Let $\boldsymbol{\Pi} \in \mathbb{R}^{n \times k}$ be the matrix such that $\boldsymbol{\Pi}_{i,j} = \mathbb{1}[\pi(i) = j]$. Using this notation, the k-means problem can be written as

$$\min : \|\boldsymbol{A} - \boldsymbol{\Pi}\boldsymbol{C}\|_F^2 \tag{k-means-matrix}$$
$$\text{Each row of } \boldsymbol{\Pi} \text{ is a standard basis vector.}$$

Observe that k-means-matrix is a special case of NMF-alternate where $\boldsymbol{W}$ is additionally constrained to have all non-zero entries to be equal to 1. Also, the k-means and k-means-matrix correspond to the CSA-geo and CSA-fac formulations of the same problem. The corresponding CSA version is

$$\min : \|\boldsymbol{A} - \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{A}\|_F^2 \tag{k-means-CSA}$$
$$\boldsymbol{U}_{i,j} = 1/\sqrt{\|U_{\cdot,j}\|_0} \quad \forall i \in [n],\, j \in [k].$$

The three main steps in our algorithm are:

1. **Reduction:** The first step is to reduce the number of rows and columns of the target matrix $\boldsymbol{A}$.
   (a) **Columns:** Replace the matrix $\boldsymbol{A}$ with the matrix $\boldsymbol{B}$ as in Lemma 5.3.5. This reduces the number of columns (dimension of the data-points) to $r = k + k/\varepsilon$.
   (b) **Rows:** For any fixed set of centers (selected from the rows of) $\boldsymbol{C}$, the cost induced by the centers is defined as

$$\text{Cost}(\boldsymbol{A}, \boldsymbol{C}) := \sum_{i=1}^{n} \text{dist}(a_i, \boldsymbol{C})^2.$$

   Where $\text{dist}(a, \boldsymbol{C}) := \min_{c \in \boldsymbol{C}} \|a - c\|_2$. A strong coreset for the k-means instance $\boldsymbol{A}$ is a subset $S \subseteq [n]$ of indices and weights w_i corresponding to each index $i \in S$ such that for any set of centers $\boldsymbol{C}$, we have

$$\text{Cost}_{w,S}(\boldsymbol{A}, \boldsymbol{C}) := \sum_{i \in S} w_i \text{dist}(a_i, \boldsymbol{C}) \in (1 \pm \varepsilon) \cdot \text{Cost}(\boldsymbol{A}, \boldsymbol{C}).$$

   Coresets for k-means of optimal size $\widetilde{O}(k\varepsilon^{-2} \min\{\sqrt{k}, \varepsilon^{-2}\})$ are known (See [59] for upper-bound and [107] for matching lower-bound). Any algorithm that efficiently

computes a coreset of size $q = \text{poly}(k/\varepsilon)$ can be used as a black box for our purposes. After using such a coreset, the new formulation is

$$\min : \|\boldsymbol{B} - \boldsymbol{\Pi}\boldsymbol{C}\|_F^2 \qquad \text{(k-means-reduced)}$$
$$\text{Each row of } \boldsymbol{\Pi} \text{ has exactly one non-zero entry equal to } w_i. \qquad (5.55)$$

where $\boldsymbol{B} \in \mathbb{R}^{q\times r},\ \boldsymbol{\Pi} \in \mathbb{R}^{q\times k}$ with their rows indexed by the set S defined by the coreset and the rows of $\boldsymbol{B}$ are the scaled rows of $\boldsymbol{A}$ according to the weight defined by the coreset for that row.

2. **Enumeration:** A naive approach is to simply enumerate all the k^q possible $\boldsymbol{\Pi}$ matrices by choosing the position of the non-zero element in each row. Simply put, we go through all possible k-clusterings of the coreset elements. Optimal choice of centers can be computed as the weighted mean of the coreset elements in each cluster. This allows us to identify the optimal $\boldsymbol{\Pi}$.

   Let $\boldsymbol{\Pi}^*$ be the optimal choice of $\boldsymbol{\Pi}$ in the k-means-reduced program. Using Lemma 5.4.15 and Lemma 5.4.16, we enumerate over the $O(\log n \cdot k \cdot \text{poly}(k/\varepsilon))^{O(k\log k + k/\varepsilon)} = O(k/\varepsilon \cdot \log n)^{\widetilde{O}(k/\varepsilon)}$ number of possible pairs of matrices $\boldsymbol{SB}$ and $\boldsymbol{S\Pi}$. For each such pair, we find the $\boldsymbol{C}$ that minimizes $\|\boldsymbol{SB} - \boldsymbol{S\Pi C}\|_F^2$. For every such $\boldsymbol{C}$, evaluate the cost induced by these centers with the coreset (w, S). Let $\overline{\boldsymbol{C}}$ be the set of centers that has the lowest cost with respect to the coreset from the enumeration described before. The cost induced by this set of centers is

$$\min_{\boldsymbol{\Pi}} \|\boldsymbol{B} - \boldsymbol{\Pi}\overline{\boldsymbol{C}}\|_F^2 \le \min_{\boldsymbol{\Pi}} \|\boldsymbol{B} - \boldsymbol{\Pi}\widehat{\boldsymbol{C}}\|_F^2 \qquad (5.56)$$
$$\le \|\boldsymbol{B} - \boldsymbol{\Pi}^*\widehat{\boldsymbol{C}}\|_F^2 \qquad (5.57)$$
$$\le (1+\varepsilon)\|\boldsymbol{B} - \boldsymbol{\Pi}^*\boldsymbol{C}^*\|_F^2 \qquad (5.58)$$
$$\le (1+\varepsilon)^2 \cdot \text{OPT}. \qquad (5.59)$$

   Using the coreset property, we imply that the cost of the centers $\overline{C}$ on the original instance $\boldsymbol{A}$ is at most $(1+\varepsilon)^3 \cdot \text{OPT}$.

(5.60)

We start with the following known result (see Theorem 38 of [52]).

Lemma 5.4.15. *Given matrices $\boldsymbol{B} \in \mathbb{R}^{q\times r}$ and $\boldsymbol{\Pi}^* \in \mathbb{R}^{q\times k}$, there exists a matrix $\boldsymbol{S} \in \mathbb{R}^{t\times q}$ and such that*

1. *Each row of $\boldsymbol{S}$ contains exactly one positive non-zero element from the set $W = \{2^i : 0 \le i \le N\}$.*
2. *If $\boldsymbol{C}^* = \arg\min_{\boldsymbol{C}\in\mathbb{R}^{k\times r}} \|\boldsymbol{B} - \boldsymbol{\Pi}^*\boldsymbol{C}\|_F^2$ and $\widehat{\boldsymbol{C}} = \arg\min_{\boldsymbol{C}\in\mathbb{R}^{k\times r}} \|\boldsymbol{SB} - \boldsymbol{S\Pi}^*\boldsymbol{C}\|_F^2$, then*

$$\|\boldsymbol{B} - \boldsymbol{\Pi}^*\widehat{\boldsymbol{C}}\|_F^2 \le (1+\varepsilon)\cdot\|\boldsymbol{B} - \boldsymbol{\Pi}^*\boldsymbol{C}^*\|_F^2. \qquad (5.61)$$

3. $t = O(k \log k + k/\varepsilon)$.

Note that [52] do not require the non-zero element of $\boldsymbol{S}$ to come from W. Indeed, it will be proportional to the leverage score. However, note that we can "discretize" the leverage scores (while keeping a factor two approximation to each one), and still obtain all the guarantees that we require. Finally, since the leverage scores add up to the matrix dimension, we have the bound $N = O(\log n)$.

Lemma 5.4.16. *Given a matrix $\boldsymbol{B} \in \mathbb{R}^{q \times r}$, the number of possible matrices*

1. *of the form $\boldsymbol{SB}$ is at most $O(Nq)^\top$.*
2. *of the form $\boldsymbol{S\Pi}$ where $\boldsymbol{\Pi}$ satisfies Equation* (5.55) *is at most $O(Nkq)^\top$.*

where $\boldsymbol{S}$ satisfies property 1 in Lemma 5.4.15.

Proof. Each row of $\boldsymbol{SB}$ is simply a row of $\boldsymbol{B}$ that is scaled by 2^i for some $0 \leq i \leq N$. This leaves Nq choices for each of the t rows of $\boldsymbol{SB}$ which is $((N+1)q)^\top$ possibilities. Each row of $\boldsymbol{S\Pi}$ is a row of $\boldsymbol{\Pi}$ scaled by 2^i for some $0 \leq i \leq N$. The choices to make for each row of $\boldsymbol{S\Pi}$ is a row of $\boldsymbol{\Pi}$ (which includes choices for non-zero element and weight w_j for $j \in [q]$) and a scaling factor from $\boldsymbol{S}$. This leaves $(N+1)kq$ choices for each of the t rows of $\boldsymbol{S\Pi}$. □

Theorem 5.4.17. *Given an instance $\boldsymbol{A} \in \mathbb{R}^{n \times d}$ of k-means, there is an algorithm that computes a $(1+\varepsilon)$-approximate solution to k-means in $O(nnz(\boldsymbol{A}) + 2^{\widetilde{O}(k/\varepsilon)} + n^{o(1)})$ time.*

Proof. The time complexity of the three step procedure is dominated by the enumeration step which takes time $O(\log n \cdot k/\varepsilon)^{\widetilde{O}(k/\varepsilon)}$ time. If $\log n \leq (k/\varepsilon)^2$, then this running time is $O(k/\varepsilon)^{\widetilde{O}(k/\varepsilon)} = 2^{\widetilde{O}(k/\varepsilon)}$. Otherwise, if $\log n \geq (k/\varepsilon)^2$, then the running time is $(\log n)^{\widetilde{O}(\sqrt{\log n})} = n^{o(1)}$. □

5.4.5 Sparse-PCA [65]

The sparse PCA problem is a well-studied variant of PCA in which the components found are required to be sparse. In other words, the basis matrix $\boldsymbol{U} \in \mathbb{R}^{d \times k}$ is constrained to have sparsity requirements. There are two natural ways to formalize this question: the first is by requiring $\boldsymbol{U}$ to have at most s non-zero entries in total. Another is to require the number of non-zero *rows* of $\boldsymbol{U}$ to be bounded by a parameter s. In the popular case of $d = 1$, both of these definitions coincide. Let us focus on the first variant for now.[2] More formally, the mathematical program formulation we consider is

$$\max : \langle \boldsymbol{A}\boldsymbol{A}^\top, \boldsymbol{U}\boldsymbol{U}^\top \rangle \qquad \text{(sparse-PCA-max)}$$

$$\boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}_k, \ \sum_{j \in [k]} \|\boldsymbol{U}_{\cdot,j}\|_0 \leq s. \qquad (5.62)$$

[2]Our result follows via a black-box application of algorithms from [65]; since their algorithms work for both variants, so do our results.

Program sparse-PCA-max can also be formulated as a minimization version

$$\min : \|\boldsymbol{A} - \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{A}\|_F^2 \qquad \text{(sparse-PCA-min)}$$

$$\boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}_k, \ \sum_{j\in[k]} \|\boldsymbol{U}_{\cdot,j}\|_0 \leq s. \tag{5.63}$$

The following theorem from [65] shows how to find an optimal solution to sparse-PCA-max (and hence also to sparse-PCA-min) when $\text{rank}(\boldsymbol{A}\boldsymbol{A}^\top) = \text{rank}(\boldsymbol{A}) = t$.

Theorem 5.4.18 (Theorem 1 in [65])**.** *There is an algorithm that finds an optimal solution to sparse-PCA-max in*

$$O\left(d^{\min\{k,t\}(t^2+t)}\left(\min\{k,t\}dt^2 + d\log d\right)\right), \tag{5.64}$$

where t denotes the rank of the matrix $\boldsymbol{A}$.

Theorem 5.4.19. *Given an instance $(\boldsymbol{A} \in \mathbb{R}^{d\times n}, k, s)$ of sparse-PCA, there is an algorithm that runs in time*

$$O\left(d^{kr^2+kr}\left(dkr^2 + d\log d\right)\right) \tag{5.65}$$

with $r = k + k/\varepsilon$ that computes a $\varepsilon\|\boldsymbol{A} - \boldsymbol{A}_k\|_F^2$ additive approximate solution to both sparse-PCA-max and sparse-PCA-min. This is guaranteed as a $(1+\varepsilon)$-approximate solution to sparse-PCA-min because $\|\boldsymbol{A} - \boldsymbol{A}_k\|_F^2$ is a lower bound to sparse-PCA-min.

Proof. First step is to replace $\boldsymbol{A}$ with the matrix $\boldsymbol{B}$ as in Lemma 5.3.5. This step takes time T_0. Solve for Equation (sparse-PCA-max) exactly using Theorem 5.4.18, with $t = k + k/\varepsilon$. This step takes time $O\left(d^{kr^2+kr}\left(dkr^2 + d\log d\right)\right)$. Using Lemma 5.3.7, the solution obtained is a $(1+\varepsilon)$-approximate solution to sparse-PCA-min. In fact, for $p = 2$, we know that the error is at most $\varepsilon\|\boldsymbol{A} - \boldsymbol{A}_k\|_F^2$ which implies that this also gives an additive approximation of $\varepsilon\|\boldsymbol{A} - \boldsymbol{A}_k\|_F^2$ to both the minimization and maximization versions. Because the objective for the maximization is the negative of the minimization objective added with $\|\boldsymbol{A}\|_F^2$. □

5.5 Hardness of Column Subset Selection with Partition Constraint

In this section, we show that PC-CSS is at least as hard as the well-studied *sparse regression* problem [79, 93, 99, 145]. In particular, our hardness implies that PC-column subset selection remains hard even if the number of groups is only two, or if we allow violating the given partition capacity constraints by a logarithmic factor. First, we define the PC-column subset selection problem and the sparse regression problem formally.

Definition 5.5.1 (Column Subset Selection with a Partition Constraint)**.** In an instance of the *PC-column subset selection* (*PC-CSS*) problem, we are given a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and a partition matroid $\mathcal{P} = ([n] = P_1 \uplus \cdots \uplus P_\ell,\ \mathcal{I})$, $\mathcal{I} = \{S \subseteq [n] : |S \cap P_t| \leq k_t,\ \forall t \in [\ell]\}$ defined on the set of column indices $[n]$.

The objective is to select a (index) subset $S \in \mathcal{I}$ of columns of $\boldsymbol{A}$ in order to minimize the squared projection cost of all the column vectors of $\boldsymbol{A}$ onto the span of the column space induced by the subset of columns corresponding to S

$$\text{cost}_S(\boldsymbol{A}) := \sum_{i \in [n]} \|\text{proj}_{\text{span}^\perp(S)}(a_i)\|_2^2 = \|\boldsymbol{A} - \boldsymbol{A}_S \boldsymbol{A}_S^+ \boldsymbol{A}\|_F^2, \tag{5.66}$$

where a_i is the column vector corresponding to column index i in $\boldsymbol{A}$ and $\boldsymbol{A}_S$ is the matrix with columns from $\boldsymbol{A}$ corresponding to S.

Definition 5.5.2 ((g, h)-Sparse Regression)**.** Given a matrix $\boldsymbol{B} \in \mathbb{R}^{m \times n}$ and a positive integer s, for which there exists an unknown vector $x^* \in \mathbb{R}^n$ such that $\|x^*\|_0 \leq s$ and $\boldsymbol{B}x^* = \mathbb{1}$, the goal is to output an $x \in \mathbb{R}^n$ with $\|x\|_0 \leq s \cdot g(n)$ such that $\|\boldsymbol{B}x - \mathbb{1}\|_2^2 \leq h(m, n)$.

The sparse regression problem is known to be computationally hard. In particular,

Theorem 5.5.3 ([79])**.** *Let $0 < \delta < 1$. If there is a deterministic polynomial time algorithm $\mathcal{A}$ for (g, h)-sparse regression, for which $g(n) = (1 - \delta) \ln n$ and $h(m, n) = m^{1-\delta}$, then* $\text{SAT} \in \text{DTIME}(n^{O(\log \log n)})$.

Next, we prove our main hardness of approximation result for PC-column subset selection.

Theorem 5.5.4. *Assuming* $\text{SAT} \notin \text{DTIME}(n^{O(\log \log n)})$*, the PC-column subset selection problem is hard to approximate to any multiplicative factor f, even in the following special cases:*

(i) The case of $\ell = 2$ groups, where the capacities on all the groups are the same parameter s.

(ii) The case where the capacities on all the groups are the same parameter s, and we allow a solution to violate the capacity by a factor $g(n) = o(\log n)$, where n is the total number of columns in the instance.

Proof. The proof is via a reduction from sparse regression. First, we show a hardness for two groups (part (a) of the Theorem). Consider an instance of sparse regression, given by an $m \times n$ matrix B and parameter s. Now consider a matrix A whose columns are $A_1 \cup A_2$, defined as follows. A_1 is an $(m + s) \times n$ matrix whose ith column is the ith column of B appended with s zeros. A_2 is an $(m + s) \times (s + 1)$ matrix whose columns we denote by $u_1, u_2, \ldots, u_{s+1}$. We set $u_i = C \cdot e_{m+i}$ for $1 \leq i \leq s$, and $u_{s+1} = D \cdot (\mathbf{1} \oplus \mathbf{0}_s)$, for appropriately chosen parameters $C > D$.[3]

Consider any solution that chooses exactly s columns from A_1 and A_2. In the YES case of sparse regression, where there exists an s-sparse x^* with $Bx^* = \mathbf{1}$, by choosing the columns corresponding to the support of x^* from A_1 along with the columns $u_1, \ldots, u_s$ from A_2, we

[3]As is standard, $\mathbf{1} \oplus \mathbf{0}_s$ is simply the all ones vector (here in m dimensions) with s zeros appended.

obtain an approximation error at most $\|A_1\|_F^2$. Consider the NO case of sparse regression. We will choose C large enough, so that even if one of the u_i for $i \leq s$ is not chosen, the error is $\geq C$. Next, suppose all the $\{u_i\}_{i\in[s]}$ are chosen. For any choice of s columns from A_1, the error on the column u_{s+1} is at least $D \cdot h(m,n)$, by assumption. Thus in either case, the approximation error is $\geq \min(C, D \cdot h(m,n))$. We can now choose C, D large enough (e.g., $> f \cdot n\|A_1\|_F^2$), and obtain the desired hardness of approximation.

Next, suppose we are allowed to choose αs columns from each group, for some slack parameter α (assumed to be an integer ≥ 1 and $< g(n)$, where the latter function comes from the hardness for sparse regression). Now let T be a parameter we will choose later (integer ≥ 1), and consider an instance of PC-column subset selection where we have $(T+1)$ groups of vectors (matrices), $A_1, A_2, \ldots, A_{T+1}$, and the vectors (columns) have dimension Tm. We view each column vector as consisting of T blocks of size m. For $1 \leq j \leq T$, the columns of A_j are identical to those of B in the jth block, and zero everywhere else. The matrix A_{T+1} has T columns, denoted $u_1, \ldots, u_T$, where u_j is the vector that has $\mathbf{1}$ in the jth block and zero everywhere else, scaled by parameter D.

As before, in the YES case of sparse regression, the approximation error is $\leq T\|A_1\|_F^2$. In the NO case, consider any solution that chooses at most αs vectors from each A_j. By assumption, the error in the jth block (of u_j) is at least $h(m,n)$, for any vector u_j that is not picked from A_{T+1}. If we set $T > 2\alpha s$, then at least $(T/2)$ of the vectors u_j cannot be picked, and so the total error is at least $D \cdot (T/2)h(m,n)$. Again, we can choose D large enough to obtain the desired hardness. □

This strong hardness of approximation further motivates the study of a relaxed variant, in which the set of vectors in the small-size summary S, rather than being a subset of $\boldsymbol{A}^{(1)}, \cdots, \boldsymbol{A}^{(\ell)}$, are instead required to belong to the *subspaces* spanned by the columns in each group $\boldsymbol{A}^{(1)}, \cdots, \boldsymbol{A}^{(\ell)}$. This is precisely our PC-subspace approximation problem.

5.6 Future Directions

The unconstrained ℓ_2-subspace approximation problem is efficiently solvable and serves as the foundation for many practical applications. This tractability is a notable exception, as the corresponding ℓ_p-subspace problem is computationally hard for any $p \neq 2$.

However, the complexity of the constrained ℓ_2-subspace approximation problems analyzed in this work is not yet understood. A crucial open question is whether these constrained variants retain the algorithmic efficiency of the unconstrained ℓ_2 setting or if they inherit the hardness characteristic of the general ℓ_p problems. Resolving this is essential, as it may reveal that these problems can be solved far more efficiently than is currently known.

Therefore, a primary direction for future research is to determine the precise complexity of constrained ℓ_2-subspace approximation. This requires either designing provably faster algorithms or establishing matching hardness results to close this fundamental gap.

Part III

Explainability

Chapter 6

Explainable Clustering

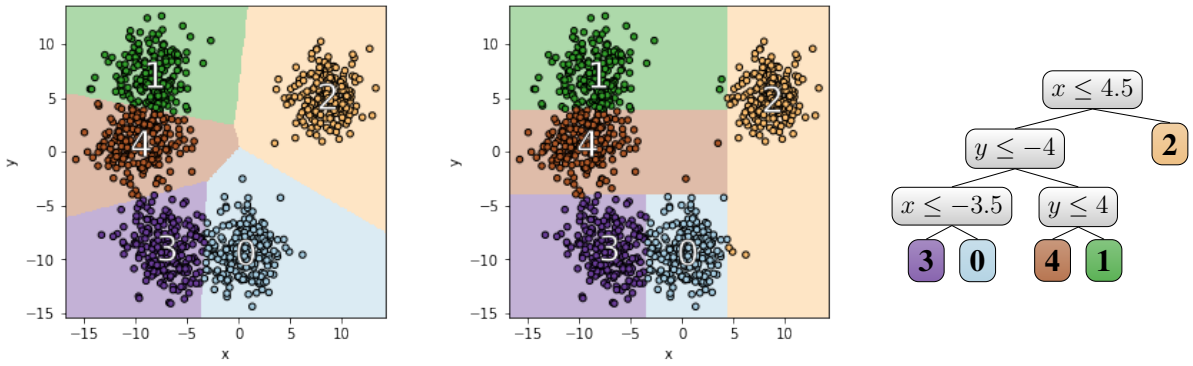
6.1 Introduction

Clustering is a central topic in optimization, machine learning, and algorithm design, with k-medians and k-means being two of the most prominent examples. In recent years, mainly motivated by the impressive but still mysterious advances in machine learning, there has been an increased interest in the transparency and in the explainability of solutions. In the context of clustering, this was formalized in a highly influential paper by Dasgupta et al. [62].

To motivate the concept of explainability, consider the task of clustering n points in $\mathbb{R}^d$ into k clusters. If we solve k-means, the clusters are in general given by a Voronoi diagram where each cluster/cell is defined by the intersection of hyperplanes. Each cluster may be defined using up to $k-1$ hyperplanes, each one of them possibly depending on all d dimensions with arbitrary coefficients. Since the dimensions typically correspond to features (e.g., "age", "weight", and "height" are natural features in a dataset of people), arbitrary linear combinations of these features may be difficult to interpret. To achieve more explainable solutions, we may need to restrict our algorithms to find clusters with simpler descriptions.

The model in [62] achieves explainability in an elegant way resembling the classical notion of decision trees in theoretical computer science. Specifically, a clustering is called *explainable* if it is given by a decision tree, where each internal node splits data points with a *threshold cut* in a single dimension (feature), and each of the k leaves corresponds to a unique cluster. This leads to more explainable solutions already in two dimensions (see, e.g., Figure 6.1); the benefit is even more clear in higher dimensions. Indeed, the binary tree structure gives an easy sequential procedure for classifying points, and since each threshold cut is axis-aligned, there is no linear combinations of features. Moreover, the total number of dimensions/features used to describe the clustering is at most $k-1$, independent of d, which is attractive for high-dimensional data[1].

[1]We remark that dimensionality reduction for k-median and k-means shows that one can reduce the dimension of the data points to $O(\log(k)/\epsilon^2)$ [24, 135]. However, those techniques take arbitrary linear combinations of the original dimensions and therefore destroy explainability.



(a) Optimal 5-means clusters *(b) Tree based 5-means clusters* *(c) Threshold tree*

Figure 6.1: Example from [62]. The optimal 5-means clustering (left) uses combinations of both features. The explainable clustering (middle) uses axis-aligned rectangles summarized by the threshold tree (right).

Explainability is thus a very desirable and appealing property, but the best explainable clustering may have cost much higher than the cost of the best unrestricted clusterings. This trade-off is captured by the *price of explainability*: the loss in cost/quality if we restrict ourselves to explainable clusterings.

In their original paper, Dasgupta et al. [62] gave a greedy algorithm which takes an arbitrary "reference" clustering and produces an explainable clustering from it. It repeatedly adds threshold cuts which separate the centers of the reference clustering until the threshold tree has one leaf for each center of the reference clustering. Since only the points separated in the threshold tree from their closest reference center suffer an increase in cost, their algorithm repeatedly selects a threshold cut that separates the fewest points from their closest reference center. They proved that it outputs an explainable clustering with cost $O(k)$ times higher for the case of k-medians, and $O(k^2)$ times higher for the case of k-means. They also show a lower bound of $\Omega(\ln k)$ for both of these problems.

Since the greedy algorithm's analysis is tight, an alternative strategy was independently proposed by [73, 83, 133]: take random cuts instead! The strategy is especially elegant in the case of k-medians (the distribution of cuts is more complex than uniform in the case of k-means):

> Repeatedly select threshold cuts *uniformly at random* among those that separate centers of the reference clustering.

We refer to this as the Random Thresholds algorithm (see §6.2.1 for a formal description). While the algorithm is easy to describe, its performance guarantee has remained an intriguing question. There are simple instances in which it increases the cost by a factor of $1 + H_{k-1}$, where $H_{k-1} = 1/1 + 1/2 + 1/3 + \ldots + 1/(k-1)$ is the $(k-1)^{th}$ harmonic number (see §6.5), and this was conjectured to be the worst case for the Random Thresholds algorithm [83].

On a high level, a difficulty in analyzing the Random Thresholds algorithm is that it may take prohibitively expensive cuts with a small probability. To avoid this and other difficulties, the results in [73, 83, 133] considered more complex variants that intuitively forbid such expen-

sive cuts. Specifically, [83] gave a variant that outputs a threshold tree whose expected cost increases by at most a $O(\ln^2 k)$ factor, and both [73, 133] obtain a better performance guarantee of $O(\ln k \ln \ln k)$ for their variants of the Random Thresholds algorithm. These results give an exponential improvement over that in [63] but fail to settle the price of explainability, and they leave open the conjectured performance of the natural Random Thresholds algorithm.

Our main results on the price of explainability are (a) to settle this conjecture in the affirmative (i.e., to give a *tight* analysis of the Random Thresholds algorithm), and (b) to show that its price of explainability of $1 + H_{k-1} = (1 + o(1)) \ln k$ is not only asymptotically correct, but also *tight up to lower order terms*: we cannot do much better regardless of the algorithm. Furthermore, we generalize these ideas to show that the same price of explainability holds for k-medians clustering with ℓ_2 norm, provided that explanations can be constructed using general hyperplane cuts instead of only axis-aligned ones. This result is established through two approaches: a direct analysis of a "random hyperplanes" algorithm and an alternative method using metric embeddings (See §6.6 for details).

Theorem 1.5.1 (Upper bound for k-medians)**.** *The price of explainability for k-medians is at most $1 + H_{k-1}$. Specifically, given any reference k-medians clustering, the* Random Thresholds *algorithm outputs an explainable clustering with expected cost at most $1 + H_{k-1}$ times the cost of the reference clustering.*

Theorem 1.5.2 (Lower Bound for k-medians)**.** *There exist instances of k-medians for which any explainable clustering has cost at least $(1 - o(1)) \ln k$ times the cost of the optimal k-medians clustering.*

These results resolve the performance of the Random Thresholds algorithm and the price of explainability for k-medians.

For k-means, we are unable to settle the price of explainability completely, but we make significant progress in closing the gap between known upper and lower bounds. Here, the best upper bound before our work was $O(k \ln k)$ [73] (see also [45] for better guarantees when the input is low-dimensional). Moreover, we know instances where any single threshold cut increases the cost of the clustering by a factor $\Omega(k)$ (see, e.g., [83]), and hence the price of explainability of k-means is at least $\Omega(k)$.

It is tempting to guess that the $O(k \ln k)$ guarantee in [73] is tight, for the following reason. The first lower bound $\Omega(\ln k)$ for k-means in [62] is obtained by arguing that (i) a single threshold cut increases the cost by at least that of the reference clustering and (ii) a threshold tree has height $\Omega(\ln k)$, and so the total cost increases by a constant $\Omega(\ln k)$ times. Since we have examples where any single cut increases the cost by $\Omega(k)$, it is reasonable to hope for more complex instances to combine the two sources of hardness, and lose a $\Omega(k) \cdot \Omega(\ln k)$ factor. However, we prove that this is *not* the case and give an improved upper bound:

Theorem 1.5.3 (Upper bound for k-means)**.** *The price of explainability for k-means is at most $O(k \ln \ln k)$. Specifically, given any reference k-means clustering, there exists an algorithm that outputs an explainable clustering with expected cost at most $O(k \ln \ln k)$ times the reference cost.*

Hence the price of explainability for k-means lies between $\Omega(k)$ and $O(k \ln \ln k)$. We leave the

tight answer as an intriguing open problem. In particular, we conjecture that the lower bound is tight and that it is achieved by the k-means variant of the Random Thresholds algorithm.

Our final contribution is to *study the approximability of explainable clustering*. So far, the literature has mostly focused on settling the price of explainability [45, 62, 73, 83, 121, 133] and its behavior in a bi-criteria setting [134] where the explainable clustering is allowed to form more than k clusters. These algorithms give upper bounds on the approximability of explainable clustering since they are all efficient, and the cost of an optimal unconstrained clustering is a valid lower bound on the best explainable one. Recent work of [18, 120] asked the question: *how well can we approximate the best explainable clustering?* They showed that the problem is APX-hard, but left open the question of whether the problem can be approximated better. Resolving this natural question positively would have the advantage of finding good explainable clusterings for those instances that do admit such clusterings, which is often the experience for more practical instances. Our result shows a surprising hardness for the k-medians and k-means problem.

Theorem 1.5.4 (Approximability)**.** *The explainable k-medians and k-means problems are hard to approximate better than $(1/2 - o(1)) \ln k$, unless $\mathsf{P} = \mathsf{NP}$.*

These results show that we cannot approximate k-medians much better than its price of explainability (unless $\mathsf{P} = \mathsf{NP}$); the approximability for k-means remains tantalizingly open.

6.1.1 Outline and Technical Overview

Upper bounding the performance of the Random Thresholds algorithm. Our main result is the tight analysis (up to lower order terms) of the price of explainability for k-medians. The upper bound of $1 + H_{k-1} = (1 + o(1)) \ln k$ is given by a tight analysis of the natural Random Thresholds algorithm. We now sketch the main ingredients of this analysis. We start with two easy but useful observations: (i) by linearity of expectations, it is sufficient to bound the expected cost of a single point, and (ii) by translation, this point can be assumed to be the origin. We thus reduce the problem to that of analyzing the expected distance from the origin to the last remaining center (i.e., the center in the same leaf of the threshold tree as the origin). We call this process the *Closest Point Process* and define it formally in §6.2.

Algorithm has no better guarantee than $1 + H_{k-1}$. For this discussion, let us make a simplifying assumption that is *not without loss of generality*: the k centers are located on separate axes, so that center i is at $\boldsymbol{e}_i \cdot d_i$, with $d_1 \leq d_2 \leq \ldots \leq d_k$, hence the closest center is at a distance d_1. As cuts are selected uniformly at random, the first cut removes some center i_1 with probability $d_{i_1}/\sum_j d_j$. Conditioned on that, the second cut removes center i_2 with probability $d_{i_2}/\sum_{j \neq i_1} d_j$, and so on. In other words, at each step, a center i is separated from the origin with probability proportional to its distance d_i. For the further special case when $d_2 = d_3 = \ldots = d_k = D$, the expected distance to the last remaining center is:

$$\begin{aligned}\Pr[1 \text{ is last center}] \cdot d_1 + (1 - \Pr[1 \text{ is last center}])\, D &\leq d_1 + (1 - \Pr[1 \text{ is last center}]) \cdot D \\ &= d_1 + \big(1 - \big(1 - \tfrac{d_1}{(k-1)D+1}\big)\big(1 - \tfrac{d_1}{(k-2)D+1}\big) \cdots \big(1 - \tfrac{d_1}{D+1}\big)\big) \cdot D\,.\end{aligned}$$

This is an increasing function of D and tends to $(1 + H_{k-1}) \cdot d_1$ when $D/d_1 \to \infty$, which shows that the Random Thresholds algorithm cannot be better than the conjectured factor of $1 + H_{k-1}$ (see also §6.5.1 for a formal description).

Inductive argument and reduction to worst-case instances. But can this special setting really be the worst-case? Perhaps surprisingly, we prove that this is the case. An inductive argument can help remove the assumption that $d_2 = d_3 = \ldots = d_k$: Since $(1 + H_{k-1}) \cdot d_1$ is the right answer for $k \leq 2$, we can try to proceed inductively on the number of centers to analyze

$$\sum_{i=1}^{k} \Pr[\text{first cut removes center } i] \cdot \mathbb{E}[\text{expected cost of process with centers } [k] \setminus \{i\}] \,.$$

Since each sub-instance in the sum has $k - 1$ centers, we can use the induction hypothesis to bound every term except $i = 1$ in the sum by $(1 + H_{k-2}) \cdot d_1$. To bound the cost of the instance with centers $[k] \setminus \{1\}$, we could proceed based on the following natural observation: the farther away a center is, the smaller probability it has to be the last remaining center, since it is more likely to be cut/removed at each step). This would mean that the expected cost of the process with centers $[k] \setminus \{1\}$ is at most $\frac{d_2+d_3+\ldots+d_k}{k-1}$. And substituting, we get that the expected distance to the last center is at most

$$\Pr[\text{first cut removes center } 1] \cdot \frac{d_2 + d_3 + \ldots + d_k}{k-1} + (1 - \Pr[\text{first cut removes center } 1])\,(1+H_{k-2}) \cdot d_1 \,,$$

which is at most $(1 + H_{k-1}) \cdot d_1$ using that $\Pr[\text{first cut removes center } 1] = \frac{d_1}{d_1+d_2+\ldots+d_k}$.

Several research groups found the above inductive proof for the separate-axis special case, and it was one of the main motivations for the conjectured performance of the Random Thresholds algorithm. To prove it for the general case, it "only" remains to remove the assumption that each center is located on a separate axis. This assumption, however, turns out to be highly non-trivial to overcome. One indication of this difficulty is that, in the general case, there are arbitrary correlations between centers: whether center i is removed impacts the probability that j is removed. This causes most natural monotonicity conditions not to hold anymore. For example, when centers are arbitrarily located, a far center can be more likely to be the last one than a closer one. We overcome these difficulties in a technical proof that manages to show that the worst-case is as above. In this proof, we write the points as a conic combination of cuts, view the cost as a function of this embedding, and naturally try to bound its derivative. This is where the technical challenges appear: since the derivative is also not "well-behaved" we define a better-behaved upper bound called the "pseudo-derivative", and show that this pseudo-derivative is maximized when all points are at the same distance D from the origin (even when they are not along separate axes). We then bound the pseudo-derivative for the non-separate-axis uniform case. This is the technically most challenging part of the chapter, and we present it in §6.5.

A Simpler proof via Competing Exponential Clocks. Interestingly, we can present not just one but two proofs of the correct $(1+o(1)) \ln k$ bound: we give an alternative simpler proof which takes the viewpoint of *competing exponential clocks* (previously used, e.g., for the multiway cut

problem [38, 88, 173]). In the separate-axis case, it boils down to sampling an exponential random variable Z_i with rate d_i for each center i. Two well-known properties of the exponential distribution are that (i) the probability that i "rings first" is proportional to its rate, i.e., $\Pr[Z_i \leq \min_{j \neq i} Z_j] = d_i / \sum_j d_j$ and that (ii) the distribution is memoryless $\Pr[Z_i \geq s + t \mid Z_i \geq t] = \Pr[Z_i \geq s]$. This implies that taking cuts in the order of the random variables $\{Z_i\}_{i \in [k]}$ (until one center remains) is identical to the Closest Point Process. We now analyze the competing exponential clocks as follows. For a center $i \in [k]$ with $i \geq 2$, let Q_i be the probability that i is the last center among the faraway centers $[k] \setminus \{1\}$. Conditioning on this, i is the last center and we pay a distance of d_i instead of d_1 if $Z_1 \leq Z_i$. Now for the probability of $Z_1 \leq Z_i$ to be maximized, Z_i should be as large as possible *in the event when i is last among $[k] \setminus \{1\}$*. So we can upper bound the contribution of center i by considering the upper quantile of the exponential distribution of Z_i with total probability mass Q_i. Now standard calculations show that the total contribution of center i to the cost is $d_1 \left(Q_i - Q_i \ln(Q_i)\right)$. We thus get the upper bound

$$\underbrace{d_1}_{\text{contribution of close center 1}} + \underbrace{d_1 \sum_{i=2}^{k} \left(Q_i - Q_i \ln(Q_i)\right)}_{\text{contribution of far centers}} = d_1(2 + \ln(k-1)) \,,$$

where used that the entropy $\sum_{i=2}^{k} -Q_i \ln(Q_i)$ is at most $\ln(k-1)$. What is particularly nice about this viewpoint is that the analysis does not use the assumption of centers being on separate axes. Indeed, we can define exponential random variables for each cut (as we did in our first proof), and the whole machinery goes through. A small complication arises due to cuts that separate the closest center 1 along with other points from the origin, but we can give a less precise but still tight (up to lower order terms) bound. Apart from achieving the factor $(1 + o(1)) \ln k$, the arguments are also arguably cleaner and easier than even the prior non-tight analyses of the Random Thresholds algorithm. We present these arguments in §6.2.

Lower-Bounding the Price of Explainability. Recall that the $\Omega(\ln k)$ on the price of explainability for k-medians [62] is based on the following idea

1. Select k centers uniformly at random from a hypercube $\{0, 1\}^d$, and
2. Add a 1-ball around each center with d points, one per dimension, giving dk points..

The optimal unconstrained clustering has cost dk, so how expensive is the best explainable clustering? Any pair of centers expect to differ in $d/2$ coordinates, and so by concentration, their distance $\approx d/2$ whp. Furthermore, in a sub-instance with k' centers, any cut separates k' points from their closest center, and these incur cost $\approx d/2$. As the threshold tree has a height of at least $\log_2 k$, the total cost of any explainable clustering can now be seen to be at least $\approx (dk/2) \log_2 k$. While asymptotically tight, the above symmetric construction does not lead to stronger lower bounds than $\frac{1}{2} \log_2 k$. We instead use an asymmetric construction to achieve our tight lower bound of $(1 - o(1)) \ln k$, and it gives us hardness of approximability too!

1. Place a special center at the origin, and take a 1-ball around it giving d points.
2. The remaining centers are located at the characteristic vectors of some carefully chosen subsets of $\{1, \ldots, d\}$, and

3. Finally, add many points colocated with the centers which force any good threshold tree to have one leaf per center.

Now the only way to separate a center from the origin is to employ a threshold cut along a dimension, which corresponds to an element in the set corresponding to that center. Our threshold cuts must thus form a *hitting set* of the set system corresponding to the non-special centers. Furthermore, the number of points separated from their closest center is equal to the size of this hitting set. This tight connection allows us to apply the known results for the hitting set problem, and we get a $(1 - o(1)) \ln k$ lower bound on the price of explainability for k-medians. In addition, the connection together with Feige's landmark paper [74] implies our hardness of approximation results. (Interestingly enough, [18] give a very similar construction, but with different parameters, which only gives them NP-hardness.) We remark that the hardness result for k-means follows from that of k-medians since all points and centers are located on the hypercube, and thus the ℓ_1-distances equal the squared ℓ_2-distances. We present the reduction from hitting set and its implications in §6.3.

Improvements for k-means. Our final result is an $O(k \ln \ln k)$ price of explainability for k-means. We observe that there are two ways to achieve the weaker $O(k \ln k)$ bound. The first transforms the k-means instance into k-medians, but this distorts distances by at most k using the Cauchy-Schwarz inequality; then we lose another $O(\ln k)$ using our analyses above. Another follows the approach of [73], of finding cuts that have a good cost-to-balance ratio. Both these approaches are tight, but we show that they cannot be tight at the same time! I.e., if we lose a factor of $\Omega(k)$ due to Cauchy-Schwarz, then the cuts partition the instance into parts that are a constant factor smaller, and the loss becomes a geometric sum that sums to $O(k)$. A quantitative version of this trade-off gives our result; the details appear in §6.4.

Outline. We present the simpler exponential clocks-based proof for k-medians in §6.2, followed by the matching hardness in §6.3. The result for k-means is in §6.4, followed by the tight $1 + H_{k-1}$ bound for k-medians in §6.5.

6.1.2 Further Related Work

We now discuss some of the related results beyond those mentioned above. Some works consider the effect of the dimension d of the price of explainability. Laber and Murtinho [121] showed an $O(d \ln k)$ price of explainability for k-medians, which was improved by Esfandiari et al. [73] to $O(\min\{d \ln^2 d, \ln k \ln \ln k\})$. Charikar and Hu [45] showed that the price of explainability is at most $k^{1-2/d} \operatorname{poly} \log k$ for k-means, and a lower bound tight up to poly-logarithmic terms. Esfandiari et al. [73] also gave a lower bound of $\Omega(d)$ for k-medians. Frost et al. [81] posed the question of getting better guarantees using more than k clusters; Makarychev and Shan [134] showed how to open $(1 + \delta)k$ centers and get a guarantee of $O(1/\delta \cdot \ln^2 k \ln \ln k)$ for k-means.

The algorithmic problem has received much less attention. Bandyapadhyay et al. [18] gave algorithms that find the best k-medians and k-means clusterings in time $n^{2d} \cdot (dn)^{O(1)}$. They also showed NP-hardness, and $W[2]$-hardness of finding the best explainable clustering; interestingly,

their hardness construction is also based on the hitting set problem and is very similar to ours, but they use a different setting of parameters and hence only infer an NP-hardness. Laber [120] gave an APX-hardness based on a reduction from finding vertex covers in triangle-free graphs. Our result showing a logarithmic hardness essentially settles the question for k-medians.

Both the k-medians and k-means problems have been studied extensively in the unconstrained setting (i.e., without the explainability requirement), both for geometric spaces (see, e.g., [54, 56, 57, 80]) and general metric spaces (see, e.g., [39, 58]). The techniques and algorithms for those settings seem orthogonal to those used for our problems.

6.1.3 Preliminaries and Notation

Given points $\mathcal{X} = \{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^n\} \subseteq \mathbb{R}^d$, a *clustering* $\mathcal{C}$ of $\mathcal{X}$ is a partition of $\mathcal{X}$ into *clusters* $\{C^1, \ldots, C^k\}$. Each cluster C^i is assigned a center $\boldsymbol{\mu}^i$ (giving *distinct* centers $\mathcal{U} = \{\boldsymbol{\mu}^1, \ldots, \boldsymbol{\mu}^k\} \subseteq \mathbb{R}^d$). Let $\pi(\boldsymbol{x})$ be the center $\boldsymbol{\mu}^j \in \mathcal{U}$ corresponding to the cluster C^j containing x, and define the q-norm cost of a clustering $\mathcal{C}$ with centers $\mathcal{U}$ as

$$\mathrm{cost}_q(\pi, \mathcal{U}) = \sum_{\boldsymbol{x} \in \mathcal{X}} \|\boldsymbol{x} - \pi(\boldsymbol{x})\|_q^q. \tag{6.1}$$

The k-medians and k-means costs of a clustering are simply the minimum values for the parameters $q = 1$ and 2, minimized over all possible centers $\mathcal{U}$.

Threshold Cuts and Trees. We call a hyperplane of the form $x_i \leq \theta$ a *threshold cut*, and represent it as (i, θ). A *threshold tree* T is a binary tree with each non-leaf node u corresponding to a threshold cut (i_u, θ_u). Define $B_u \subseteq \mathbb{R}^d$ as the region corresponding to node $u \in T$, where $B_r := \mathbb{R}^d$ for r being the root of T; if nodes $l(u)$ and $r(u)$ are the left and right children of node u, then

$$B_{l(u)} := B_u \cap \{\boldsymbol{x} \mid x_{i_u} \leq \theta_u\} \qquad \text{and} \qquad B_{r(u)} := B_u \cap \{\boldsymbol{x} \mid x_{i_u} > \theta_u\}.$$

Explainable Clusterings. Given points $\mathcal{X}$ and a threshold tree T, the clustering $\mathcal{C}_T$ of $\mathcal{X}$ *explainable* by the threshold tree T is the partition of $\mathcal{X}$ induced by the regions corresponding to leaves in T, i.e., each leaf ℓ of T generates a cluster $C^\ell := \mathcal{X} \cap B_\ell$ of $\mathcal{C}_T$. A clustering $\mathcal{C}$ of $\mathcal{X}$ is said to be an *explainable clustering* if there exists a threshold tree T such that $\mathcal{C} = \mathcal{C}_T$.

For a set of centers $\mathcal{U}$, a threshold tree T *separates* $\mathcal{U}$ if each of the regions corresponding to leaves in T contains exactly one center in $\mathcal{U}$. Let μ^ℓ denote the unique center in the singleton set $\mathcal{U} \cap \mathcal{B}_\ell$ for leaf ℓ in T. For any set of points $\mathcal{X}$, centers $\mathcal{U}$, and a threshold tree T that separates $\mathcal{U}$, each leaf in T corresponds to a cluster C^ℓ in the clustering $\mathcal{C}_T$, and also to a center μ^ℓ. Such a tree induces an assignment $\pi_T : \mathcal{X} \to \mathcal{U}$ from points to centers. With this, we can define

$$\mathrm{cost}_q(T) = \mathrm{cost}_q(\pi_T, \mathcal{U}) = \sum_{\boldsymbol{x} \in \mathcal{X}} \|\boldsymbol{x} - \pi_T(\boldsymbol{x})\|_q^q. \tag{6.2}$$

6.2 Explainable k-medians via Exponential Clocks

We now give a bound of $(1 + o(1)) \ln k$ on the price of explainability for k-medians. This is slightly weaker than the bound of $1 + H_{k-1} \approx \ln k + O(1)$ promised in Theorem 1.5.1, but the proof is simpler and more illuminating. (We give the proof of the tight bound in §6.5.)

6.2.1 The Random Threshold Algorithm and the Closest Point Process

Let us first formalize the Random Thresholds algorithm: given a reference clustering for point set $\mathcal{X}$ which opens centers $\mathcal{U}$ and maps the data points to centers using $\pi : \mathcal{X} \to \mathcal{U}$, we construct a threshold tree T randomly as follows. For simplicity, let $\mathcal{X} \subseteq [a, b]^d$ for some $a, b \in \mathbb{R}$. We start with the trivial threshold tree with the root corresponding to all of $\mathbb{R}^d$. Now while the leaves of T do not give us a separating partition for $\mathcal{U}$, we pick a dimension $i \in [d]$ and a value $\theta \in [a, b]$ independently and uniformly at random. For each leaf u of T, if this threshold cut separates at least one pair of centers which share the region B_u, partition the leaf using the threshold cut. It is easy to see that as long as all the centers in $\mathcal{U}$ are distinct, this process outputs a threshold tree that separates $\mathcal{U}$. The main question is: what is the cost of the resulting explainable clustering $\mathcal{C}_T$, in expectation?

Since the algorithm does not depend on the data points $\mathcal{X}$, and it is invariant under translations and scaling, we can use linearity of expectations and focus on the following simpler problem:

Definition 6.2.1 (Closest Point Process)**.** Given a set of k points $U \subseteq \mathbb{R}^d$, let $\boldsymbol{p}^* := \arg\min_{\boldsymbol{p} \in U} \|\boldsymbol{p}\|_1$ be the point in U closest to the origin. Assume $\|\boldsymbol{p}^*\|_1 = 1$. Run the Random Thresholds algorithm to create a random threshold tree T that separates this point set U. Consider leaf node $u \in T$ whose corresponding region $B_u \subseteq \mathbb{R}^d$ contains the origin, and let $\widehat{\boldsymbol{p}}$ be the unique point of U in this region B_u. Define

$$f(U) := \mathbb{E}[\ \|\widehat{\boldsymbol{p}}\|_1\ ]. \tag{6.3}$$

Finally, define $\alpha(k) := \max_{U:|U|=k} f(U)$.

Lemma 6.2.2 (Focus on Closest Point)**.** *Given a reference clustering $\pi : \mathcal{X} \to \mathcal{U}$, the expected cost of the explainable clustering produced by the Random Thresholds algorithm is*

$$\mathbb{E}[\mathrm{cost}(\pi_T, \mathcal{U})] \leq \alpha(|\mathcal{U}|) \cdot \mathrm{cost}(\pi, \mathcal{U})\,.$$

Therefore, the price of explainability is at most $\alpha(k)$.

Given this reduction (which we prove in Section 6.8.1), the main result of this section is:

Theorem 6.2.3 (Exponential Clocks)**.** *For any set U with k points, $f(U) \leq (1 + o(1)) \ln k$.*

6.2.2 The Exponential Clocks Viewpoint: the Last Point

We now focus on bounding the value $f(U)$ for any point set $U \in \mathbb{R}^d$. We first impose some structure, just for the sake of analysis. Since ℓ_1 metrics can be written as a non-negative sum of cut metrics (see, e.g., [68]), again using the data-obliviousness and translation-invariance of the algorithm we can assume the following without loss of generality (see Section 6.8.1).

1. there are $d = 2^k$ dimensions (one for each subset $S \subseteq U$ of the points), and
2. the instance is specified by non-negative values $\{z_S\}_{S \subseteq U}$ such that for each point $\boldsymbol{p} \in U$, it lies at location

$$\boldsymbol{p}_S := z_S \mathbf{1}(\boldsymbol{p} \in S).$$

Hence the distance of a point $\boldsymbol{p}$ is $\|\boldsymbol{p}\|_1 = \sum_S z_S \mathbf{1}(\boldsymbol{p} \in S) = \sum_{S:\boldsymbol{p} \in S} z_S$.

Given this structure and the focus on $f(U)$, we need to analyze the following process:

> *The Last Point Process.* Start with some set $V \subseteq U$, and empty sequence $\mathcal{S} \leftarrow \langle\rangle$. At each step, pick a set $S \notin \mathcal{S}$ with probability $\frac{z_S}{\sum_{T \notin \mathcal{S}} z_T}$ and add it to the end of $\mathcal{S}$. If $|V \setminus S| \neq \emptyset$, set $V \leftarrow V \setminus S$. When all remaining sets $S \notin \mathcal{S}$ have $z_S = 0$, stop and output the current V, a singleton set we call V_{final}.

An inductive argument shows that if we start with $V = U$, the final set V_{final} has the same distribution as the set of points in the region containing the origin in the Random Thresholds algorithm. Specifically, the first cut is taken with probability $\frac{z_S}{\sum_{T \notin \mathcal{S}} z_T}$ and the process inductively proceeds; the process is thus identical to the Closest Point Process, and so V_{final} contains a single point $\widehat{\boldsymbol{p}} \in U$ when the process stops with $f(U) = \mathbb{E}[\|\widehat{\boldsymbol{p}}\|_1]$.

To analyze this, we change the perspective slightly further, and recast the process in terms of "exponential clocks". Define independent exponential random variables $X_S \sim \exp(z_S)$ for each set $S \subseteq 2^U$ such that $z_S > 0$. Since exponential random variables $\{Y_i \sim \exp(r_i)\}$ have the memorylessness property, and the property that $\Pr[Y_j = \min_i\{Y_i\}] = \frac{r_j}{\sum_i r_i}$, we see the sets in the same order $\mathcal{S}$ as in the last-center process above. Moreover, this order depends only on the set U, and is independent of the starting set $V \subseteq U$.

Now consider the Last Point Process starting with different sets $V \subseteq U$ (and not just the entire point set U): naturally, the identity of the final point $\widehat{\boldsymbol{p}}$ changes. However, it turns out we can make the following claim. Define the event *point $\boldsymbol{p} \in U$ is last in V* if starting with the set V results in $V_{\text{final}} = \{\boldsymbol{p}\}$. It turns out that being last in this process has a nice "monotone" property. (We defer the proof to Section 6.8.1.)

Lemma 6.2.4 (Monotonicity)**.** *For any sets T, V such that $T \subseteq V$, and any point $\boldsymbol{p} \in V \setminus T$, we have*

$$\text{“}\boldsymbol{p} \text{ is last in } V\text{”} \Rightarrow \text{“}\boldsymbol{p} \text{ is last in } V \setminus T\text{”}.$$

6.2.3 Bounding the Expected Cost

By the definition of our process, we know that

$$f(U) = \sum_{\boldsymbol{p} \in U} \|\boldsymbol{p}\|_1 \cdot \Pr[\boldsymbol{p} \text{ is last in } U] \leq \gamma + \sum_{\boldsymbol{p}:\|\boldsymbol{p}\|_1 > \gamma} \|\boldsymbol{p}\|_1 \cdot \Pr[\boldsymbol{p} \text{ is last in } U] \tag{6.4}$$

for any γ. (We choose $\gamma > 1$, which ensures that $\boldsymbol{p} \neq \boldsymbol{p}^*$.) We now bound (6.4) as follows. Observe that whenever $\boldsymbol{p}$ is last in U, the following is true. There must exist a cut T that removes

the closest point $\boldsymbol{p}^*$ before any cut removes $\boldsymbol{p}$, i.e., $\boldsymbol{p}^* \in T, \boldsymbol{p} \notin T$. This implies $X_T \leq X_S$ for all sets S such that $\boldsymbol{p} \in S, \boldsymbol{p}^* \notin S$, which can be written as

$$X_T \leq X_{\boldsymbol{p}}, \text{ where } X_{\boldsymbol{p}} := \min_{S:\boldsymbol{p}\in S, \boldsymbol{p}^*\notin S} X_S \,.$$

Second, by the Monotonicity Lemma 6.2.4, we have that $\boldsymbol{p}$ is last in U implies that $\boldsymbol{p}$ is last in $U\backslash T$. Defining $\mathcal{F}_{\boldsymbol{p}} := \{T \mid \boldsymbol{p}^* \in T, \boldsymbol{p} \notin T\}$ to be all those cuts that could remove $\boldsymbol{p}^*$ before $\boldsymbol{p}$, therefore yields the upper bound

$$\begin{aligned} f(U) &\leq \gamma + \sum_{\boldsymbol{p}:\|\boldsymbol{p}\|_1>\gamma} \|\boldsymbol{p}\|_1 \cdot \Pr[\exists T \in \mathcal{F}_{\boldsymbol{p}} \text{ such that } X_T \leq X_{\boldsymbol{p}} \bigwedge \boldsymbol{p} \text{ is last in } U\backslash T] \\ &\leq \gamma + \sum_{\boldsymbol{p}:\|\boldsymbol{p}\|_1>\gamma} \|\boldsymbol{p}\|_1 \sum_{T\in\mathcal{F}_{\boldsymbol{p}}} \Pr[X_T \leq X_{\boldsymbol{p}} \bigwedge \boldsymbol{p} \text{ is last in } U\backslash T] \,. \qquad \text{(union bound)} \end{aligned}$$

We upper bound the contribution of a fixed point $\boldsymbol{p}$ to the above expression. By the law of total probability, $\Pr[X_T \leq X_{\boldsymbol{p}} \bigwedge \boldsymbol{p}$ is last in $U\backslash T]$ equals

$$\int_{-\infty}^{\infty} \Pr[X_T \leq t \bigwedge \boldsymbol{p} \text{ is last in } U\backslash T \mid X_p = t] f_{X_p}(t) dt \,, \tag{6.5}$$

where $f_{X_p}(t)$ denotes the probability density function of X_p. The event $X_T \leq t$ is independent from the event "$\boldsymbol{p}$ is last in $U\backslash T$" because, T does not cut any points in $U\backslash T$ and hence the value of X_T is irrelevant to the process restricted to points in $U\backslash T$. We also know that X_T and $X_{\boldsymbol{p}}$ are independent. These observations can be used to rewrite the above expression as

$$\int_{-\infty}^{\infty} \Pr[X_T \leq t] \cdot \Pr[\boldsymbol{p} \text{ is last in } U\backslash T \mid X_p = t] f_{X_p}(t) dt \,. \tag{6.6}$$

As $\Pr[X_T \leq t]$ is an increasing function of t, the above expression is maximized if the probability mass of the event "$\boldsymbol{p}$ is last in $U\backslash T$" is on large values of t. Formally, if we select $\boldsymbol{\tau}$ to be threshold so that

$$\int_{-\infty}^{\infty} \Pr[\boldsymbol{p} \text{ is last in } U\backslash T \mid X_p = t] f_{X_p}(t) dt = \int_{-\infty}^{\infty} \mathbf{1}[t \geq \boldsymbol{\tau}] f_{X_p}(t) dt = \int_{\boldsymbol{\tau}}^{\infty} f_{X_p}(t) dt \tag{6.7}$$

then (6.6) is upper bounded by

$$\int_{\boldsymbol{\tau}}^{\infty} \Pr[X_T \leq t] f_{X_p}(t) dt \,. \tag{6.8}$$

To understand this expression, recall that X_T is an exponential random variable with rate z_T. Further, the random variable X_p is the minimum of exponentials, and hence is itself exponentially distributed with rate $\ell(\boldsymbol{p}) = \sum_{S:\boldsymbol{p}\in S, \boldsymbol{p}^*\notin S} z_S$. In other words, $\Pr[X_T \leq t] = 1 - e^{-z_T \cdot t}$ and $f_{X_p}(t) = \ell(\boldsymbol{p}) e^{-\ell(\boldsymbol{p})t}$ for $t \geq 0$. This gives us that the choice of $\boldsymbol{\tau}$ that satisfies the identity (6.7) is

$$\boldsymbol{\tau} = \frac{-\ln Q_T(\boldsymbol{p})}{\ell(\boldsymbol{p})}, \text{ where } Q_T(\boldsymbol{p}) \text{ is the probability that } \boldsymbol{p} \text{ is last in } U\backslash T.$$

The integral (6.8) can be upper-bounded by standard calculations:

$$\begin{aligned}\int_{\boldsymbol{\tau}}^{\infty} \Pr[X_T \le t] f_{X_p}(t) dt &= \int_{\boldsymbol{\tau}}^{\infty} (1 - e^{-z_T t}) \cdot \ell(\boldsymbol{p}) \cdot e^{-\ell(\boldsymbol{p})t} dt \\ &= Q_T(\boldsymbol{p}) - \frac{\ell(\boldsymbol{p})}{\ell(\boldsymbol{p}) + z_T} e^{-(\ell(\boldsymbol{p})+z_T)\cdot\boldsymbol{\tau}} \\ &\le Q_T(\boldsymbol{p}) \left(1 - \frac{\ell(\boldsymbol{p})}{\ell(\boldsymbol{p}) + z_T} \cdot \left(1 + \frac{z_T \ln Q_T(\boldsymbol{p})}{\ell(\boldsymbol{p})}\right)\right) \\ &= Q_T(\boldsymbol{p}) \cdot z_T \left(\frac{1 - \ln Q_T(\boldsymbol{p})}{\ell(\boldsymbol{p}) + z_T}\right).\end{aligned}$$

Substituting in this upper bound, we have

$$\begin{aligned}f(U) &\le \gamma + \sum_{\boldsymbol{p}:\|\boldsymbol{p}\|_1 > \gamma} \|\boldsymbol{p}\|_1 \sum_{T \in \mathcal{F}_{\boldsymbol{p}}} Q_T(\boldsymbol{p}) \cdot z_T \left(\frac{1 - \ln Q_T(\boldsymbol{p})}{\ell(\boldsymbol{p}) + z_T}\right) \\ &\le \gamma + \sum_{\boldsymbol{p}:\|\boldsymbol{p}\|_1 > \gamma} \frac{\|\boldsymbol{p}\|_1}{\|\boldsymbol{p}\|_1 - 1} \sum_{T \in \mathcal{F}_{\boldsymbol{p}}} Q_T(\boldsymbol{p}) \cdot z_T \left(1 - \ln Q_T(\boldsymbol{p})\right),\end{aligned}$$

where we use that $\ell(\boldsymbol{p}) + z_T \ge \|\boldsymbol{p}\|_1 - 1$. Indeed $\ell(\boldsymbol{p}) = \sum_{S:\boldsymbol{p}\in S, \boldsymbol{p}^* \notin S} z_S \ge \|\boldsymbol{p}\|_1 - \|\boldsymbol{p}^*\|_1 \ge \|\boldsymbol{p}\|_1 - 1$. Using the fact that $x/x-1$ is a decreasing function and then replacing the summation over all $\boldsymbol{p} : \|\boldsymbol{p}\|_1 > \gamma$ to all $\boldsymbol{p} \ne \boldsymbol{p}^*$, and exchanging the summations gives

$$\le \gamma + \frac{\gamma}{\gamma - 1} \sum_{T \ni \boldsymbol{p}^*} z_T \sum_{\boldsymbol{p} \in U \setminus T} Q_T(\boldsymbol{p})(1 - \ln Q_T(\boldsymbol{p})).$$

Observe that for any cut T, we have $\sum_{\boldsymbol{p}\in U\setminus T} Q_T(\boldsymbol{p}) = 1$, and the sum is over at most $k-1$ points. As the entropy $\sum_{\boldsymbol{p}\in U\setminus T} Q_T(\boldsymbol{p})(-\ln Q_T(\boldsymbol{p}))$ of $|U\setminus T| \le k-1$ outcomes is at most $\ln(k-1)$, the inner sum is at most $1 + \ln(k-1)$. Finally, using that $\sum_{T\ni\boldsymbol{p}^*} z_T = \|\boldsymbol{p}^*\|_1 = 1$, we get

$$f(U) \le \gamma + \frac{\gamma}{\gamma - 1}(1 + \ln(k-1)) \le \ln(k-1) + 2\sqrt{1 + \ln(k-1)} + 2$$

by optimizing over γ. This proves Theorem 6.2.3, and gives us an asymptotically optimal bound on the price of explainability. In the next section we show that the bound is, in fact, tight up to lower-order terms.

6.3 Lower Bounds on the Price of Explainability

In this section, we prove a tight lower bound on the price of explainability (up to lower order terms), and a lower bound on the approximability of explainable clustering. Both results are obtained via a reduction from the classic *hitting set problem*: given a set system $([d], \mathcal{S})$, where $[d] = \{1, \ldots, d\}$ denotes the ground set and $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$ is a family of k subsets of $[d]$, the task is to find the smallest subset $H \subseteq [d]$ that hits every subset in $\mathcal{S}$, i.e., $H \cap S_i \ne \emptyset$ for all $S_i \in \mathcal{S}$. We further say that a hitting set instance $([d], \mathcal{S})$ is *s-uniform if all subsets of $\mathcal{S}$ are of the same size s*. We now first present the reduction from s-uniform hitting set instances to explainable clustering, and we then analyze its implications.

Reducing hitting set to explainable clustering. Given an s-uniform hitting set instance $([d], \mathcal{S} = \{S_1, S_2, \ldots, S_k\})$, define the following data set $\mathcal{X}$ in $\{0,1\}^d$ and reference solution $\mathcal{U}$:

1. The reference clustering has $k+1$ centers $\mathcal{U} := \{\boldsymbol{\mu}_0, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k\}$, where $\boldsymbol{\mu}_0$ is at the origin, and each other $\boldsymbol{\mu}_i \in \{0,1\}^d$ is the characteristic vector of the set S_i.
2. The data set $\mathcal{X}$ consists of one point at each of the locations $\{\mathbf{e}_i\}_{i\in[d]}$, and $M = \mathrm{poly}(d,k) \gg \max\{d,k\}$ "colocated" points at each of the $k+1$ locations in $\mathcal{U}$, giving $|\mathcal{X}| = d + M \cdot (k+1)$.

The cost of this reference clustering $\mathcal{U}$ with $k+1$ centers is at most d, since all the d non-colocated points can be assigned to the center $\boldsymbol{\mu}_0$. We proceed to analyze the cost of an optimal *explainable* clustering with $k+1$ centers.

Lemma 6.3.1. *Let h be the size of an optimal solution to the hitting set instance $([d], \mathcal{S})$ and let OPT be the cost of an optimal $(k+1)$-median explainable clustering of the data set $\mathcal{X}$. Then*

$$d + h(s - 2 - o(1)) \le \mathrm{OPT} \le d + h(s-2)\,.$$

Moreover, the same bounds hold for the optimal $(k+1)$-means explainable clustering.

Proof. We present the proof for $(k+1)$-median and then observe that all the distance calculations also hold for $(k+1)$-means since all the points of $\mathcal{X}$ have binary coordinates and, as we will show, the centers will be (arbitrarily close) to such coordinates as well. Note that $\|\mathbf{p}-\mathbf{q}\|_1 = \|\mathbf{p}-\mathbf{q}\|_2^2$ if $\mathbf{p}, \mathbf{q} \in \{0,1\}^d$. We now proceed with the analysis for $(k+1)$-median.

The M points colocated with each of the reference centers $\boldsymbol{\mu}_i$ ensure that the best explainable clustering separates each of the centers $\boldsymbol{\mu}_i$. Separating a center $\boldsymbol{\mu}_i$ from $\boldsymbol{\mu}_0$ using a threshold cut means choosing some dimension $j \in S_i$ and a value $\theta \in (0,1)$, which in turn also separates the data point $\boldsymbol{e}_j$ from $\boldsymbol{\mu}_0$. Since $M \gg k$, the center for the final cluster containing $\boldsymbol{e}_j$ is located at some location very close the reference center in it, and hence this data point now incurs cost $s - (1 + o(1)))$ instead of 1. Here we used the fact that each set has size s and the term $o(1)$ accounts for the potential small difference in the locations of the centers in the explainable clustering compared to those in the reference clustering. The above observations imply that

- the collection of threshold cuts that separate $\boldsymbol{\mu}_0$ from other centers must form a hitting set for the set system $([k], \mathcal{S})$; and
- if this hitting set has size h', the cost of the explainable clustering is at least $h'(s - (1 + o(1)) + (d - h') = d + h'(s - 2 - o(1))$.

We thus have $\mathrm{OPT} \ge d + h(s - 2 - o(1))$ since h is the smallest size of a hitting set.

For the upper bound $\mathrm{OPT} \le d + h(s-2)$, let $H = \{i_1, i_2, \ldots, i_h\} \subseteq [d]$ be an optimal hitting set of size h. Starting with the reference clustering $\mathcal{U}$, build a threshold tree by adding the threshold cuts along dimensions $i_1, i_2, \ldots, i_h$ with thresholds $1/2$. Specifically, the cut along dimension i_1 is at the root of the tree and the remaining cuts are recursively added to the subinstance that contain the reference center $\boldsymbol{\mu}_0$. After adding these cuts we have separated $\boldsymbol{\mu}_0$ from all other centers, since H is a hitting set. Furthermore, the only points in $\mathcal{X}$ that are separated from their

closest center in $\mathcal{U}$ are $\boldsymbol{e}_{i_1}, \boldsymbol{e}_{i_2}, \ldots, \boldsymbol{e}_{i_h}$. Note that the tree may still contain centers $\boldsymbol{\mu}_i, \boldsymbol{\mu}_j$ with $i, j \geq 1$ that are yet not separated. But they can be separated without incurring any additional cost since, in their subinstance, all points of $\mathcal{X}$ are colocated with the centers (or have already been separated from their closest center $\boldsymbol{\mu}_0$). Hence, we can build a threshold tree that has one leaf per center in $\mathcal{U}$ and the only points of $\mathcal{X}$ that are separated from their closest center are $\boldsymbol{e}_{i_1}, \boldsymbol{e}_{i_2}, \ldots, \boldsymbol{e}_{i_h}$. Each of these separated points $\boldsymbol{e}_{i_j}$ has cost at most $s-1$ instead of 1 since the hitting set instance was s-uniform and the final center $\boldsymbol{\mu}_q$ that $\boldsymbol{e}_{i_j}$ is assigned to correspond to a set S_q that contains i_j, and so $\|\boldsymbol{e}_{i_j} - \boldsymbol{\mu}_q\|_1 = s-1$. Hence, the total cost of the clustering is at most $h \cdot (s-1) + d - h = d + h(s-2)$, which completes the proof of the lemma. □

Having described our reduction, we now proceed to its implications.

Price of explainability for k-median. As aforementioned, the cost of the reference clustering $\mathcal{U}$ is at most d. Furthermore, Lemma 6.3.1 says that the optimal $(k+1)$-median explainable clustering costs at least $h(s-2-o(1))+d$, where h is the size of an optimal hitting set of $([d], \mathcal{S})$. It thus suffices to construct a set system $\mathcal{S}$ having large $h(s-2-o(1)) \approx hs$. For example, letting $d = |\mathcal{S}| = k$ and defining $\mathcal{S}$ based on the Hadamard code would give us $s = k/2$ and a hitting set of size $\log_2 k$, and hence a lower bound of $\approx \frac{1}{2}\log_2 k$. A better guarantee follows using a probabilistic construction (selecting uniformly at random sets), whose proof we defer to the appendix.

Lemma 6.3.2 (Hitting Set Lemma)**.** *For large enough k, there exist set systems $([k], \mathcal{S})$ with k sets of size s each, such that the minimum hitting set satisfies $h(s-2-o(1))/k \geq \ln k - O(\ln \ln k)$.*

Combining the above lemma with our reduction shows that the price of explainability is at least $(1-o(1))\ln k$, giving the proof of Theorem 1.5.2.

Theorem 1.5.2 (Lower Bound for k-medians)**.** *There exist instances of k-medians for which any explainable clustering has cost at least $(1-o(1))\ln k$ times the cost of the optimal k-medians clustering.*

Hardness of approximation Our reduction from the hitting set problem to explainable clustering immediately leads to a hardness result as well. Feige, in his landmark paper [74], proved that it is hard to distinguish whether an s-uniform hitting set instance $([d], \mathcal{S})$ [2]

- *(yes case:)* has a hitting set of d/s elements; or
- *(no case:)* any hitting set has size at least $(1-o(1))\ln(k) \cdot d/s$, where $k = |\mathcal{S}|$.

[2]We remark that the result in [74] is stated in the terminology of set cover. The instances constructed there has a ground set of size n and a family of m subsets. Furthermore, they can be assumed to be regular: each element is contained in s subsets and each subset is of size ℓ. Now in the yes case, there is a set cover so that each element is covered by exactly one set. By the regularity, this implies that the set cover has size $n/\ell = m/s$ in the yes case. Here we used that $n \cdot s = m \cdot \ell$. In the no case however, any set cover is at least a factor $(1-o(1))\ln n$ larger. Now in the terminology of hitting set, this is an hitting set instance with $d = m$ elements and a family $\mathcal{S}$ of $k = n$ many sets, each of size s, with the stated yes case and no case.

Here, "hard" means that there is no polynomial-time algorithm can distinguish between these two cases unless $\mathsf{P} = \mathsf{NP}$; it was under the stronger assumption in Feige's original paper [74] and then subsequently improved to hold under the weakest possible assumption $P \neq NP$ [69, 142].

Our reduction runs in polynomial time so the above hardness together with Lemma 6.3.1 implies the following. Assuming $P \neq NP$, there is no polynomial-time algorithm that, given a data set $\mathcal{X} \subseteq \mathbb{R}^d$, distinguishes whether

- *(yes case:)* there is an explainable clustering with $k+1$ clusters of cost at most $2d$; or
- *(no case:)* any such clustering has cost at least $(1 - o(1)) \ln(k) d$.

As any approximation algorithm with better guarantee than $(1/2 - o(1)) \ln(k)$ would allow us to distinguish between the two cases, we have the following hardness of approximation result for explainable clustering.

Theorem 1.5.4 (Approximability)**.** *The explainable k-medians and k-means problems are hard to approximate better than $(1/2 - o(1)) \ln k$, unless $\mathsf{P} = \mathsf{NP}$.*

The above hardness result settles the approximability of explainable k-medians up to small constants: it is the same as its price of explainability! For k-means, the situation is different. Our hardness of approximation result is far from the lower bound $\Omega(k)$ on its price of explainability. We conjecture that there is no such hardness result matching $\Omega(k)$ and, in contrast to k-medians, that there are significantly better approximation algorithms for explainable k-means than its price of explainability.

6.4 Explainable k-means clustering

We now prove our improved bound on the price of explainability of the k-means problem, which improves on the previous bound of $O(k \ln k)$. Our main result is the following:

Theorem 6.4.1. *Given a data set $\mathcal{X}$ and a base clustering with centers $\mathcal{U}$ and map π, we can output a random threshold tree T separating $\mathcal{U}$ such that*

$$\mathbb{E}[\mathrm{cost}_2(T)] \leq O(k \ln \ln k) \cdot \mathrm{cost}_2(\pi, \mathcal{U}).$$

At a high level, the approach is similar to that for k-medians: we give an algorithm to separate a given set of centers, but since we are dealing with squared Euclidean distances, we choose cuts from a non-uniform distribution over dimensions and coordinate values. However, since a single cut can increase the cost by a factor of $\Omega(k)$ we have to be careful not to lose another factor of $\Omega(\ln k)$ due to the recursion. Here we use a win-win analysis: we define a quantity called the *stretch* of a pair of points and argue that the loss due to a single cut is just the stretch: moreover, we show that if stretch is large, the recursive problems are relatively balanced and the loss in the recursion is a geometric sum, adding up to $\approx O(k)$. On the other hand, if the stretch is low, we lose less-than-the-worst-case in each round (although we now need to take a collection of "bulk" cuts).

6.4.1 The Closest Point Process Again

Recall that we proved the performance of the Random Thresholds algorithm for k-medians by reducing to the perspective of a single data point and analyzing the expected increase in its cost. We can also define the closest point process for any ℓ_q norm and for any algorithm $\mathcal{A}$ separating point sets U that is invariant under translations and scaling, as follows:

Definition 6.4.2 (ℓ_q-Norm Closest Point Process)**.** Given a set of k points $U \subseteq \mathbb{R}^d$, let $\boldsymbol{p}^* := \arg\min_{\boldsymbol{p} \in U} \|\boldsymbol{p}\|_q$ be the point in U closest to the origin according to the ℓ_q metric. Assume $\|\boldsymbol{p}^*\|_q = 1$. Run the algorithm $\mathcal{A}$ to create a random threshold tree T that separates this point set U. Consider leaf node $u \in T$ whose corresponding region $B_u \subseteq \mathbb{R}^d$ contains the origin, and let $\widehat{\boldsymbol{p}}$ be the unique point of U in this region B_u. Define

$$f_{q,\mathcal{A}}(U) := \mathbb{E}[\ \|\widehat{\boldsymbol{p}}\|_q^q\ ]. \tag{6.9}$$

Finally, define $\alpha_{q,\mathcal{A}}(k) := \max_{U:|U|=k} f_{q,\mathcal{A}}(U)$.

A proof identical to that of Lemma 6.2.2 shows that the price of explainability for ℓ_q-norm clustering is at most $\alpha_{q,\mathcal{A}}(k)$. In the rest of this section, we give some terminology and then an algorithm $\mathcal{A}$ which separates the input point set $U \subseteq \mathbb{R}^d$; we then bound the resulting value of $f_{q,\mathcal{A}}(U)$.

6.4.2 Terminology

We use similar terminology as in [83]. Given a set $U \subseteq \mathbb{R}^d$ of points, and a dimension $i \in [d]$, let $\ell_i := \min_{v \in U} v_i$ and $u_i := \max_{v \in U} v_i$ be the leftmost and rightmost coordinates of points. Given two values $x, y \in \mathbb{R}$, let $\mathcal{I}_i(x, y)$ be the set of consecutive intervals along the i-th dimension delimited by the coordinates x and y themselves and the projections of points in U that lie between x and y. For example, consider the 2-dimensional point set U shown in Figure 6.2 (the same example was given in [83]). On the horizontal axis, two coordinate values x and y are marked along with the projections of the points: $\mathcal{I}_1(x, y)$ consists of the three consecutive intervals $[x, v_1^4]$, $[v_1^4, v_1^2]$, and $[v_1^2, y]$.

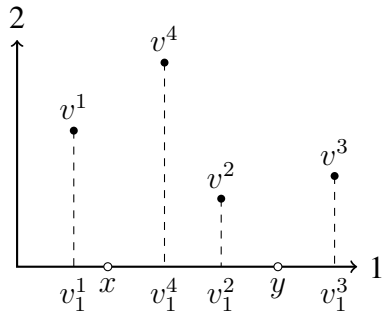


Figure 6.2: Intervals defined by projections.

By the definition of $\mathcal{I}_i(x, y)$, we have $|x - y| = \sum_{[a,b] \in \mathcal{I}_i(x,y)} |b - a|$. Let

$$\mathcal{I}_{\text{all}} := \cup_{i \in [d]} \{(i, [a, b]) \mid [a, b] \in \mathcal{I}_i(\ell_i, u_i)\}$$

denote the collection of all dimension-interval pairs which are delimited by the projections of the

points onto the respective dimensions; for brevity, define $\mathcal{I}_i := \mathcal{I}_i(\ell_i, u_i)$. Define

$$L_2 := \sum_{(i,[a,b])\in\mathcal{I}_{\text{all}}} |b-a|^2.$$

A key definition is that of the *pseudo-distance*: for points $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$, let

$$\mathcal{I}(\boldsymbol{x}, \boldsymbol{y}) = \bigcup_{i\in[d]} \{(i, [a,b]) \mid [a,b] \in \mathcal{I}_i(x_i, y_i)\}.$$

We then define the pseudo-distance between $\boldsymbol{x}$ and $\boldsymbol{y}$ as

$$d_2(\boldsymbol{x}, \boldsymbol{y}) = \sum_{(i,[a,b])\in\mathcal{I}(x,y)} |b-a|^2.$$

It follows that $\|\boldsymbol{x}-\boldsymbol{y}\|_2^2 \geq d_2(\boldsymbol{x}, \boldsymbol{y}) \geq \frac{1}{|U\cup\{\boldsymbol{x},\boldsymbol{y}\}|-1} \cdot \|\boldsymbol{x}-\boldsymbol{y}\|_2^2$.

We define a distribution D_2 as follows: first select a dimension–interval pair $(i, [a,b]) \in \mathcal{I}_{\text{all}}$ with probability $|b-a|^2/L_2$, and then pick $\theta \in [a,b]$ randomly such that the p.d.f. is

$$P_{a,b}(\theta) := \frac{4}{(b-a)^2}\min(\theta-a, b-\theta) \quad = \quad \frac{4}{(b-a)^2}\min_{\boldsymbol{v}\in U}\{|\theta - v_i|\}. \tag{6.10}$$

Often we refer to the above concepts not for the entire point set U but for some subset $V \subseteq U$; in those cases we refer to the partition $\mathcal{I}_{\text{all}}(V)$, the sum $L_2(V)$, or the distribution $D_2(V)$, etc. Finally, for subset $V \subseteq U$ of points we define:

(i) Let $\Delta(V) := \max_{\boldsymbol{x},\boldsymbol{y}\in V} \|\boldsymbol{x}-\boldsymbol{y}\|^2$ be the squared diameter of point set V.

(ii) Call a pair of points $\boldsymbol{x}, \boldsymbol{y} \in V$ *far* if $\|\boldsymbol{x}-\boldsymbol{y}\|_2^2 \geq \Delta(V)/2$, and *close* if $\|\boldsymbol{x}-\boldsymbol{y}\|_2^2 < \Delta(V)/k^4$.

(iii) Define the *stretch* of a pair $\boldsymbol{x}, \boldsymbol{y} \in V$ to be $s_{\boldsymbol{xy}}(V) := \|\boldsymbol{x}-\boldsymbol{y}\|_2^2/d_2(\boldsymbol{x}, \boldsymbol{y})$. Define the *stretch of the set* $s(V)$ to be the maximum stretch of any *far pair* in V.

6.4.3 The Algorithm

The process to construct the threshold tree T for k-means is slightly more complex than for k-medians: as before we start off with the root representing the entire point set $\mathbb{R}^d$. Now, given a node v representing some box B_v (giving us a subset $U_v := U \cap B_v$), define the distribution $D_2(v) := D_2(U_v)$. Now consider the stretch $s(v) := s(U_v)$ of the set of points.

1. *(Solo Cuts)* If $s(v) \geq \frac{|U_v|}{\ln^2|U_v|}$, let $\boldsymbol{p}^*, \boldsymbol{q}^*$ be a pair of far points in U_v of stretch $s(v)$, and pick a threshold cut $(i, \theta) \sim D_2(v)$, conditioned on separating the pair $\boldsymbol{p}^*, \boldsymbol{q}^*$. This partitions the box B_v into two boxes, and recurse on both.

2. *(Bulk Cuts)* Else if $s(v)$ is smaller than the quantity above. In this case, repeatedly sample cuts from $D_2(v)$ conditioned on not separating any close pairs of points in U_v, until all pairs of far points in U_v are separated. Apply all these cuts in sequence, partitioning B_v into potentially multiple pieces; recurse on each of them.

The process stops when each leaf of T contains a single point from U. For the analysis below, we consider a *compressed threshold tree* T' which is a tree with branching at least two: if we perform a solo cut at node v, we call it a *solo node*, and it has two children corresponding to the two parts obtained by this cut. If we perform bulk cuts at node v, it has potentially multiple children (one for each smaller box obtained by applying these cuts), and we call it a *bulk node*.

In the following, let $S(T')$ and $B(T')$ denote the solo and bulk nodes in T'. For any node $v \in T'$, let $L_2(v) := L_2(U_v)$, and similarly for other parameters defined above.

6.4.4 The Expected Cost Increase

We will bound the cost for the solo cuts and bulk cuts separately. First we give some preliminary lemmas, then we bound the cost due to bulk cuts, and finally the cost for solo cuts.

Lemma 6.4.3 (Separation Probability)**.** *For any subset of points $V \subseteq U$ and a point $\boldsymbol{p} \in V$,*

$$\Pr_{(i,\theta)\sim D_2(V)}[(i,\theta) \text{ separates the origin from } \boldsymbol{p}] \leq \frac{2\|\boldsymbol{p}\|_2^2}{L_2(V)}\,.$$

Proof. The probability that a random cut sampled from $D_2(V)$ separates the origin $\mathbf{0}$ and $\boldsymbol{p}$ is:

$$\begin{aligned}
&\sum_{i\in[d]}\sum_{[a,b]\in\mathcal{I}_i}\frac{(a-b)^2}{L_2(V)}\int_a^b P_{a,b}(\theta)\cdot\mathbb{1}[\theta \text{ is between } 0 \text{ and } p_i]\,d\theta \\
&= \frac{4}{L_2(V)}\sum_{i\in[d]}\sum_{[a,b]\in\mathcal{I}_i}\int_a^b \min(\theta-a, b-\theta)\cdot\mathbb{1}[\theta \text{ is between } 0 \text{ and } p_i]\,d\theta \qquad (6.11)\\
&\leq \frac{4}{L_2(V)}\sum_{i\in[d]}\sum_{[a,b]\in\mathcal{I}_i}\int_a^b |\theta-p_i|\cdot\mathbb{1}[\theta \text{ is between } 0 \text{ and } p_i]d\theta \qquad (6.12)\\
&= \frac{4}{L_2(V)}\sum_{i\in[d]}\int_{-\infty}^{\infty}|\theta-p_i|\cdot\mathbb{1}[\theta \text{ is between } 0 \text{ and } p_i]\,d\theta = \frac{4}{L_2(V)}\sum_{i\in[d]}(p_i)^2/2 = \frac{2\|\boldsymbol{p}\|_2^2}{L_2(V)}.
\end{aligned}$$

Equality (6.11) and inequality (6.12) use the expressions for $P_{a,b}(\theta)$ given in (6.10). □

Given a set V, suppose we sample cuts from distribution $D_2(V)$ with an added rejection step if the cut separates some pair of points in V whose distance is at most $\Delta(V)/k^4$. Formally, let $R(V) \subseteq \mathcal{I}_{\text{all}}(V)$ be the subset of intervals which are contained in projections of close centers in V onto the coordinate axis. Let

$$L_2'(V) := \sum_{(i,[a,b])\in\mathcal{I}_{\text{all}}(V)\setminus R(V)} |b-a|^2.$$

The distribution $D_2'(V)$ picks an interval $[a,b]$ in $\mathcal{I}_{\text{all}}'(V) := \mathcal{I}_{\text{all}}(V)\setminus R(V)$ with probability $\frac{(b-a)^2}{L_2'(V)}$ and then a cut is chosen from this interval with the same distribution as $P_{a,b}(\theta)$.

Proposition 6.4.4. *For any subset of points V, we have $L_2'(V) \geq L_2(V)/2$.*

Proof. The sum of squared length of intervals in $R(V)$ is at most the total sum of squared distance between all pairs of close centers, which is at most

$$\binom{|V|}{2} \cdot \frac{\Delta(V)}{k^4} \leq \frac{\Delta(V)}{2k^2} \leq \frac{L_2(V)}{2k}.$$

This implies that $L_2'(V) \geq L_2(V)(1 - 1/2k) \geq \frac{1}{2} \cdot L_2(V)$. □

Lemma 6.4.5 (Expected Number of Cuts)**.** *For any node V, the expected number of cuts from $D_2'(V)$ until all far pairs in V are separated is at most $24 \ln |V| \cdot s(V) \cdot \frac{L_2(V)}{\Delta(V)}$.*

Proof. Consider a collection of $M := 3 \ln |V| \cdot \frac{4s(V) \cdot L_2(V)}{\Delta(V)}$ cuts sampled from $D_2'(V)$, and consider two "far" points $\boldsymbol{p}, \boldsymbol{q}$, i.e., such that $\|\boldsymbol{p} - \boldsymbol{q}\|_2^2 > \Delta(V)/2$. Then, the probability that any one cut separates the two is at least

$$\begin{aligned}
\Pr_{(i,\theta) \sim D_2'(V)}[(i,\theta) \text{ separates } \boldsymbol{p}, \boldsymbol{q}] \geq \frac{d_2(\boldsymbol{p}, \boldsymbol{q}) - \binom{k}{2}\frac{\Delta(V)}{k^4}}{L_2(V)} &\geq \frac{\|\boldsymbol{p} - \boldsymbol{q}\|_2^2}{s(V) \cdot L_2(V)} - \frac{\Delta(V)}{2k^2 \cdot L_2(V)} \\
&\geq \frac{\Delta(V)}{2s(V) \cdot L_2(V)} - \frac{\Delta(V)}{2k^2 \cdot L_2(V)} \\
&\geq \frac{\Delta(V)}{4s(V) \cdot L_2(V)}.
\end{aligned}$$

The last inequality in the above equation follows using $s(V) \leq k$ and $k \geq 2$. Hence the probability that the M cuts do not separate some pair at distance at least $\Delta(V)/2$ can be upper-bound using a union bound by

$$\binom{|V|}{2} \cdot \left(1 - \frac{\Delta(V)}{4 \cdot s(V) \cdot L_2(V)}\right)^M \leq |V|^2 \left(\frac{1}{e}\right)^{(3 \ln |V|)} = 1/|V|.$$

Hence, these M cuts separate all pairs that have squared distance at least $\Delta(V)/2$ with probability at least $1 - 1/|V| \geq 1/2$. In turn, the expected number of cuts is at most $2M$. □

We can now start to bound the cost incurred due to bulk cuts.

Lemma 6.4.6 (Logarithmic Number of Relevant Levels)**.** *For any cut (i, θ), we have*

$$\sum_{v \in B(T')} \mathbb{1}_{[\mathbf{0} \in B_v]} \cdot \mathbb{1}_{[(i,\theta) \in \mathrm{supp}(D_2'(v))]} \leq 4 \ln k.$$

Proof. The bulk nodes in the compressed tree T' that correspond to the part containing the origin $\mathbf{0}$ lie on a root-leaf path; call these $v_1, v_2, \ldots, v_\ell$, with v_1 closest to the root. Our algorithm ensures that $\Delta(v_j) \leq \Delta(v_{j-1})/2$. Consider the lowest integer j such that (i, θ) belongs to the support of $D_2'(v_j)$, and let $\boldsymbol{p}, \boldsymbol{q} \in U \cap B_{v_j}$ be the closest pair of points in B_{v_j} separated by (i, θ). The definition of the probability distribution $D_2'(v_j)$ ensures that $\|\boldsymbol{p} - \boldsymbol{q}\|_2^2 \geq \Delta(v_j)/k^4$. For $j' = j + 4 \ln k + 1$ we have that $\Delta(v_{j'}) < \Delta(v_j)/k^4$, and so there are no pairs of points separated by (i, θ)—implying that this cut will no longer be in the support of $D_2(v_{j''})$ for $j'' \geq j'$. □

Lemma 6.4.7 (Cost for Bulk Cuts)**.** *The expected cost increase due to bulk cuts is at most* $O(k) \cdot \|p^*\|_2^2$.

Proof. Consider a bulk node v in the decision tree created by the algorithm: we generate a random number of bulk cuts $\mathcal{K}_v$ at this node, and each of these can cause an increase in cost. Let Y_t be the following upper bound on the increase in cost due to the t^{th} such cut (i, θ):

$$Y_t := \Delta(v) \cdot \mathbb{1}_{[(i,\theta) \text{ seps } \mathbf{0}, \boldsymbol{p}^*]} \cdot \mathbb{1}_{[\{\mathbf{0}, \boldsymbol{p}^*\} \subseteq B_v]}.$$

Moreover, let N be the number of such cuts, then the total expected cost is $\mathbb{E}[\sum_{t=1}^{N} Y_t]$. Since the Y_t variables are independent and N is a stopping time, we can use Wald's equation to infer that the total expected cost due to these cuts is $\mathbb{E}[N] \cdot \mathbb{E}[Y_t]$. Taking expectations of the expression for Y_t above (with respect to the distribution $D_2'(v)$), and using Lemma 6.4.5 to bound $\mathbb{E}[N]$, this is at most

$$\left(O(s(v) \ln |U_v|) \cdot \frac{L_2(v)}{\Delta(v)}\right) \cdot \Delta(v) \cdot \sum_{(i,[a,b]) \in \mathcal{I}'_{\text{all}}(v)} \frac{(b-a)^2}{L_2'(v)} \int_a^b P_{a,b}(\theta) \cdot \mathbb{1}_{[(i,\theta) \text{ seps } \mathbf{0}, \boldsymbol{p}^*]} \cdot \mathbb{1}_{[\{\mathbf{0}, \boldsymbol{p}^*\} \subseteq B_v]} \, d\theta.$$

Using Proposition 6.4.4, we know that $L_2'(v) \geq L_2(v)/2$, so the above expression is at most

$$O(s(v) \ln |U_v|) \cdot \sum_{(i,[a,b]) \in \mathcal{I}_{\text{all}}(v)} (b-a)^2 \int_a^b P_{a,b}(\theta) \cdot \mathbb{1}_{[(i,\theta) \text{ seps } \mathbf{0}, \boldsymbol{p}^*]} \cdot \mathbb{1}_{[\{\mathbf{0}, \boldsymbol{p}^*\} \subseteq B_v]} \cdot \mathbb{1}_{[(i,\theta) \in \text{supp}(D_2'(v))]} \, d\theta.$$

Next, we observe that for any dimension i, we have

$$(b-a)^2 \cdot P_{a,b}(\theta) \cdot \mathbb{1}_{[\boldsymbol{p}^* \in B_v]} \leq 4|\theta - \boldsymbol{p}_i^*|.$$

Moreover, for each bulk node we have $s(v) \ln |U_v| \leq |U_v| / \ln |U_v|$. This in turn is at most $k / \ln k$, since the function $\frac{x}{\ln x}$ is monotone and $|U_v| \leq k$. Substituting both these facts, we get

$$O(k/\ln k) \cdot \sum_i \int_{-\infty}^{\infty} |\theta - \boldsymbol{p}_i^*| \cdot \mathbb{1}_{[(i,\theta) \text{ seps } \mathbf{0}, \boldsymbol{p}^*]} \cdot \mathbb{1}_{[\mathbf{0} \in B_v]} \cdot \mathbb{1}_{[(i,\theta) \in \text{supp}(D_2'(v))]} \, d\theta.$$

Next, we use Lemma 6.4.6 to get:

$$\sum_{v \in B(T')} \mathbb{1}_{[\mathbf{0} \in B_v]} \cdot \mathbb{1}_{[(i,\theta) \in \text{supp}(D_2'(v))]} \leq O(\ln k).$$

Now summing over all v, we get

$$O(k) \cdot \sum_i \int_{-\infty}^{\infty} |\theta - \boldsymbol{p}_i^*| \cdot \mathbb{1}_{[(i,\theta) \text{ seps } \mathbf{0}, \boldsymbol{p}^*]} \, d\theta = O(k) \cdot \|\boldsymbol{p}^*\|_2^2.$$

This completes the proof. □

Finally, we turn our attention to solo cuts. For solo node v let $\boldsymbol{p}_v, \boldsymbol{q}_v$ be the two far nodes such that their stretch is at least $\frac{|U_v|}{(\ln |U_v|)^2}$, and define the distribution $D_2''(v)$ to be the distribution $D_2(v)$ conditioned on separating this far pair.

Lemma 6.4.8 (Ratio for Solo Cuts)**.** *For any solo node $v \in T'$,*

$$\frac{\mathbb{E}_{(i,\theta)\sim D_2''(v)}[\textit{ cost increase at node } v\ ]}{\mathbb{E}_{(i,\theta)\sim D_2''(v)}[\textit{ size of smaller child of node } v\ ]} \leq 32\|\boldsymbol{p}^*\|_2^2 \cdot \left(1 + \ln\left(\frac{|U_v|}{s(v)}\right)\right).$$

Proof. Lemma 6.4.3 implies that

$$\mathbb{E}_{(i,\theta)\sim D_2(v)}[\text{ cost increase }] \leq \Delta(v) \cdot \frac{2\|\boldsymbol{p}^*\|_2^2}{L_2(v)}.$$

Moreover, the probability of separating $\boldsymbol{p}_v, \boldsymbol{q}_v$ is at least

$$\frac{d_2(\boldsymbol{p}_v, \boldsymbol{q}_v \mid U_v)}{L_2(v)} \geq \frac{\|\boldsymbol{p}-\boldsymbol{q}\|_2^2}{s(v)\cdot L_2(v)} \geq \frac{\Delta(v)}{2\cdot s(v)\cdot L_2(v)}.$$

Since the cost increase is non-negative, we get that

$$\mathbb{E}_{(i,\theta)\sim D_2''(v)}[\text{ cost increase }] \leq \Delta(v) \cdot \frac{2\|p^*\|_2^2}{L_2(v)} \cdot \frac{2\cdot s(v)\cdot L_2(v)}{\Delta(v)} \leq 4s(v)\cdot\|\boldsymbol{p}^*\|_2^2. \tag{6.13}$$

This bounds the numerator of the desired quantity; for the denominator, we prove the following claim in Section 6.8.3:

Claim 6.4.9. If for a cut (i,θ), we define $H^+ := \{x \mid x_i \geq \theta\}$ and $H^- = \mathbb{R}^d \setminus H^+$, then

$$\mathbb{E}_{(i,\theta)\sim D_2''(v)}\left[\min\left(|U_v \cap H^+|, |U_v \cap H^-|\right)\right] \geq \frac{s(v)}{8(1+\ln(|U_v|/s(v)))}.$$

Using Claim 6.4.9 with (6.13) finishes the proof. □

Lemma 6.4.10 (Cost for Solo Cuts)**.** *For any internal solo node $v \in T'$, the expected cost increase due to solo cuts made in the subtree T'_v is at most*

$$32|U_v|(1+2\ln\ln|U_v|)\cdot\mathbb{1}_{[\mathbf{0}\in B_v]}\cdot\|\boldsymbol{p}^*\|_2^2.$$

Proof. The proof is by induction. The base cases are when node v has $|U_v| \leq 3$. For v to be an internal node, $|U_v| \in \{2,3\}$. If it has two nodes, then $s(v) = 1 < \frac{2}{\ln^2 2}$, and hence u cannot be a solo node. If we have 3 points, then the only solo node in the subtree T'_v is u itself, so it suffices to argue that the expected cost increase due to this solo cut is at most $32|U_v|(1+2\ln\ln|U_v|)\cdot\|\boldsymbol{p}^*\|_2^2$. From (6.13) we know that the expected cost increase is at most $4s(u)\cdot\|\boldsymbol{p}^*\|_2^2$. Now using $s(u) \leq |U_v|$ and $|U_v| \leq 3$, we obtain the required bound.

Else consider some node v with $|U_v| \geq 4$, and let $\chi(v)$ be the set of solo nodes whose closest solo ancestor is v: it follows that $\sum_{w\in\chi(v)}|U_w| \leq |U_v|$. Moreover, suppose the random solo cut $(i_v,\theta_v) \sim D_2''(v)$ partitions U_v into parts of size at least $\sigma(v)$, then each $|U_v| - |U_w| \geq \sigma(v)$. Using the induction hypothesis on each $w \in \chi(v)$, the total expected cost increase is at most

$$\mathbb{E}[\text{ cost increase at } v\ ] + \sum_{w\in\chi(v)} 32|U_w|(1+2\ln\ln|U_w|)\cdot\mathbb{1}_{[\mathbf{0}\in B_w]}\cdot\|\boldsymbol{p}^*\|_2^2. \tag{6.14}$$

Since the origin belongs to at most one of the sets B_{w^*}, the sum contributes at most

$$32|U_{w^*}|(1 + 2\ln\ln|U_v|) \cdot \|\boldsymbol{p}^*\|_2^2 \leq 32(|U_v| - \mathbb{E}[\sigma(v)])(1 + 2\ln\ln|U_v|) \cdot \|\boldsymbol{p}^*\|_2^2. \tag{6.15}$$

(The RHS is non-negative, since $\sigma(v) \leq |U_v|$, so the bound holds even when $\chi(v) = \emptyset$.) Now Lemma 6.4.8 implies that the cost at v is at most

$$\mathbb{E}[\sigma(v)] \cdot 32(1 + \ln(|U_v|/s(v))) \cdot \|\boldsymbol{p}^*\|_2^2.$$

Finally, using that $s(v) \geq |U_v|/(\ln|U_v|)^2$ for a solo node, and summing the two terms, completes the proof. □

We can wrap up: Using Lemma 6.4.7 and Lemma 6.4.10, we know that the expected cost increase due to all the cuts used to separate the points in U is at most $O(k\ln\ln k) \cdot \|\boldsymbol{p}^*\|_2^2$. If $\widehat{\boldsymbol{p}}$ is the unique point in U in the region B_u corresponding to the leaf node u such that $\mathbf{0} \in B_u$, then the cost is upper bounded by

$$\|\widehat{\boldsymbol{p}}\|_2^2 \leq 2 \cdot \|\widehat{\boldsymbol{p}} - \boldsymbol{p}^*\|_2^2 + 2 \cdot \|\boldsymbol{p}^*\|_2^2 \tag{6.16}$$

using the generalized triangle inequality. Taking expectation on both sides of Equation (6.16) and plugging in $\|\boldsymbol{p}^*\|_2^2 = 1$ gives $f_2(U) \leq O(k\ln\ln k)$.

6.5 Tight Bounds for the Random Threshold Algorithm

We now improve the bound of $(1 + o(1))\ln k$ from §6.2.1 to give an exact bound of $1 + H_{k-1}$ for the Random Thresholds algorithm; we first show an example which achieves this bound, and then give our precise analysis of the algorithm.

6.5.1 A Lower Bound

Consider an instance in $\mathbb{R}^k$ given by a reference clustering having one "close" center $\boldsymbol{\mu}^1 = e_1$, and $k-1$ "far" centers $\boldsymbol{\mu}^i = M\,e_i$ for each $i \in \{2, \ldots, k\}$, where scalar $M \gg 1$. We consider a single data point at the origin. (As always, we can imagine there being many points colocated with each of the centers.) Let the expected assignment cost due to the algorithm for the point at the origin on an instance with j far points be denoted by $g(j)$. Then we get a recurrence:

$$g(k) = \tfrac{1}{M(k-1)+1} \cdot M + \tfrac{M(k-1)}{M(k-1)+1} \cdot g(k-1),$$

and $g(\mathbf{0}) = 1$. As $M \to \infty$, this gives us $g(k) \to 1 + H_{k-1}$, as claimed. In the next section, we will prove a matching upper bound of $1 + H_{k-1}$.

6.5.2 Towards a Matching Upper Bound

Our proof for this case is technical, so the reader may want to keep three special cases in mind:

1. The "axis-aligned" case, inspired by the bad example: the points in U are $\{\boldsymbol{p}_i := d_i \boldsymbol{e}_i\}$, where $1 = d_1 \leq d_2 \leq \ldots \leq d_k$,
2. the "orthogonal closest-point" case, where the closest point is $\boldsymbol{e}_1$, and all other points lie in the orthogonal subspace to it, and
3. the "uniform" case, where all points other than the closest are at the same distance $d \geq 1$.

Many of our proofs become simpler in these special cases, and thinking about these cases will give us crucial intuition.

We start with a set of points $U \subseteq \mathbb{R}^d$, recall that $f(U)$ was defined to be $\mathbb{E}[\|\widehat{\mathbf{p}}\|_1]$, where $\widehat{\mathbf{p}}$ is the unique point in the box containing the origin in the Random Thresholds algorithm. As in §6.2.3, assume we have a dimension for each cut $S \subseteq U$, and point $\boldsymbol{p} \in U$ has value $z_S \mathbb{1}[\boldsymbol{p} \in S] \geq 0$ in this coordinate. Finally defining $\mathcal{C}_S$ to be the collection of sets that cross $S \subseteq U$, we get that for any $S \subseteq U$,

$$f(S) = \frac{\sum_{E \in \mathcal{C}_S} z_E \cdot f(S \setminus E)}{\sum_{E \in \mathcal{C}_S} z_E}, \qquad \text{and} \qquad f(\{\boldsymbol{p}\}) = \|\boldsymbol{p}\|_1. \tag{6.17}$$

Moreover, when $\sum_{E \in \mathcal{C}_S} z_E = 0$, the value of $f(S) = 0$. Let us denote by $\ell(\boldsymbol{p}) := \|\boldsymbol{p}\|_1$ and $\beta_{k-1} := 1 + H_{k-1}$. Using Equation (6.17), we think of the function $f(U)$ purely as an algebraic function of the z_S values, and our central goal in this section to prove the following:

Theorem 6.5.1 (Main Goal)**.** *For any point $\boldsymbol{p} \in U$, the value $f(U) \leq \beta_{k-1} \cdot \ell(\boldsymbol{p})$.*

Since the ratio $f(U)/\ell(\boldsymbol{p})$ is difficult to argue about, we instead focus on bounding the derivative $\frac{\partial f(U)}{\partial z_E}$ by β_{k-1}. The following lemma shows that, by integrating along a path from the origin to the point $\boldsymbol{p}$, such a bound on the derivative suffices: (a formal proof appears in §6.8.4)

Lemma 6.5.2. *For any $\boldsymbol{p} \in U$, if $\frac{\partial f(U)}{\partial z_E} \leq \beta_{k-1}$ for all $E \subseteq U$ with $\boldsymbol{p} \in E$, then $f(U) \leq \beta_{k-1} \cdot \ell(\boldsymbol{p})$.*

6.5.3 Bounding the Derivative

We start by taking definition (6.17) and calculating the derivative $\frac{\partial f(S)}{\partial z_T}$ for the case $|S| \geq 2$:

$$\frac{\partial f(S)}{\partial z_T} = \frac{\sum_{E \in \mathcal{C}_S} z_E \cdot \frac{\partial f(S \setminus E)}{\partial z_T} + \mathbb{1}[T \in \mathcal{C}_S] \cdot \big(f(S \setminus T) - f(S)\big)}{\sum_{E \in \mathcal{C}_S} z_E}. \tag{6.18}$$

When $|S| = 1$, if $S = \{r\}$, then $f(S) = \ell(r)$ by definition, and so $\frac{\partial f(S)}{\partial z_T} = \frac{\partial \ell(r)}{\partial z_T}$. Henceforth, let us fix a set $S \subseteq U$ and some subset T. The next lemma (in §6.8.4) follows by direct calculations:

Lemma 6.5.3. *The partial derivatives satisfy:*

(i) If $T \supseteq S$, then $\frac{\partial f(S)}{\partial z_T} = 1$.
(ii) If $T \cap S = \emptyset$, then $\frac{\partial f(S)}{\partial z_T} = 0$.

(iii) We have $f(S) \geq \sum_{E \supseteq S} z_E$.

The partial derivative $\frac{\partial f(S)}{\partial z_T}$ is not very well-behaved: e.g., it is not guaranteed to be non-negative (which turns out to make an inductive proof difficult). To address this issue, we define a surrogate *pseudo-derivative* operator, and use bounds on this pseudo-derivative to bound the derivative.

Definition 6.5.4 (Pseudo-Derivative)**.** The pseudo-derivative of $f(S)$ with respect to the variable z_T such that T crosses S (i.e., $T \in \mathcal{C}_S$) is:

$$\frac{\widehat{\partial} f(S)}{\widehat{\partial} z_T} = \frac{\sum_{E \in \mathcal{C}_S} z_E \cdot \frac{\widehat{\partial} f(S \setminus E)}{\widehat{\partial} z_T} + f(S \setminus T) - \sum_{E \supseteq S} z_E}{\sum_{E \in \mathcal{C}_S} z_E}. \tag{6.19}$$

It is defined to be 1 if $T \supseteq S$, and 0 if $T \cap S = \emptyset$.

Observe the differences with (6.18), which are marked in red: when $T \in \mathcal{C}_S$, each smaller derivative term $\frac{\partial f(S \setminus E)}{\partial z_T}$ in the numerator of the derivative is naturally replaced with the corresponding pseudo-derivative term $\frac{\widehat{\partial} f(S \setminus E)}{\widehat{\partial} z_T}$, but crucially, the term $f(S)$ is replaced with $\sum_{E \supseteq S} z_E$. These are the terms corresponding to the "shift" of the set S—i.e., the cuts that separate all points in S from the origin—and hence they form a lower bound on $f(S)$, the expected distance to the closest point in S. This latter change makes the following arguments easier (and indeed, possible), but still maintains the intuition of the derivative being invariant under translations. In §6.8.4, we prove the following lemma, showing it is indeed an upper bound.

Lemma 6.5.5. *The pseudo-derivative is non-negative, and bounds the derivative from above. I.e.,*

$$\max\left(\frac{\partial f(S)}{\partial z_T}, 0\right) \leq \frac{\widehat{\partial} f(S)}{\widehat{\partial} z_T}.$$

Given Lemma 6.5.5, it suffices to upper bound the pseudo-derivative by β_{k-1}, which we do next.

Theorem 6.5.6. *For any $S \subseteq U$ and any $T \neq \emptyset$, we have $\frac{\widehat{\partial} f(S)}{\widehat{\partial} z_T} \leq \beta_{|S \setminus T|}$.*

Theorem 6.5.6 implies our desired bound on the derivative, because $|S \setminus T| \leq k - 1$ for any $S \subseteq U, T \neq \emptyset$. To prove Theorem 6.5.6, we first prove it for the special case when all points in $S \setminus T$ have the same norm (the *uniform* case), and then reduce the general case to this uniform case.

Proof of Theorem 6.5.6: the Uniform Case

The main reason that it is easier to prove Theorem 6.5.6 for the uniform case is because we know the value of $f(S \setminus T)$ exactly which will be equal to the norm of all the points in $S \setminus T$. Otherwise, it is hard to obtain any upper bound to $f(S \setminus T)$ in the general case. Moreover, we know that the uniform property holds true for all subsets of S. This enables us to use the upper bound of Theorem 6.5.6 to derivative terms $\frac{\widehat{\partial} f(S \setminus E)}{\widehat{\partial} z_T}$ by $\beta_{|S \setminus (E \cup T)|}$.

Lemma 6.5.7. *If all points in $S \setminus T$ have the same norm, then $\frac{\widehat{\partial} f(S)}{\widehat{\partial} z_T} \leq \beta_{|S \setminus T|}$.*

Proof. The proof is by induction on $|S|$. If $T \notin \mathcal{C}_S$, we know that $\frac{\widehat{\partial} f(S)}{\widehat{\partial} z_T}$ is either 0 or 1. But $\beta_{|S \setminus T|} \geq 1$ as $\beta_m \geq 1$ for any $m \geq 0$ which implies that we are done. Hence, for $|S| = 1$, we know that $T \notin \mathcal{C}_S$ and we are done. From now on, we can assume $|S| \geq 2$ and $T \in \mathcal{C}_S$. So we use the definition of the pseudo-derivative from (6.19) and use the upper bound of the lemma inductively for the recursive terms $\frac{\widehat{\partial} f(S \setminus E)}{\widehat{\partial} z_T}$ to get:

$$\frac{\widehat{\partial} f(S)}{\widehat{\partial} z_T} \leq \frac{\sum_{E \in \mathcal{C}_S} z_E \cdot \beta_{|S \setminus (E \cup T)|} + f(S \setminus T) - \sum_{E \supseteq S} z_E}{\sum_{E \in \mathcal{C}_S} z_E}. \tag{6.20}$$

In order to show that $\frac{\widehat{\partial} f(S)}{\widehat{\partial} z_T} \leq \beta_{|S \setminus T|}$, it is sufficient to upper bound the RHS of Equation (6.20) by $\beta_{|S \setminus T|}$. Simplifying gives the following sufficient condition

$$f(S \setminus T) \leq \sum_{E \supseteq S} z_E + \sum_{E \in \mathcal{C}_S} z_E \left(\beta_{|S \setminus T|} - \beta_{|S \setminus (T \cup E)|} \right). \tag{6.21}$$

From here on, we will prove Equation (6.21). Since $T \in \mathcal{C}_S$, we know that $|S \setminus T| \geq 1$. All points in $S \setminus T$ have the same norm, so we can write

$$f(S \setminus T) = \frac{1}{|S \setminus T|} \sum_{r \in S \setminus T} \ell(r) \tag{6.22}$$

$$\overset{(6.45)}{=} \frac{1}{|S \setminus T|} \sum_{r \in S \setminus T} \sum_{E : r \in E} z_E \tag{6.23}$$

$$= \sum_E z_E \cdot \frac{|(S \setminus T) \cap E|}{|S \setminus T|}. \tag{6.24}$$

When $E \supseteq S$, the coefficient of z_E in Equation (6.21) is 1. The coefficient of z_E in Equation (6.24) is also 1 because in this case, $S \subseteq E$ and hence $|(S \setminus T) \cap E| = |S \setminus T|$. Otherwise, the coefficients of z_E in Equation (6.21) and Equation (6.24) are $\beta_{|S \setminus T|} - \beta_{|S \setminus (T \cup E)|}$ and $|(S \setminus T) \cap E| / |S \setminus T|$ respectively. It remains to show

$$\frac{|(S \setminus T) \cap E|}{|S \setminus T|} \leq \beta_{|S \setminus T|} - \beta_{|S \setminus (T \cup E)|}. \tag{6.25}$$

We can justify Equation (6.25) because the left hand side is sum of $|(S \setminus T) \cap E|$ copies of $\frac{1}{|S \setminus T|}$, whereas the right hand side is the sum of $|(S \setminus T) \cap E|$ terms, the smallest of them equal to $\frac{1}{|S \setminus T|}$. □

Corollary 6.5.8. *If $T \in \mathcal{C}_S$ and $|S \setminus T| = 1$, we have $\frac{\widehat{\partial} f(S)}{\widehat{\partial} z_T} \leq \beta_1 = 2$.*

Proof. Follows from Lemma 6.5.7 because when there is only one point in $S \setminus T$, we can say that all points in $S \setminus T$ have the same norm. □

6.5.4 Proof of Theorem 6.5.6: Reducing to the Uniform Case

The case for points in $U\backslash T$ have different norms is the technical heart of the proof. In this case, we "lift" the points in such a way that the value of the pseudo-derivative $\frac{\hat{\partial} f(S)}{\hat{\partial} z_T}$ is monotonically increasing, thereby reducing to the uniform case of Section 6.5.3. To begin, we give a few supporting lemmas.

Supporting lemmas

Observation 6.5.9. $\max_{b \in S}(\ell(b)) \geq f(S) \geq \min_{a \in S}(\ell(a))$

Proof. The proof follows from the fact that $f(S)$ is an expectation of norms of points in S. □

Let us denote $S\backslash\{\boldsymbol{p}\}$ simply as $S - \boldsymbol{p}$.

Observation 6.5.10. For any point $\boldsymbol{p} \in S$ such that $|S| \geq 2$, if $E \in \mathcal{C}_S$ and $E \notin \mathcal{C}_{S-\boldsymbol{p}}$, then $E \cap S$ is equal to either $\{\boldsymbol{p}\}$ or $S - \boldsymbol{p}$.

Observation 6.5.11. Let $A = \sum_{j=1}^{m} w_j \cdot v_j / \sum_{j=1}^{m} w_j$ be a weighted average of m items $t_1, \ldots, t_m$ with values $v_1, \ldots, v_m$ weighted by respective weights $w_1, \ldots, w_m$. Adding any number $m' - m$ items of value A (with any weights) does not change the weighted average. I.e., the new weighted average is

$$A' = \frac{\sum_{j=1}^{m} w_j \cdot v_j + \sum_{j=m+1}^{m'} w_j \cdot A}{\sum_{j=1}^{m} w_j + \sum_{j=m+1}^{m'} w_j} = A$$

Observation 6.5.11 is simple but powerful.

Lemma 6.5.12. *Let $\boldsymbol{p}$ be a point in S having the minimum ℓ_1 norm. If $S - \boldsymbol{p} \neq \emptyset$, then*

$$f(S - \boldsymbol{p}) \geq f(S) \geq f(\{\boldsymbol{p}\}) \tag{6.26}$$

Proof. The lower bound follows from Observation 6.5.9. Let us prove the upper bound by induction in $|S|$. If $|S| = 2$, then $f(S - \boldsymbol{p})$ is the norm of the maximum norm point in S which is more than $f(S)$ by Observation 6.5.10. If $|S| \geq 3$, then $|S - \boldsymbol{p}| \geq 2$ so we will use Equation (6.17) to expand both $f(S)$ and $f(S - \boldsymbol{p})$ as

$$f(S - \boldsymbol{p}) = \frac{\sum_{E \in \mathcal{C}_{S-\boldsymbol{p}}} z_E \cdot f((S - \boldsymbol{p})\backslash E)}{\sum_{E \in \mathcal{C}_{S-\boldsymbol{p}}} z_E},\ f(S) = \frac{\sum_{E \in \mathcal{C}_S} z_E \cdot f(S\backslash E)}{\sum_{E \in \mathcal{C}_S} z_E}. \tag{6.27}$$

Using Observation 6.5.11 and Observation 6.5.10, we can re-write $f(S - \boldsymbol{p})$ as

$$f(S - \boldsymbol{p}) = \frac{\sum_{E \in \mathcal{C}_{S-\boldsymbol{p}}} z_E \cdot f((S - \boldsymbol{p})\backslash E) + z_{\{\boldsymbol{p}\}} \cdot f(S - \boldsymbol{p}) + z_{S-\boldsymbol{p}} \cdot f(S - \boldsymbol{p})}{\sum_{E \in \mathcal{C}_S} z_E}. \tag{6.28}$$

In the numerators of $f(S - \boldsymbol{p})$ and $f(S)$, the coefficient of z_E in $f(S - \boldsymbol{p})$ is more than that of $f(S)$ either by induction on $|S|$ ($S \leftarrow S\backslash E$) or using the fact that $f(S - \boldsymbol{p}) \geq f(\{\boldsymbol{p}\})$ from Observation 6.5.9. □

From now, many other lemmas will have their proofs very similar to that of Lemma 6.5.12 where we compare the function values of two different sets by expanding the functions and using Observation 6.5.11 to normalize the denominators and then use induction and other arguments to conclude.

The lifting operation

Now let us define the lift operation. For a set $S' = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_{k'}\} \subseteq U$, consider the following trajectory for the $\boldsymbol{z}$ vector

$$z_E(t) := \begin{cases} z_E + t & E = \{\boldsymbol{p}\},\ \boldsymbol{p} \in S' \\ z_E & \text{otherwise .} \end{cases} \tag{6.29}$$

Given this trajectory for the embedding, we are interested in how this changes the function value. Let us define

$$\frac{\partial f(S)}{\partial S'} := \left.\frac{\mathrm{d} f_t(S)}{\mathrm{d}t}\right|_{t=0}.$$

Lemma 6.5.13. *Let S' be a subset of U. Then,*

1. *If $S \cap S' = \emptyset$, $\frac{\partial f(S)}{\partial S'} = 0$.*
2. *If $|S| = 1$, $S \subseteq S'$, $\frac{\partial f(S)}{\partial S'} = 1$.*
3. *If $|S| \geq 2$,*

$$\frac{\partial f(S)}{\partial S'} = \frac{\sum_{E \in \mathcal{C}_S} z_E \cdot \frac{\partial f(S \backslash E)}{\partial S'} + \sum_{\boldsymbol{p}_i \in S \cap S'} \left(f(S - \{\boldsymbol{p}_i\}) - f(S)\right)}{\sum_{E \in \mathcal{C}_S} z_E}. \tag{6.30}$$

Moreover, if points in $S' \cap S$ if any, have the least norm out of points in S, then $\frac{\partial f(S)}{\partial S'} \geq 0$.

Proof. Since $f(S)$ is purely a function of points in S, and hence only depends on variables z_E such that $\boldsymbol{p} \in E$ for some $\boldsymbol{p} \in S$. But the only variables that change with t are $z_{\{\boldsymbol{p}'\}}, \boldsymbol{p}' \in S'$. Since $S' \cap S = \emptyset$, we can conclude statement 1. When $S = \{\boldsymbol{p}\}$ and $\boldsymbol{p} \in S'$, $f_t(S) = \ell_t(\boldsymbol{p})$. Since $z_{\{\boldsymbol{p}\}}$ is the only variable that is changing and $\ell(\boldsymbol{p})$ contains this, we can conclude statement 2. Once we know that $|S| = 2$, using Equation (6.17) and applying the $\frac{\partial}{\partial S'}$ operator on both sides gives

$$\frac{\partial f(S)}{\partial S'} = \frac{\sum_{E \in \mathcal{C}_S} \left(z_E \cdot \frac{\partial f(S \backslash E)}{\partial s'} + \frac{\partial z_E}{\partial S'} \cdot f(S \backslash E)\right)}{\sum_{E \in \mathcal{C}_S} z_E} - \frac{f(S)}{\sum_{E \in \mathcal{C}_S} z_E} \cdot \frac{\partial \sum_{E \in \mathcal{C}_S} z_E}{\partial S'}. \tag{6.31}$$

Using the fact that the only variables that are changing with t are $z_{\{\boldsymbol{p}\}}$ for $\boldsymbol{p} \in S'$ in Equation (6.31) gives Equation (6.30). Let us argue that $\frac{\partial f(S)}{\partial S'} \geq 0$ when points in $S \cap S'$ have the least norm of points in S using induction on $|S|$. For $|S| = 1$, using statement 2 or statement 1, we are done. Otherwise, using Equation (6.30), we have that the recursive derivative terms $\frac{\partial f(S \backslash E)}{\partial S'}$ in the numerator of Equation (6.30) are non-negative by inductive hypothesis. From Lemma 6.5.12, we know that $f(S - \boldsymbol{p}_i) - f(S) \geq 0$ for every $\boldsymbol{p}_i \in S \cap S'$. Combining both these observations, we can conclude statement 3 and hence the lemma. □

Lemma 6.5.14. *If S' is the set of points in $U \backslash T$ of minimum norm, then*

$$\frac{\partial}{\partial S'}\left(\frac{\hat{\partial} f(S)}{\hat{\partial} z_T}\right) \geq 0.$$

Proof. The proof is by induction on $|S|$. If $T \notin \mathcal{C}_S$, this is trivially true because $\frac{\hat{\partial} f(S)}{\hat{\partial} z_T}$ is either 0 or 1. So for $|S| = 1$, we are done. Otherwise, we have $|S| \geq 2$ and $T \in \mathcal{C}_S$. Recall from Equation (6.19) that

$$\frac{\hat{\partial} f(S)}{\hat{\partial} z_T} = \frac{\sum_{E \in \mathcal{C}_S} z_E \cdot \frac{\hat{\partial} f(S \backslash E)}{\hat{\partial} z_T} + f(S \backslash T) - \sum_{E \supseteq S} z_E}{\sum_{E \in \mathcal{C}_S} z_E}.$$

Applying the operator $\frac{\partial}{\partial S'}$ on both sides gives

$$\frac{\sum_{E \in \mathcal{C}_S} z_E \cdot \frac{\partial}{\partial S'}\left(\frac{\hat{\partial} f(S \backslash E)}{\hat{\partial} z_T}\right) + \frac{\partial f(S \backslash T)}{\partial S'} - \frac{\partial}{\partial S'}\left(\sum_{E \supseteq S} z_E\right) + \sum_{\boldsymbol{p} \in S' \cap S}\left(\frac{\hat{\partial} f(S - \boldsymbol{p})}{\hat{\partial} z_T} - \frac{\hat{\partial} f(S)}{\hat{\partial} z_T}\right)}{\sum_{E \in \mathcal{C}_S} z_E}.$$

Since $|S| \geq 2$, the term $\frac{\partial}{\partial S'}\left(\sum_{E \supseteq S} z_E\right)$ is zero. Using this, what remains is

$$\frac{\sum_{E \in \mathcal{C}_S} z_E \cdot \frac{\partial}{\partial S'}\left(\frac{\hat{\partial} f(S \backslash E)}{\hat{\partial} z_T}\right) + \frac{\partial f(S \backslash T)}{\partial S'} + \sum_{\boldsymbol{p} \in S \cap S'}\left(\frac{\hat{\partial} f(S - \boldsymbol{p})}{\hat{\partial} z_T} - \frac{\hat{\partial} f(S)}{\hat{\partial} z_T}\right)}{\sum_{E \in \mathcal{C}_S} z_E}.$$

This means, it is sufficient to show that when $T \in \mathcal{C}_S$,

$$\frac{\partial f(S \backslash T)}{\partial S'} + \sum_{\boldsymbol{p} \in S \cap S'}\left(\frac{\hat{\partial} f(S - \boldsymbol{p})}{\hat{\partial} z_T} - \frac{\hat{\partial} f(S)}{\hat{\partial} z_T}\right) \geq 0. \tag{6.32}$$

Because, the recursive derivatives $\frac{\partial}{\partial S'}\left(\frac{\hat{\partial} f(S \backslash E)}{\hat{\partial} z_T}\right)$ are at least zero by induction on $|S|$. The proof of Equation (6.32) is now given in Lemma 6.5.15. □

Lemma 6.5.15. *Let $S' = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_{k'}\}$ be the set of points in $U \backslash T$ of minimum norm. For $T \in \mathcal{C}_S$,*

$$\frac{\partial f(S \backslash T)}{\partial S'} + \sum_{\boldsymbol{p} \in S \cap S'}\left(\frac{\hat{\partial} f(S - \boldsymbol{p})}{\hat{\partial} z_T} - \frac{\hat{\partial} f(S)}{\hat{\partial} z_T}\right) \geq 0.$$

Proof. The proof is by induction on $|S|$. Let us work out boundary cases first. If $S \cap S' = (S \backslash T) \cap S' = \emptyset$, then $\frac{\partial f(S \backslash T)}{\partial S'} = 0$ and the summation is empty which makes the entire expression equal to 0 in which case, we are fine. So from now, we can assume $S \cap S' \neq \emptyset$. If $|S \backslash T| = 1$, then $S \backslash T = \{\boldsymbol{p}^*\} \subseteq S'$, which implies $\frac{\partial f(S \backslash T)}{\partial S'} = 1$. The expression in this case is simply

$$1 + \frac{\hat{\partial} f(S - \boldsymbol{p}^*)}{\hat{\partial} z_T} - \frac{\hat{\partial} f(S)}{\hat{\partial} z_T} = 2 - \frac{\hat{\partial} f(S)}{\hat{\partial} z_T}$$

because $S - \boldsymbol{p}^* \in T$. For this boundary case, it remains to prove that $\frac{\hat{\partial} f(S)}{\hat{\partial} z_T} \leq 2$. This is taken care of Corollary 6.5.8. For the base case $|S| = 2$, we have $|S \backslash T| = 1$ in which case, we are done from the preceding arguments. Otherwise, if $|S \backslash T| \geq 2$ we can expand all the derivative terms. But before we do that, we need Observation 6.5.11 so that we have the same denominator for all the three derivatives. Now using Equation (6.30) and the fact that $(S \backslash T) \cap S' = S \cap S'$,

$$\begin{aligned}
\frac{\partial f(S \backslash T)}{\partial S'} &= \frac{\sum_{E \in \mathcal{C}_{S \backslash T}} z_E \cdot \frac{\partial f(S \backslash (T \cup E))}{\partial S'} + \sum_{\boldsymbol{p} \in S \cap S'} \left(f(S \backslash (T \cup \{\boldsymbol{p}\})) - f(S \backslash T) \right)}{\sum_{E \in \mathcal{C}_{S \backslash T}} z_E} \\
&= \frac{\sum_{E \in \mathcal{C}_{S \backslash T}} z_E \cdot \frac{\partial f(S \backslash (T \cup E))}{\partial S'} + \sum_{E \in \mathcal{C}_S \backslash \mathcal{C}_{S \backslash T}} z_E \cdot \frac{\partial f(S \backslash T)}{\partial S'} + \sum_{\boldsymbol{p} \in S \cap S'} \left(f(S \backslash (T \cup \{\boldsymbol{p}\})) - f(S \backslash T) \right)}{\sum_{E \in \mathcal{C}_S} z_E} \\
&\geq \frac{\sum_{E \in \mathcal{C}_{S \backslash T}} z_E \cdot \frac{\partial f(S \backslash (T \cup E))}{\partial S'} + \sum_{E \in \mathcal{C}_S \backslash \mathcal{C}_{S \backslash T}} z_E \cdot \frac{\partial f(S \backslash T)}{\partial S'}}{\sum_{E \in \mathcal{C}_S} z_E}. \qquad (6.33)
\end{aligned}$$

Using Equation (6.19) and the fact that $T \in \mathcal{C}_{S - \boldsymbol{p}}$ for any $\boldsymbol{p} \in S \cap S'$,

$$\begin{aligned}
\frac{\hat{\partial} f(S - \boldsymbol{p})}{\hat{\partial} z_T} &= \frac{\sum_{E \in \mathcal{C}_{S - \boldsymbol{p}}} z_E \cdot \frac{\hat{\partial} f(S \backslash (E \cup \{\boldsymbol{p}\}))}{\hat{\partial} z_T} + f(S \backslash (T \cup \{\boldsymbol{p}\})) - \sum_{E \supseteq S - \boldsymbol{p}} z_E}{\sum_{E \in \mathcal{C}_{S - \boldsymbol{p}}} z_E} \\
&= \frac{\sum_{E \in \mathcal{C}_{S - \boldsymbol{p}}} z_E \cdot \frac{\hat{\partial} f(S \backslash (E \cup \{\boldsymbol{p}\}))}{\hat{\partial} z_T} + \sum_{E \in \mathcal{C}_S \backslash \mathcal{C}_{S - \boldsymbol{p}}} z_E \cdot \frac{\hat{\partial} f(S \backslash \{\boldsymbol{p}\})}{\hat{\partial} z_T} + f(S \backslash (T \cup \{\boldsymbol{p}\})) - \sum_{E \supseteq S - \boldsymbol{p}} z_E}{\sum_{E \in \mathcal{C}_S} z_E}.
\end{aligned}$$

Using Equation (6.19),

$$\frac{\hat{\partial} f(S)}{\hat{\partial} z_T} = \frac{\sum_{E \in \mathcal{C}_S} z_E \cdot \frac{\hat{\partial} f(S \backslash E)}{\hat{\partial} z_T} + f(S \backslash T) - \sum_{E \supseteq S} z_E}{\sum_{E \in \mathcal{C}_S} z_E}.$$

The numerator of $\frac{\hat{\partial} f(S - \boldsymbol{p})}{\hat{\partial} z_T} - \frac{\hat{\partial} f(S)}{\hat{\partial} z_T}$ is

$$\begin{aligned}
&\sum_{E \in \mathcal{C}_{S - \boldsymbol{p}}} z_E \left(\frac{\hat{\partial} f(S \backslash (E \cup \{\boldsymbol{p}\}))}{\hat{\partial} z_T} - \frac{\hat{\partial} f(S \backslash E)}{\hat{\partial} z_T} \right) + \sum_{E \in \mathcal{C}_S \backslash \mathcal{C}_{S - \boldsymbol{p}}} z_E \left(\frac{\hat{\partial} f(S \backslash \{\boldsymbol{p}\})}{\hat{\partial} z_T} - \frac{\hat{\partial} f(S \backslash E)}{\hat{\partial} z_T} \right) \\
&- \sum_{E : S \backslash E = \{\boldsymbol{p}\}} z_E + \left(f(S \backslash (T \cup \{\boldsymbol{p}\})) - f(S \backslash T) \right) \\
&\geq \sum_{E \in \mathcal{C}_{S - \boldsymbol{p}}} z_E \left(\frac{\hat{\partial} f(S \backslash (E \cup \{\boldsymbol{p}\}))}{\hat{\partial} z_T} - \frac{\hat{\partial} f(S \backslash E)}{\hat{\partial} z_T} \right) + \sum_{E \in \mathcal{C}_S \backslash \mathcal{C}_{S - \boldsymbol{p}}} z_E \left(\frac{\hat{\partial} f(S \backslash \{\boldsymbol{p}\})}{\hat{\partial} z_T} - \frac{\hat{\partial} f(S \backslash E)}{\hat{\partial} z_T} \right) \\
&- \sum_{E : S \backslash E = \{\boldsymbol{p}\}} z_E. \qquad (6.34)
\end{aligned}$$

Note that the condition $E \in \mathcal{C}_S \backslash \mathcal{C}_{S - \boldsymbol{p}}$ holds only when $\boldsymbol{p} \in S$ and $E \cap S$ is either $S \backslash \{\boldsymbol{p}\}$ or $\{\boldsymbol{p}\}$ from Observation 6.5.10. When $E \cap S = \{\boldsymbol{p}\}$, the second derivative term in Equation (6.34) is

zero. Otherwise, it is $\frac{\hat{\partial} f(S \backslash \{\boldsymbol{p}\})}{\hat{\partial} z_T} - \frac{\hat{\partial} f(\{\boldsymbol{p}\})}{\hat{\partial} z_T} = \frac{\hat{\partial} f(S \backslash \{\boldsymbol{p}\})}{\hat{\partial} z_T}$. Using these observations, Equation (6.34) can be simplified and can be re-written as

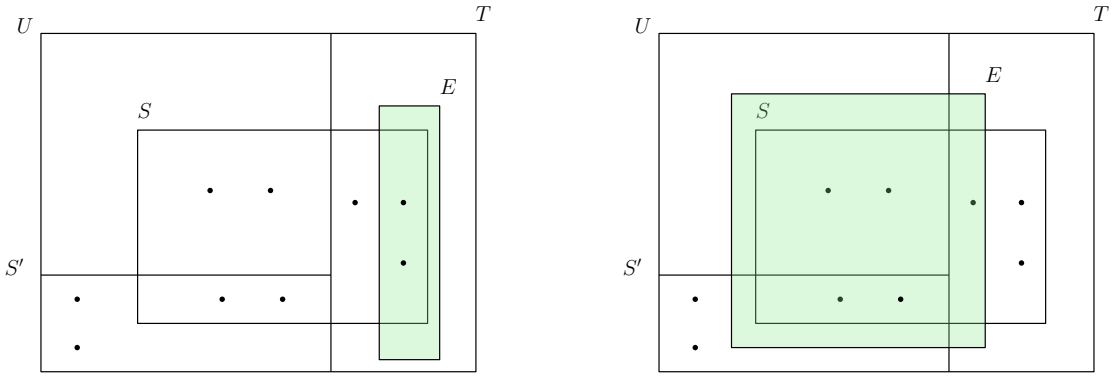
$$\sum_{E \in \mathcal{C}_{S-\boldsymbol{p}}} z_E \left(\frac{\hat{\partial} f(S \backslash (E \cup \{\boldsymbol{p}\}))}{\hat{\partial} z_T} - \frac{\hat{\partial} f(S \backslash E)}{\hat{\partial} z_T} \right) + \sum_{E : S \backslash E = \{\boldsymbol{p}\}} z_E \left(\frac{\hat{\partial} f(S \backslash \{\boldsymbol{p}\})}{\hat{\partial} z_T} - 1 \right). \quad (6.35)$$

The remaining numerator of $\frac{\partial f(S \backslash T)}{\partial S'}$ from Equation (6.33) is

$$\sum_{E \in \mathcal{C}_{S \backslash T}} z_E \cdot \frac{\partial f(S \backslash (T \cup E))}{\partial S'} + \sum_{E \in \mathcal{C}_S \backslash \mathcal{C}_{S \backslash T}} z_E \cdot \frac{\partial f(S \backslash T)}{\partial S'}. \quad (6.36)$$

It remains to prove that the sum of Equation (6.36), and Equation (6.35) summed over all $\boldsymbol{p} \in S \cap S'$ is non-negative. Let us do that by carefully partitioning all the E into groups and do a case analysis.

Case 1: ($E \in \mathcal{C}_S \backslash \mathcal{C}_{S \backslash T}$) First, observe that this happens only when either $\emptyset \neq S \cap E \subseteq S \cap T$ or $\emptyset \neq S \backslash E \subseteq S \cap T$ as shown in figures Figure 6.3a and Figure 6.3b respectively. Observe



(a) Case 1a *(b) Case 1b*

Figure 6.3: Case 1

that in either case, for that E, we have $E \in \mathcal{C}_{S-\boldsymbol{p}}$ for any $\boldsymbol{p} \in S \cap S'$.

Case 1a: ($\emptyset \neq S \cap E \subseteq S \cap T$) In this case, the coefficient of z_E in the sum of Equation (6.36) and, Equation (6.35) summed over all $\boldsymbol{p} \in S \cap S'$ is

$$\frac{\partial f(S \backslash T)}{\partial S'} + \sum_{\boldsymbol{p} \in S \cap S'} \left(\frac{\hat{\partial} f(S \backslash (E \cup \{\boldsymbol{p}\}))}{\hat{\partial} z_T} - \frac{\hat{\partial} f(S \backslash E)}{\hat{\partial} z_T} \right). \quad (6.37)$$

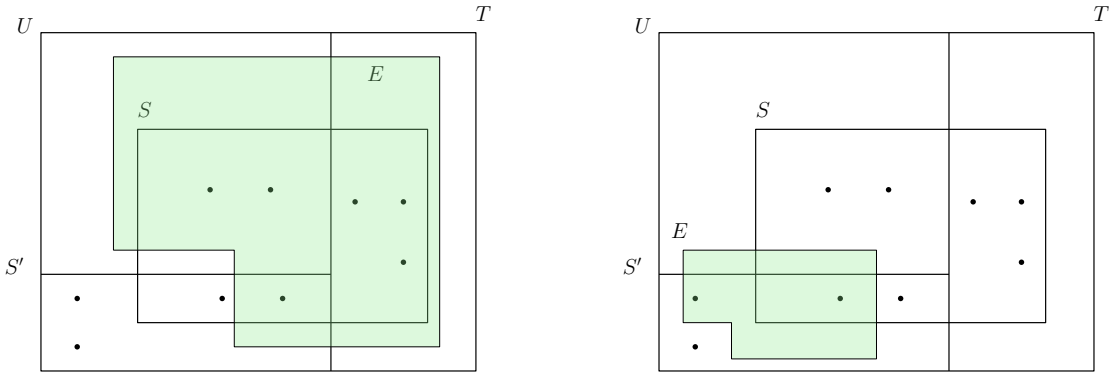
Which we can re-write as

$$\frac{\partial f((S \backslash E) \backslash T)}{\partial S'} + \sum_{\boldsymbol{p}_r \in (S \backslash E) \cap S'} \left(\frac{\hat{\partial} f((S \backslash E) \backslash \{\boldsymbol{p}\})}{\hat{\partial} z_T} - \frac{\hat{\partial} f(S \backslash E)}{\hat{\partial} z_T} \right).$$

This, we can argue is non-negative by induction on $|S|$ by setting $S \leftarrow S \backslash E$. But note that before using inductive hypothesis, we need $T \in \mathcal{C}_{S\backslash E}$ which is not guaranteed when $E = S \cap T$. But in that case, the terms inside the summation of Equation (6.37) are zero because $T \cap (S\backslash E) = \emptyset$ which implies that the coefficient is $\frac{\partial f(S\backslash T)}{\partial S'} = \frac{\partial f(S\backslash(T\cup E))}{\partial S'} \geq 0$ from Lemma 6.5.13.

Case 1b: ($\emptyset \neq S\backslash E \subseteq S \cap T$) In this case, the coefficient of z_E is the same expression as Equation (6.37) but observe that $S\backslash(E \cup \{\boldsymbol{p}\}) = S\backslash E$ which implies that the coefficient is simply $\frac{\partial f(S\backslash T)}{\partial S'}$ which we know is non-negative from Lemma 6.5.13.

Case 2: ($E \in \mathcal{C}_{S\backslash T}$) Let us branch based on whether there exists $\boldsymbol{p} \in S \cap S'$ such that $E \notin \mathcal{C}_{S-\boldsymbol{p}}$. This can happen in two ways. First, there exists a point $\boldsymbol{p}^* \in S \cap S'$ such that $S\backslash E = \{\boldsymbol{p}^*\}$. Second, there exists a point point $\boldsymbol{p}^* \in S \cap S'$ such that $S\backslash E = S\backslash\{\boldsymbol{p}^*\}$



(a) Case 2a *(b) Case 2b*

Figure 6.4: Case 2

Case 2a: ($\exists \boldsymbol{p}^* \in S \cap S', S\backslash E = \{\boldsymbol{p}^*\}$) First, note that $E \in \mathcal{C}_{S\backslash T}$ for such a cut (see Figure 6.4a). Observe that for any $\boldsymbol{p} \neq \boldsymbol{p}^* \in S \cap S'$, we have $S\backslash(E \cup \{\boldsymbol{p}\}) = S\backslash E$ and there cannot be a different $\boldsymbol{p}$ such that $S\backslash E = \{\boldsymbol{p}\}$ for the same E. This implies that the coefficient of z_E in the sum of Equation (6.36) and, Equation (6.35) summed over all $\boldsymbol{p} \in S \cap S'$ is

$$\begin{aligned}
&\frac{\partial f(S\backslash(T \cup E))}{\partial S'} + \frac{\hat{\partial} f(S\backslash\{\boldsymbol{p}^*\})}{\hat{\partial} z_T} - 1 \\
&= \frac{\partial f(\{\boldsymbol{p}^*\})}{\partial S'} + \frac{\hat{\partial} f(S\backslash\{\boldsymbol{p}^*\})}{\hat{\partial} z_T} - 1 \\
&= \frac{\hat{\partial} f(S\backslash\{\boldsymbol{p}^*\})}{\hat{\partial} z_T} \geq 0
\end{aligned}$$

Case 2b: ($\exists \boldsymbol{p}^* \in S \cap S', S\backslash E = S\backslash\{\boldsymbol{p}^*\}$) First, note that $E \in \mathcal{C}_{S\backslash T}$ for such a cut (see Figure 6.4b). Observe that for any $\boldsymbol{p} \neq \boldsymbol{p}^* \in S \cap S'$, we have $E \in \mathcal{C}_{S-\boldsymbol{p}}$. This implies

that the coefficient of z_E in the sum of Equation (6.36) and, Equation (6.35) summed over all $\boldsymbol{p} \in S \cap S'$ is

$$\frac{\partial f(S\backslash(T \cup E))}{\partial S'} + \sum_{\boldsymbol{p} \neq \boldsymbol{p}^* \in S \cap S'} \left(\frac{\hat{\partial} f(S\backslash(E \cup \{\boldsymbol{p}\}))}{\hat{\partial} z_T} - \frac{\hat{\partial} f(S\backslash E)}{\hat{\partial} z_T} \right)$$
$$= \frac{\partial f((S\backslash\{\boldsymbol{p}^*\})\backslash T)}{\partial S'} + \sum_{\boldsymbol{p} \in (S\backslash \boldsymbol{p}^*) \cap S'} \left(\frac{\hat{\partial} f((S\backslash\{\boldsymbol{p}^*\})\backslash\{\boldsymbol{p}\})}{\hat{\partial} z_T} - \frac{\hat{\partial} f(S\backslash\{\boldsymbol{p}^*\})}{\hat{\partial} z_T} \right).$$

This, we can argue is non-negative by induction on $|S|$ ($S \leftarrow S\backslash\{\boldsymbol{p}^*\}$). Note that $T \in \mathcal{C}_{S-\boldsymbol{p}^*}$ holds so we can use induction hypothesis.

Case 2c: ($E \in \mathcal{C}_{S-\boldsymbol{p}}, \forall \boldsymbol{p} \in S \cap S'$) In this case, the coefficient of z_E is simply,

$$\frac{\partial f(S\backslash(T \cup E))}{\partial S'} + \sum_{\boldsymbol{p} \in S \cap S'} \left(\frac{\hat{\partial} f(S\backslash(E \cup \{\boldsymbol{p}\}))}{\hat{\partial} z_T} - \frac{\hat{\partial} f(S\backslash E)}{\hat{\partial} z_T} \right)$$
$$= \frac{\partial f((S\backslash E)\backslash T)}{\partial S'} + \sum_{\boldsymbol{p} \in (S\backslash E) \cap S'} \left(\frac{\hat{\partial} f((S\backslash E)\backslash\{\boldsymbol{p}\}))}{\hat{\partial} z_T} - \frac{\hat{\partial} f(S\backslash E)}{\hat{\partial} z_T} \right).$$

This, we can again argue is non-negative by induction on $|S|$ ($S \leftarrow E$). Note that if $T \notin \mathcal{C}_{S\backslash E}$, then $E \supseteq S \cap T$ in which case, the terms in the summation are zero and we are done directly without induction.

This completes the proof of Lemma 6.5.15. □

We can now wrap up: Lemma 6.5.15 was the missing piece in the proof of Lemma 6.5.14. In turn, because of Lemma 6.5.14 we can assume all points in $S\backslash T$ to have the same norm. Lemma 6.5.7 shows the desired bound of Theorem 6.5.6 for this uniform case, completing the proof.

6.6 Price of Explainability with General Threshold Cuts

Generalized Threshold Cuts and Trees. In Section 6.1.3, we defined threshold cuts as hyperplanes of the form $x_i = \theta$. We now generalize this to arbitrary hyperplanes of the form $\langle \boldsymbol{a}, \boldsymbol{x} \rangle = \theta$ for some direction $\boldsymbol{a} \in \mathbb{R}^d$. We denote such a hyperplane by $(\boldsymbol{a}, \theta)$. A *generalized threshold tree* T is a binary tree where each internal node u corresponds to a cut $(\boldsymbol{a}_u, \theta_u)$. Let $B_u \subseteq \mathbb{R}^d$ denote the region associated with node $u \in T$. For the root node r, we have $B_r := \mathbb{R}^d$. If $l(u)$ and $r(u)$ denote the left and right children of u, then:

$$B_{l(u)} := B_u \cap \{\boldsymbol{x} \mid \langle \boldsymbol{a}_u, \boldsymbol{x} \rangle \leq \theta_u\},$$
$$B_{r(u)} := B_u \cap \{\boldsymbol{x} \mid \langle \boldsymbol{a}_u, \boldsymbol{x} \rangle > \theta_u\}.$$

This generalization allows us to extend the notion of explainability to include such threshold trees.

Given a dataset $\mathcal{X} = \{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^n\} \subseteq \mathbb{R}^d$, a set of centers $\mathcal{U} = \{\boldsymbol{\mu}^1, \ldots, \boldsymbol{\mu}^k\} \subseteq \mathbb{R}^d$, and an assignment map $\pi : \mathcal{X} \to \mathcal{U}$, the k-medians cost under the ℓ_q norm is defined as:

$$\mathrm{cost}_q^{\text{median}}(\pi, \mathcal{U}) = \sum_{\boldsymbol{x} \in \mathcal{X}} \|\boldsymbol{x} - \pi(\boldsymbol{x})\|_q. \tag{6.38}$$

6.6.1 k-Medians with ℓ_2 Norm

We now describe a natural extension of the random thresholds algorithm (from Section 6.2.1) to the ℓ_2 setting using general hyperplanes. Assume the dataset $\mathcal{X}$ lies within the ball $B(0, r)$ for some $r \in \mathbb{R}_{\geq 0}$. Initialize the threshold tree T with a single root corresponding to $\mathbb{R}^d$. While some leaf of T contains more than one center, perform the following:

- Sample a random Gaussian vector $\boldsymbol{g} \sim \mathcal{N}_d(\mathbf{0}, I)$.
- Sample $\theta \in [-r, r]$ uniformly at random.
- For each leaf u of T, if the cut $(\frac{\boldsymbol{g}}{\|\boldsymbol{g}\|_2}, \theta)$ separates any pair of centers in $B_u \cap \mathcal{U}$, then partition the region using this cut.

This process, called the *random hyperplanes algorithm*, continues until all centers are isolated in disjoint leaves.

Analyzing Explainability. As with the random thresholds algorithm, this method is independent of the dataset $\mathcal{X}$ and is invariant under translations and scalings. Therefore, it suffices to analyze the expected cost for a single point at the origin, using the ℓ_2 norm and hyperplane-based splits.

Let $U := \mathcal{U}$ denote the set of centers. Let $D_2^{\text{med}}(U)$ be the distribution over hyperplane cuts induced by the random hyperplanes algorithm. For any subset $T \subseteq U$, define:

$$z_T := \Pr_{(\boldsymbol{a}, \theta) \sim D_2^{\text{med}}(U)} \left[U \cap \{\boldsymbol{x} \mid \mathrm{sign}(\theta) \cdot \langle \boldsymbol{a}, \boldsymbol{x} \rangle \geq |\theta|\} = T \right]. \tag{6.39}$$

Since the distribution of cuts depends only on U, it remains fixed throughout. Let $f(S)$ denote the expected ℓ_2 norm of the unique center in the leaf region that contains the origin, assuming the current region includes centers $S \subseteq U$. Then we have the recursive relation:

$$f(S) = \frac{\sum_{E \in \mathcal{C}_S} z_E \cdot f(S \setminus E)}{\sum_{E \in \mathcal{C}_S} z_E}, \qquad f(\{\boldsymbol{p}\}) = \|\boldsymbol{p}\|_2,$$

where $\mathcal{C}_S$ is the collection of possible subsets E corresponding to valid cuts.

Lemma 6.6.1. *Let $\boldsymbol{p} \in U$ be a center. Then:*

$$\sum_{T \ni \boldsymbol{p}} z_T = \|\boldsymbol{p}\|_2 \cdot g(r, d),$$

for some function $g(r, d)$ depending only on r and the dimension d.

Proof. The left-hand side represents the probability that a hyperplane drawn from $D_2^{\text{med}}(U)$ separates $\boldsymbol{p}$ from the origin:

$$\sum_{T \ni \boldsymbol{p}} z_T = \Pr_{(\boldsymbol{a},\theta)\sim D_2^{\text{med}}(U)} [\boldsymbol{p} \text{ is separated from the origin by the hyperplane } \langle \boldsymbol{a}, \boldsymbol{x} \rangle = \theta] .$$

Due to rotational symmetry, we may assume $\boldsymbol{a} = \boldsymbol{e}_1$ and consider $\boldsymbol{p}$ as a uniformly random point on the sphere of radius $\|\boldsymbol{p}\|_2$. Then, the probability that $\boldsymbol{p}$ is separated from the origin by the plane $x_1 = \theta$, for $\theta \sim \text{Uniform}[-r, r]$, is proportional to the expected ℓ_1 norm of a uniformly random point on the unit sphere S^d:

$$= \frac{\|\boldsymbol{p}\|_2}{2rd} \cdot \mathbb{E}[\ell_1 \text{ norm of a random point on } S^d].$$

□

Using Lemma 6.6.1 and the recurrence for $f(U)$, we obtain the bound $f(U) \leq \beta_{k-1} \cdot \min_{\boldsymbol{q}\in U} \|\boldsymbol{q}\|_2$, implying that the price of explainability is at most β_{k-1}.

Alternative Approach via Metric Embeddings. An alternate method to establish a bound on POE is via metric embeddings. Specifically, consider the point set $\mathcal{X} \cup \mathcal{U}$, and embed it linearly and isometrically from the ℓ_2 space into an ℓ_1 space. Then, apply the random thresholds algorithm (designed for ℓ_1) in the embedded space. Finally, interpret the resulting threshold tree as a generalized threshold tree in the original ℓ_2 space.

6.7 Future Directions

Our work leaves several important avenues for future research:

- **Closing the Approximation Gap for k-Means.** For k-median, the price of explainability is settled at $\Theta(\ln k)$. A significant gap remains for k-means, however, between the $\Omega(\ln k)$ hardness of approximation and the best-known $O(k \ln \ln k)$ upper bound. Closing this gap—either by improving the approximation algorithm or tightening the hardness result—is a central open problem.
- **Exploring General Explanation Models.** Our framework relies on explanations from axis-aligned decision trees. This could be generalized in several ways:
  - **General Hyperplane Partitions.** A natural extension is to allow cluster boundaries defined by arbitrary (non-axis-aligned) hyperplanes. It is currently unclear if this added expressive power actually reduces the price of explainability, as the best known upper and lower bounds remain unchanged from the axis-aligned model. A key question is whether more expressive models can provably improve the explainability-cost trade-off.

- **Feature-Based Explainability.** Another direction is to define an explainable clustering via feature selection. For instance, an explanation could be a pair (S, f), where $S \subseteq [d]$ is a small subset of features (e.g., $|S| = k$) and $f : \mathbb{R}^S \to [k]$ is a clustering function over that subset. This framework captures a form of feature-based attribution and connects our work to the broader goals of interpretable machine learning.

6.8 Appendix for Chapter 6

6.8.1 Proofs from Section 6.2

Proof of Lemma 6.2.2. For any client $\boldsymbol{x} \in \boldsymbol{X}$. Since the Random Thresholds algorithm is translation and scaling invariant, we imagine that $\boldsymbol{x} = \mathbf{0}$. Now the expected cost incurred by this client is at most the distance to the unique center in its leaf region in the tree produced by the Random Thresholds algorithm, which is at most $\alpha(|\mathcal{U}|)$. The claim now follows by scaling by the true distance to the closest center $\|\boldsymbol{x} - \pi(\boldsymbol{x})\|$, summing over all $\boldsymbol{x} \in \boldsymbol{X}$, and using linearity of expectations. □

Going from ℓ_1 Metrics to Cut Metrics

It is known that point set in ℓ_1 can be written as a non-negative sum of cut metrics [68]; we give the details here for completeness. Given a point set $V \in \mathbb{R}^d$, define $\ell_i := \min_{\boldsymbol{v} \in V} v_i$ and $u_i = \max_{\boldsymbol{v} \in V} v_i$. Then $L_1(V) := \sum_{i=1}^{d} (u_i - \ell_i)$, and $D_1(V)$ is the uniform distribution over $\{(i, \theta) \mid \theta \in [\ell_i, u_i]\}$. Define for any $S \subseteq V$ the non-negative quantity

$$z_S = L_1(V) \cdot \Pr_{(i,\theta) \sim D_1(V)} [(V \cap \{\boldsymbol{x} \mid \text{sign}(\theta) \cdot x_i \geq \theta\}) = S].$$

This is a scaled version of the probability that for a random threshold cut, the points of V in the halfspace not containing the origin equals S. A direct calculation shows that for all $\boldsymbol{p}, \boldsymbol{q} \in V$, we have

$$\|\boldsymbol{p} - \boldsymbol{q}\|_1 = \sum_S z_S \mathbb{1}_{(|S \cap \{\boldsymbol{p}, \boldsymbol{q}\}| = 1)}.$$

Now define an ℓ_1-embedding $\varphi : V \to \mathbb{R}_{\geq 0}^{2^{|V|}}$ by setting, for any $S \subseteq V$,

$$\varphi(\boldsymbol{p})_S = \sum_S z_S \mathbb{1}_{(\boldsymbol{p} \in S)}.$$

Again, $\|\boldsymbol{p} - \boldsymbol{q}\|_1 = \|\varphi(\boldsymbol{p}) - \varphi(\boldsymbol{q})\|_1$. Moreover, if the origin belongs to V, we get $\varphi(\mathbf{0}) = \mathbf{0}$.

Proof of Lemma 6.2.4

Let $\mathcal{S} = \langle S_1, \ldots, S_{2^k} \rangle$ be the sequence of cuts in increasing order of their sample values X_S. (We add the subsets with $z_S = 0$ at the end of the sequence in some fixed but arbitrary order.) However, it is not true that we remove points from U in this order: we need to reject cuts that do not cross the remaining set U. (We say a set A *crosses* B if $B \cap A$ is a non-empty proper subset

of B: i.e., if both $B \setminus A$ and $B \cap A$ are non-empty.) Hence, we recast the "last-point" process again as follows. Given any subset of points $U \subseteq V$:

1. Define $U^0 := U$ and $\mathcal{S}^0 = \langle\rangle$. In general, U^r is the set of points remaining after considering sets $S_1, \ldots, S_r$, and let $\mathcal{S}^r$ is a sequence of cuts selected until this point from the sequence $\mathcal{S}$.
2. We define $\mathcal{S}^{r+1} \leftarrow \mathcal{S}^r \circ \langle S_{r+1} \rangle$ if S_{r+1} crosses U^r, else we define $\mathcal{S}^{r+1} \leftarrow \mathcal{S}^r$. Either way, $U^{r+1} = U \setminus \bigcup_{S \in \mathcal{S}^{r+1}} S$.

Note that U^r and $\mathcal{S}^r$ are both functions of $(U, \mathcal{S})$. Call the cuts in $\mathcal{S}^{2^k}$ to be the *valid* cuts for set U. Given the same sequence of cuts $\mathcal{S}$ we may get different subsequences for each subset U of V. So it is not necessarily true that $(U \setminus T)^r = U^r \setminus T$, because the set of valid cuts can differ when considering point sets U and $U \setminus T$. But it turns out that we can still relate $(U \setminus T)^r$ and $U^r \setminus T$ in some settings.

Lemma 6.8.1. *Given sequence $\mathcal{S}$ and index $0 \leq r \leq 2^k$ such that $U^r \setminus T \neq \emptyset$, we have $(U \setminus T)^r = U^r \setminus T$.*

Proof of Lemma 6.8.1. We prove this by induction on r. For $r = 0$, we know that $U^0 \setminus T = U \setminus T = U^0 \setminus T$. Suppose the claim holds for $r = t$, then we want to show it holds for $r = t + 1$. Suppose $U^{t+1} \setminus T \neq \emptyset$, then since $U^{t+1} \subseteq U^\top$ we have $U^t \setminus T \neq \emptyset$, and by the inductive hypothesis we get that $(U \setminus T)^\top = U^\top \setminus T$. In particular, we get that $(U \setminus T)^\top \subseteq U^\top$. Hence if the new cut S_{t+1} crosses $(U \setminus T)^\top$, then it also crosses $U^\top$, and therefore

$$(U \setminus T)^{t+1} = (U \setminus T)^t \setminus S_{t+1} = (U^\top \setminus T) \setminus S_{t+1} = (U^{t+1} \setminus T).$$

So suppose the new cut S_{t+1} does not cross $(U \setminus T)^\top$, and thus $(U \setminus T)^{t+1} = (U \setminus T)^t$. There are two cases: either $(U \setminus T)^\top \subseteq S_{t+1}$ or $(U \setminus T)^\top \cap S_{t+1} = \emptyset$. In the first case, if S_{t+1} crosses $U^\top$, then $U^{t+1} \setminus T = (U^\top \setminus S_{t+1}) \setminus T = \emptyset$, and hence there is nothing to prove. Else if S_{t+1} does not cross $U^\top$, then $U^{t+1} = U^\top$ and also $(U \setminus T)^{t+1} = (U \setminus T)^t$, so we are done using the inductive hypothesis.

In the second case, $(U \setminus T)^t \cap S_{t+1} = (U^t \setminus T) \cap S_{t+1} = \emptyset$, so we get that $(U^\top \cap S_{t+1}) \subseteq T$. This means that $U^{t+1} \setminus T = U^t \setminus T$ regardless of whether S_{t+1} crosses $U^\top$. Since $U^t \setminus T = (U \setminus T)^t = (U \setminus T)^{t+1}$, we are done. □

As discussed above, $|U^{2^k}| = 1$, and we define this unique point $p \in U^{2^k}$ to be the "last" point in U, and we call this event "p is last in U".

Lemma 6.2.4 (Monotonicity)**.** *For any sets T, V such that $T \subseteq V$, and any point $\boldsymbol{p} \in V \setminus T$, we have*

$$\text{“}\boldsymbol{p} \text{ is last in } V\text{”} \Rightarrow \text{“}\boldsymbol{p} \text{ is last in } V \setminus T\text{”}.$$

Proof. Using the definition of the event "p is last in U", we know that $U^{2^k} = \{p\}$ and since $p \notin T$, we have $U^{2^k} \setminus T \neq \emptyset$. Using Lemma 6.8.1, we can say that $(U \setminus T)^{2^k} = \{p\}$. □

6.8.2 Proofs from Section 6.3

Lemma 6.3.2 (Hitting Set Lemma)**.** *For large enough k, there exist set systems $([k], \mathcal{S})$ with k sets of size s each, such that the minimum hitting set satisfies $h(s-2-o(1))/k \geq \ln k - O(\ln \ln k)$.*

Proof of Lemma 6.3.2. For some parameter $p \in (0, 1/2)$, consider k independently chosen sets, each obtained by adding in each element of $[k]$ independently with probability p. The expected size of each such set is $\bar{s} := pk$; moreover, each element of $[k]$ should hit a p fraction of the sets, so hitting all the k sets should require $\bar{h} := \ln_{1-p}(1/k) \approx (1/p)\ln k$ sets, giving $\bar{s}\bar{h}/k = \ln k$. We now show that with non-zero probability, there does exist a set system with parameters close to these.

Define $\mathcal{B}_1$ to be the bad event that some set S_i has size smaller than $s := (1-\varepsilon)pk$, and $\mathcal{B}_2$ to be the bad event that the hitting set has size at most some parameter h. We now show that for suitable choices of ε and h, we have $\Pr[\mathcal{B}_1] < 1/2$ and $\Pr[\mathcal{B}_2] \leq 1/2$, which completes the proof. We consider the event $\mathcal{B}_2$ first: a union bound shows that

$$\Pr[\mathcal{B}_2] \leq \sum_{H:|H|=h} \Pr\left[\forall i \in [k] : S_i \cap H \neq \emptyset\right] = \binom{k}{h}(1-(1-p)^h)^k \leq \frac{(2k)^h}{2} \cdot e^{-k(1-p)^h}.$$

Setting this upper bound to equal $1/2$, we get

$$\ln h - h\ln(1-p) = \ln\left(\frac{k}{\ln 2k}\right). \tag{6.40}$$

We now use that $-p - p^2 \leq \ln(1-p) \leq -p$ for $p \in [0, 1/2]$ to get

$$\ln h + hp \leq \ln\left(\frac{k}{\ln 2k}\right) \leq \ln h + hp(1+p). \tag{6.41}$$

Since $h \geq 1$, the left-most inequality of (6.41) gives $h \leq (1/p)\ln(\frac{k}{\ln 2k})$. However, we want a lower bound on h, so we substitute this into the right-most inequality of (6.41) to get

$$\frac{1}{1+p} \cdot \ln \underbrace{\left(\frac{k}{\ln 2k} - \frac{1}{p}\ln\left(\frac{k}{\ln 2k}\right)\right)}_{(\star)} \leq p\,h = \frac{s\,h}{(1-\varepsilon)k}. \tag{6.42}$$

We can now set $p := \frac{2\ln 2k}{k^{1/3}}$ (which ensures that the second term in $(\star)$ is at most half the first for a large enough k) and get

$$sh/k \geq \frac{1-\varepsilon}{1+p} \cdot \ln\left(\frac{k}{2\ln 2k}\right).$$

Now setting $\varepsilon := 1/k^{1/3}$ and using a Chernoff bound and a union bound,

$$\Pr[\mathcal{B}_1] < k \cdot e^{-\varepsilon^2 pk/2} = \frac{1}{2}.$$

Taking a union bound over the two bad events, we get that with non-zero probability our sets in $\mathcal{S}$ are of size $\approx k^{2/3}\ln k$, the hitting set is of size $\approx k^{1/3}$, and $hs/k \geq (1 - \frac{2\ln 2k}{k^{2/3}})(\ln k - O(\ln\ln k))$. □

6.8.3 Proofs from Section 6.4

Proof of the Stretch-vs.-Separation Claim 6.4.9

We prove a lemma about point sets that immediately implies Claim 6.4.9. Consider a set $S \subseteq \mathbb{R}^d$ of points, and focus on $p, q \in S$. Consider some dimension i, and consider the projection of the points onto that dimension (as in the figure). Let $a_1, \ldots, a_\ell$ be the lengths of intervals into which the line segment joining p_i and q_i is partitioned by projections of other points in S onto the i^{th} dimension. Any cut (i, θ) intersecting the j^{th} interval splits the centers into two groups with at least $\min(j, \ell+1-j)$ centers on either side.

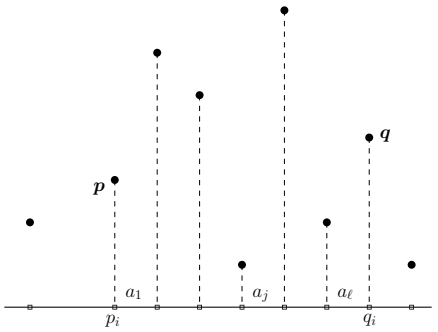


Figure 6.5: projection of points onto an axis

Define the *stretch* s_i between p and q in the i^{th} dimension, and the *expected separation* sep_i after choosing a random cut that cuts the j^{th} interval with probability proportional to a_j^2 as

$$s_i := \frac{\left(\sum_{j\in[\ell]} a_j\right)^2}{\sum_{j\in[\ell]} a_j^2} \qquad \text{and} \qquad \text{sep}_i := \frac{\sum_{j\in[\ell]} a_j^2 \cdot \min(j, \ell+1-j)}{\sum_{i\in[\ell]} a_j^2}.$$

Lemma 6.8.2. *$sep_i \geq \frac{s_i}{8(1+\ln((2|S|)/s_i))}$.*

Before we prove this, let us generalize this to higher dimensions:

Corollary 6.8.3. *Consider a set $S \subseteq \mathbb{R}^d$ and two points $p, q \in S$ having stretch s. If we choose a threshold cut (i, θ) from the distribution $D_2(S)$ and condition on separating p, q, the expected number of points in each side of the cut decreases by at least $\frac{s}{8(1+\ln((2|S|)/s))}$.*

Proof. The stretch between p, q, and the expected separation conditioned on separating the two centers is

$$\overline{s} := \frac{\sum_{i\in[d]} \left(\sum_{j\in[\ell_i]} a_{i,j}\right)^2}{\sum_{i\in[d]} \sum_{j\in[\ell_i]} a_{i,j}^2} \qquad \text{and} \qquad \overline{sep} := \frac{\sum_{i\in[d]} \left(\sum_{j\in[\ell_i]} a_{i,j}^2 \cdot \min(j, \ell_i+1-j)\right)}{\sum_{i\in[d]} \sum_{j\in[\ell_i]} a_{i,j}^2},$$

where $a_{i,j}$ is the width in the partition defined above along dimension i. Define a random variable $I \in [d]$ on the dimensions, that takes on value i with probability

$$\frac{\sum_{j\in[\ell_i]} a_{i,j}^2}{\sum_{i\in[d]} \sum_{j\in[\ell_i]} a_{i,j}^2}.$$

Then we have $\overline{s} = \mathbb{E}_I[s_I]$ and $\overline{\text{sep}} = \mathbb{E}_I[\text{sep}_I]$. Finally, the function $h(x) = \frac{x}{1+\ln(\alpha/x))}$ being

convex for $0 \le x \le \alpha$, we can use Jensen's inequality to get

$$\overline{\text{sep}} = \mathbb{E}_I[\text{sep}_I] \overset{\text{(Lem.6.8.2)}}{\geq} \mathbb{E}_I\left[\frac{s_I}{8\left(1+\ln\left(\frac{2|S|}{s_I}\right)\right)}\right] \geq \frac{\overline{s}}{8\left(1+\ln\left(\frac{|S|}{\overline{s}}\right)\right)}. \qquad \square$$

Finally, translating to the language of §6.4 shows that Claim 6.4.9 is just a reformulation of Corollary 6.8.3. So it suffices to prove Lemma 6.8.2, which we do next.

Proof of Lemma 6.8.2. Let us look at the following constrained minimization problem

$$\begin{aligned} \min \text{sep}_i = \textstyle\sum_j a_j^2 \cdot \min(j, \ell+1-j) \\ s.t. \quad \textstyle\sum_j a_j = \sqrt{s_i} \\ \textstyle\sum_j a_j^2 = 1. \end{aligned} \tag{6.43}$$

The Lagrangian dual of the above primal program is

$$\mathcal{L}(a,\lambda,\gamma) = \sum_j a_j^2 \cdot \min(j,\ell+1-j) - 2\lambda\Big(\sum_j a_j - \sqrt{s_i}\Big) + \gamma\Big(\sum_j a_j^2 - 1\Big),$$

and setting the gradient to zero means the minima for a occur when $a_j = \frac{\lambda}{\gamma+\min(j,\ell+1-j)}$. (We assume $\gamma \ge 0$.) Substituting and simplifying gives

$$2\lambda\sqrt{s_i} - \sum_j \frac{\lambda^2}{\gamma+\min(j,\ell+1-j)} - \gamma.$$

We can maximize over λ which happens when $\sum_j \frac{\lambda}{\gamma+\min(j,\ell+1-j)} = \sqrt{s_i}$; substituting gives

$$\frac{s_i}{\sum_j \frac{1}{\gamma+\min(j,\ell+1-j)}} - \gamma \ge \frac{s_i}{2\ln(1+\frac{\ell+1}{2\gamma})} - \gamma \ge \frac{1}{2}\Big(\frac{s_i}{\ln(1+\frac{|S|}{2\gamma})} - 2\gamma\Big).$$

It remains to choose γ. For convenience, we set the above expression to γ; this means

$$\frac{s_i}{\ln(1+\frac{|S|}{2\gamma})} = 4\gamma \qquad \Longleftrightarrow \qquad \frac{2|S|}{s_i} = \frac{|S|/2\gamma}{\ln(1+\frac{|S|}{2\gamma})}.$$

Using Observation 6.8.4 below, we get $\gamma \ge \frac{s_i}{8(1+\ln((2|S|)/s_i))}$. $\square$

Observation 6.8.4. For any $x \ge 0$ and $y \ge 1$ such that $y = \frac{x}{\ln(1+x)}$, we have $x \le 2y(1+\ln y)$.

Proof. Since $x/\ln(1+x)$ is an increasing function, it is sufficient to prove that

$$y \le \frac{2y(1+\ln y)}{\ln(1+2y(1+\ln y))}.$$

Finally, taking derivatives shows $\ln(1+2y(1+\ln y)) \le 2(1+\ln y)$ for $y \ge 1$. $\square$

6.8.4 Proofs from Section 6.5

Proof of Lemma 6.5.2. For a point $\boldsymbol{p} \in U$, define

$$z_E(t) := \begin{cases} z_E \cdot t & \boldsymbol{p} \in E \\ z_E & \boldsymbol{p} \notin E. \end{cases} \tag{6.44}$$

Let $f_t(U)$ and $\ell_t(\boldsymbol{p})$ be the function value of points in U and norm of $\boldsymbol{p}$ at time t when the embedding is changing according to the trajectory given by Equation (6.44).

We first claim that $\ell_0(\boldsymbol{p}) = 0$ and $f_0(U) = 0$. Indeed, We know that $\ell_t(\boldsymbol{p}) = \sum_{S:\boldsymbol{p}\in S} z_S(t) = \sum_{S:\boldsymbol{p}\in S} z_S \cdot t = t \cdot \ell(\boldsymbol{p})$, which implies that $\ell_0(\boldsymbol{p}) = 0$. The second part is proved by induction on $|U|$: If $|U| = 1$, then $f_0(U) = \ell_0(\boldsymbol{p}) = 0$. If $|U| \geq 2$ and $z_E(0) = 0$ for all $E \in \mathcal{C}_U$, then $f_0(U) = 0$ by definition. Otherwise, using Equation (6.17), we can write

$$f_0(U) = \frac{\sum_{E\in\mathcal{C}_U} z_E(0) \cdot f_0(U\backslash U_E)}{\sum_{E\in\mathcal{C}_U} z_E(0)}.$$

In the numerator of $f_0(U)$, either $z_E(0) = 0$ when $\boldsymbol{p} \in E$, or $f_0(U\backslash U_E) = 0$ when $\boldsymbol{p} \in U\backslash U_E$ by inductive hypothesis which concludes the proof.

Now using the chain rule and the assumed bound on the derivative,

$$\begin{aligned} \frac{\mathrm{d}f_t(U)}{\mathrm{d}t} = \sum_{E\subseteq[k]} \frac{\partial f(U)}{\partial z_E}\Big|_{\boldsymbol{z}=\boldsymbol{z}(t)} \cdot \frac{\mathrm{d}z_E(t)}{\mathrm{d}t} &= \sum_{E:\boldsymbol{p}\in E} z_E \cdot \frac{\partial f(U)}{\partial z_E}\Big|_{\boldsymbol{z}=\boldsymbol{z}(t)} \\ &\leq \sum_{E:\boldsymbol{p}\in E} z_E \cdot \beta_{k-1} = \beta_{k-1} \cdot \ell(\boldsymbol{p}). \end{aligned}$$

Integrating gives $f_1(U) - f_0(U) = f(U) - f_0(U) = f(U) \leq \beta_{k-1} \cdot \ell(\boldsymbol{p})$. □

Proof of Lemma 6.5.3. The proof is by induction on $|S|$. The base case is when $|S| = 1$. If $S = \{r\}$, then we have $\frac{\partial f(S)}{\partial z_T} = \frac{\partial \ell(r)}{\partial z_T}$. We know that $\ell(r)$ can be written as

$$\ell(r) = \sum_{E\ni r} z_E. \tag{6.45}$$

The derivative $\frac{\partial \ell(r)}{\partial z_T}$ is equal to 1 if z_T appears as a term in the expansion of $\ell(r)$ as in Equation (6.45) and is equal to 0 otherwise. In the case of statement (i), since $T \in \mathcal{C}_S^\ell$, we have $S \subseteq T \implies r \in T$ this concludes the base case for statement (i). Similarly, if statement (ii) holds, z_T does not appear as a term in the expansion of $\ell(r)$ and hence the partial derivative of $f(S) = \ell(\mu_r)$ with respect to z_T is zero.

For $|S| \geq 2$, we have $\mathbb{1}[T \in \mathcal{C}_S] = 0$ in both the cases of statements (i) and (ii). Using this in Equation (6.18) gives

$$\frac{\partial f(S)}{\partial z_T} = \frac{\sum_{E\in\mathcal{C}_S} z_E \cdot \frac{\partial f(S\backslash E)}{\partial z_T}}{\sum_{E\in\mathcal{C}_S} z_E}. \tag{6.46}$$

For the inductive step, since $S \subseteq T$ in statement (i), we have $(S \backslash E) \subseteq T$. By the inductive hypothesis, the recursive derivative terms $\frac{\partial f(S \backslash E)}{\partial z_T}$ are equal to 1. Using Equation (6.46), we are done. The inductive step for the case of statement (ii) follows similarly where all terms $\frac{\partial f(S \backslash E)}{\partial z_T}$ are equal to 0. Finally, using the fact that $f(S)$ is always non-negative for any z, and the fact that $\frac{\partial f(S)}{\partial z_E} = 1$ for any $S \subseteq E$, we have $f(S) \geq \sum_{E \supseteq S} z_E$. □

Proof of Lemma 6.5.5. The proof is by induction on $|S|$. Observe that for $T \notin \mathcal{C}_S$, either $T \supseteq S$ or $T \cap S = \emptyset$, and then $\frac{\widehat{\partial} f(S)}{\widehat{\partial} z_T} = \frac{\partial f(S)}{\partial z_T}$ by Definition 6.5.4 and Lemma 6.5.3. The fact that the pseudo-derivative is non-negative in this case is immediate from the Definition 6.5.4. For $|S| = 1$, since $T \notin \mathcal{C}_S$, we are done. If $|S| \geq 2$ and $T \in \mathcal{C}_S$, the inductive hypothesis implies that $\frac{\widehat{\partial} f(S \backslash E)}{\widehat{\partial} z_T} \geq \max\left(\frac{\partial f(S \backslash E)}{\partial z_T}, 0\right)$. It remains to prove that

$$\max\left(f(S \setminus T) - f(S), 0\right) \leq f(S \setminus T) - \sum_{E \supseteq S} z_E.$$

Using Lemma 6.5.3, we get $f(S \setminus T) - f(S) \leq f(S \setminus T) - \sum_{E \supseteq S} z_E$. The other inequality $\sum_{E \supseteq S} z_E \leq f(S \setminus T)$ follows from the below argument, again using statement (iii) of Lemma 6.5.3:

$$f(S \setminus T) \geq \sum_{E \supseteq S \backslash T} z_E \geq \sum_{E \supseteq S} z_E$$

This concludes the proof of the lemma. □

Part IV

Robustness

Chapter 7

Combinatorial Optimization using Comparison Oracles

7.1 Introduction

Consider the following optimization problem: we are given a *ground set* U of n elements, a family $\mathcal{F} \subseteq 2^U$ of *feasible subsets* of the ground set U, and an unknown objective function $f : 2^U \to \mathbb{R}_+$. At each step, we can compare any two feasible sets $S, T \in \mathcal{F}$, and learn the sign of $f(S) - f(T)$, which could be negative, zero, or positive. Clearly, we can optimize the function f over the family $\mathcal{F}$ using brute-force, in $|\mathcal{F}| - 1$ queries, which can be exponential in $|U|$. But, can we do so with only polynomially many queries in $|U|$, at least for linear functions and structured classes of problems?

As a concrete representative, and a problem of central importance to algorithm design, consider the problem of finding a minimum-weight-cut ("min-cut") in an unknown undirected graph $G = (V, E, w : E \to \mathbb{R}_+)$. Initially we only know the vertices V, but nothing about the edges (or their weights). At each step, we can query a (pairwise) comparison oracle by presenting two non-trivial subsets of vertices $A, B \subseteq V$ to it: in response, we learn which of the two cuts ∂A and ∂B has smaller weight (or if they have the same weight). This is a special case of the abstract problem above, where U is the set of vertices in a graph, $\mathcal{F} = \{S \subseteq U : S \notin \{\emptyset, U\}\}$ is the set of all non-trivial cuts, and $f(S) = \sum_{e \in \partial S} w_e$ is the total weight of edges in the cut induced by S. Can we find the min-cut using a small number of these cut-comparison queries?

Apart from min-cut being a fundamental problem, and a good testbed for algorithmic techniques, a second motivation for considering it is that computing the min-cut without explicit access to the graph has recently seen intensive activity. However, this has been in the (more informative) *value oracle* model, where the algorithm can query the value of $f(S)$ for any set $S \in \mathcal{F}$ (see, e.g., [11, 143, 162]). By now, we know algorithms that compute the min-cut efficiently with only linearly many value queries [11]. Observe that we can implement a comparison query using two value queries, so these value queries are at least (half) as informative as comparison queries.

This begs the question: *how many comparison queries do we need to compute the min-cut?* Being able to compare two structures is perhaps the minimal amount of power we can give to an algorithm looking for the min-cost structure: what is the power and what are the limitations of the comparison model for optimization?

Beyond the algorithmic interest in this model, the importance of comparisons (or *ordinal queries*) as opposed to numerical scores (*cardinal queries*) is well-recognized in several disciplines.

In economics, ordinal utility was introduced in the classic work of Pareto [153]. More recently, ordinal utility has played an important role in recommendation systems, where it is often crowd-sourced using pairwise comparisons [46], since different users may have different scales and may also find it difficult to score items, but they may find it easier to compare two items head-to-head. Ordinal preferences have been used to model consumer behavior [22]. In statistics, a well-studied problem is that of inference from experiments involving pairwise comparisons [30]. In social sciences as well, a standard model of assessment is via pairwise comparisons [104].

7.1.1 Our Results and Techniques

In this work, we explore the algorithmic power and limitations of this comparison-based model for several basic combinatorial problems. Our primary focus is the *linear optimization* setting: where there is an unknown weight w_e for each element $e \in U$, and the cost of a feasible set $S \in \mathcal{F}$ is linear, namely $f(S) := \sum_{e \in S} w_e$. While some natural problems such as min-cut do not fall directly into this form—since the objective is not linear over the vertex set—they still admit a representation that fits within our framework. In the min-cut problem, the objective is $f(S) = w(\partial(S))$, which is linear over the set of cut edges $\partial(S)$, even though it is not linear in the vertex set. This viewpoint still allows us to apply our techniques. We begin by studying problems where the feasible family has rich structure—not only graph cuts discussed above, but also matroid bases, matroid intersections, or paths in a graph. Despite the rich variety in these optimization tasks, we show that they remain efficiently solvable using only comparison queries.

Graph Cuts. We consider the setting of (unweighted) simple graphs, i.e., all edge weights are 1. We extend the work of [162], who considered the case of unweighted simple graphs in the more informative *value oracle* model, and match their results using just a *comparison oracle* (we use m, n to denote the number of graph edges and vertices respectively):

Theorem 1.6.1 (Minimum cut)**.** *There is a randomized algorithm that computes the exact minimum cut of a simple graph G with high probability using $\widetilde{O}(n)$ cut comparison queries and $\widetilde{O}(n^2)$ time.*

In addition to finding the minimum cut, we show that it is possible to recover the entire edge set of G deterministically using cut comparisons except in some degenerate cases: when $G \in \{K_2, \bar{K}_2, K_3, \bar{K}_3\}$.[1]

[1] K_2 is a single edge and K_3 a triangle. Their complement graphs $\bar{K}_2, \bar{K}_3$ are respectively empty graphs on 2 and 3 vertices.

Theorem 1.6.2 (Graph recovery)**.** *A simple unweighted graph $G \notin \{K_2, \bar{K}_2, K_3, \bar{K}_3\}$ can be recovered using $O(\min\{(m+n)\log n, n^2\})$ cut comparison queries and in $\widetilde{O}(n^2)$ time.*

Finally, we show that cut comparisons suffice to construct sparsifiers for a broad class of graph problems called *heavy subgraph* problems (which include *densest subgraph* and *max-cut*) by combining our edge-sampling techniques with the results of [72]; see Section 7.2.4 for details.

We note that in these problems, there is a qualitative difference between the value oracle and comparison oracle settings. For example, in the value oracle model, graph reconstruction is immediate: evaluating the expression $1/2(f(\{u\}) + f(\{v\}) - f(\{u,v\}))$ immediately reveals the presence/absence of edge (u,v). In contrast, reconstructing a graph is not straightforward with cut comparisons. For instance, all non-trivial cuts in K_3 (i.e., the triangle graph) have the same value, and this is also the case in its complement graph $\bar{K}_3$. Thus, these graphs cannot be distinguished by comparison queries alone. Similarly, $K_2, \bar{K}_2$ have only a single non-trivial cut, rendering the comparison model useless. More generally, for weighted graphs, there is a sharp distinction between the value oracle and comparison oracle models. In the former, the weight of every edge (u,v) can be recovered using the expression above $1/2(f(\{u\})+f(\{v\})-f(\{u,v\}))$. In contrast, weighted graphs cannot be recovered at all in the comparison oracle model, even up to scaling (see Section 7.2.5). Hence, it comes as a surprise that we can recover simple graphs completely (except in the degenerate cases $K_2, \bar{K}_2, K_3, \bar{K}_3$) using comparison queries. The main idea is simple in hindsight: if we consider a vertex v and set $S \not\ni v$, then comparing $f(S+v)$ and $f(S)$ tells us whether more than half of v's incident edges go into S. We build on this observation to show how to identify edges incident to v: consider a sequence of nested sets $\emptyset \neq S_0 \subseteq S_1 \subseteq \ldots \subseteq S_{n-1} \neq V \setminus \{v\}$, and ask the question above setting $S = S_i$ for each one. If we find a "tipping point"—a query in this sequence where the sign of $f(S+v) - f(S)$ changes—we have found an edge. We can further refine the process to make edge recovery more efficient, and to perform efficient sampling on G. These primitives can then be used in existing algorithmic frameworks for min-cut to get a randomized algorithm for min-cut using $\widetilde{O}(n)$ comparison queries. The details of our min-cut algorithm appear in Section 7.2.

We note that our techniques don't extend to min-cut in weighted graphs. It appears that weighted graphs pose some unique challenges that will need new ideas beyond those in this chapter. For instance, the ideas sketched above also give us *non-adaptive* algorithms for min-cut and graph recovery in simple graphs that use $\mathrm{poly}(n)$ comparison queries. In contrast, we show that on weighted graphs, any algorithm using non-adaptive comparison queries has exponential query complexity for these tasks (see Claim 7.2.7). Furthermore, natural primitives such as finding the maximum weight edge incident to a vertex, or sampling a random edge incident to a vertex, are provably not implementable using comparison queries alone in weighted graphs (see Section 7.2.5 for further discussions about these bottlenecks). We leave finding the min-cut in a weighted graph using $\mathrm{poly}(n)$ comparison queries efficiently, or showing that this is impossible, as an interesting open question.

Matroid Bases, Matroid Intersections, and Paths. Next, we consider other basic combinatorial objects in graphs: *spanning trees*, (bipartite) *matchings*, and s-t paths. Going beyond graphs, we also consider natural extensions of the first two problems to *matroid bases* and *matroid inter-*

sections. In all these problems, we have an unknown weight function w_e on the edges/elements, and the oracle compares $f(A), f(B)$ for any two feasible sets A, B, where $f(S) = \sum_{e \in S} w_e$.

First, we consider the problem of finding a minimum-weight basis of a matroid, which captures the minimum spanning tree problem as a special case:

Theorem 1.6.3 (Matroid Bases)**.** *There is an algorithm outputs the minimum-weight basis of a matroid on n elements using $O(n \log n)$ comparison queries in $\widetilde{O}(n^2)$ time.*

Recall that knowing the order of element weights is enough to optimize over matroids (using the greedy algorithm), but how can we compare element weights? The first observation is that if two elements share a circuit, then we can find two bases which differ on exactly these two elements, and hence compare them. But, what about elements that do not share a circuit? We show that one can decompose any matroid so that we never need to compare such elements.

Next, we consider the problem of finding a minimum-weight independent set in the intersection of two matroids defined on the same ground set:

Theorem 1.6.4 (Matroid Intersection)**.** *Let $\mathcal{M}_1 = (U, \mathcal{I}_1)$ and $\mathcal{M}_2 = (U, \mathcal{I}_2)$ be two matroids defined on a ground set U containing n elements. Then, there is an algorithm that outputs the minimum-weight set that is in both $\mathcal{I}_1, \mathcal{I}_2$ using $O(n^4)$ comparison queries in $O(n^4)$ time.*

Note that this theorem can solve maximum weight bipartite matching as a special case by first changing all edge weights from w_e to $-w_e$ and then solving minimum weight bipartite matching for these new weights.

For matroid intersection (and its special case, bipartite matchings), sorting the element weights is neither doable, nor sufficient for the problem. Instead, we show how to implement the shortest augmenting path algorithm (and its natural extension for matroid intersection) using only comparisons between matchings (i.e., common independent sets). The details are in Section 7.3.2.

In a similar vein to the previous result, we show that the Bellman-Ford algorithm for shortest s-t paths can be implemented using comparisons between s-t walks to obtain the following theorem (details in Section 7.3.3):

Theorem 1.6.5 (s-t walks)**.** *There is an algorithm that finds the minimum-length s-t path in a graph G (or a negative cycle, if one exists) using $O(n^3)$ s-t walk comparisons and $O(n^3)$ time.*

General Linear Optimization

Given these positive results, we ask: *for which families $\mathcal{F} \subseteq 2^U$ can we solve $\arg\min_{S \in \mathcal{F}} \sum_{e \in S} w_e$ for some unknown weight function $w : U \to \mathbb{R}$, using only $\mathrm{poly}(|U|)$ comparison queries?* By adapting the powerful active classification framework of [113] to the optimization setting, we establish a surprisingly general *positive* result for linear optimization: *we can optimize over all Boolean families with a small number of queries!*

Theorem 1.6.6 (Boolean Linear Optimization)**.** *For any family $\mathcal{F} \subseteq 2^U$ and unknown weight function $w : U \to \mathbb{R}$, we can solve $\arg\min_{S \in \mathcal{F}} \sum_{e \in S} w_e$ using $O(n \log n \cdot \log |\mathcal{F}|) = \widetilde{O}(n^2)$ comparison queries, where $n = |U|$.*

We find Theorem 1.6.6 remarkable: it shows that the number of comparison queries required to find the optimal solution is only $\widetilde{O}(n^2)$, *regardless of the complexity of* the set system $\mathcal{F}$. Indeed, we could consider $\mathcal{F}$ to represent set covers, independent sets, cliques, or other NP-hard problems for which finding the optimal set is believed to be computationally hard—nonetheless, the number of comparisons required is still nearly quadratic! Indeed, while the query complexity of the procedure in Theorem 1.6.6 is polynomial, its running time could be exponential (under standard complexity-theoretic assumptions). This shows a large gap between the information complexity and the computational complexity in this model.

To give intuition about the results, we also present a direct, albeit weaker, proof that shows a qualitatively similar result for settings with bounded integer weights.

Theorem 7.1.1 (Boolean Linear Optimization: Bounded Weights)**.** *For any family $\mathcal{F} \subseteq 2^U$ and unknown integer weight function satisfying $|w_e| \leq W$, we can sort all sets $S \in \mathcal{F}$ by their corresponding weight $\sum_{e \in S} w_e$ using $O(nW \log nW)$ comparison queries, where $|U| = n$.*

The main observation is that solutions can have at most $O(nW)$ distinct weights. The key insight is that the difference vector between any two solutions with the same weight is orthogonal to the true weight vector w^*. Our algorithm builds a single vector subspace from these difference vectors. A new solution's weight can often be inferred without comparisons by checking if its difference from a known solution lies in this shared subspace. Queries are only needed when discovering a new weight value or when a solution expands the dimension of this subspace (details in Section 7.4.1). While Theorem 7.1.1 relies on weights being integral and bounded, it gives a sense of how one can use the structure of Boolean linear optimization in these results. The main difference between Theorems 1.6.6 and 7.1.1 is that the first result uses *conic* spans while the second result uses only linear spans.

The framework of [113] allows us to also optimize over arbitrary point sets in $\mathbb{R}^d$, with the number of queries being related to their "conic dimension":

Theorem 1.6.7 (General Linear Optimization)**.** *There is an algorithm that, for any point set $\mathcal{P} \subseteq \mathbb{R}^d$ with conic dimension k and unknown weights $w \in \mathbb{R}^d$, returns the minimizer $x^* = \arg\min_{x \in \mathcal{P}} \langle w, x \rangle$ using at most $O(k \log k \log |\mathcal{P}|)$ comparisons.*

This result implies Theorem 1.6.6 using the fact that the conic dimension of the Boolean hypercube is $O(d \log d)$; moreover, it can be used to establish upper bounds on the query complexity for exact optimization when the point sets have bounded representation size, and for approximate optimization when they have bounded norm. We discuss these results in Section 7.4. We also note that some condition (like the bound on conic dimension) on the point set $\mathcal{P}$ is required in the above theorem: without any condition, [113, Theorem 4.8] shows a lower bound of $\Omega(|\mathcal{P}|)$ for general point sets in $\mathbb{R}^3$.

Interestingly, for the problem of *maximum cut* in a graph $G = (V, E)$, this result implies an upper bound of $\widetilde{O}(|V|^3)$ queries; moreover, our results can be made deterministic using the techniques of [114, Section 5]. The work of [154] shows a lower bound of $\Omega(|V|^2)$ queries in the more informative *value query* model for this problem; so, our result brings the gap between upper and lower bounds on the query complexity down to $O(|V|)$ for the max-cut problem.

In summary, these results highlight the surprising power and generality of comparison queries for combinatorial optimization, particular with linear objective functions. We hope that our work will lead to further exploration of this largely uncharted landscape in the future.

Organization. The rest of the chapter roughly follows the outline above: we present our results for min-cuts in Section 7.2, and our results for other combinatorial problems—matroids, matroid intersections, and shortest paths—in Section 7.3. We give our results for general and Boolean linear optimization in Section 7.4, and end the chapter with some open problems in Section 7.5. To maintain a smooth flow of ideas, we defer many technical proofs and digressions to the appendix.

7.1.2 Related Work

The min-cut problem has seen a lot of work in the *value query* model, where the algorithm accesses the input graph only by querying the cardinality/weight of individual cuts. The model was proposed by [162] who gave an algorithm using $\widetilde{O}(n)$ queries in simple graphs; this was extended by [11, 143] to the case of weighted graphs. These algorithms are randomized; the best deterministic bound is $O(n^2/\log n)$ [91] which allows full reconstruction of the input graph. There are also lower bounds: [90] define the *cut dimension* and use it to give a lower bound of $3n/2 - 2$ queries for weighted graphs; [126] improve the lower bound to $2n - 2$, which remains the best deterministic lower bound for this problem. The best randomized lower bound for the problem is $\Omega(n \log\log n/\log n)$ queries [12, 156].

The study of value oracles is natural for general submodular function minimization, given their natural representation is typically exponential sized.

Comparison oracles have been also studied for submodular functions and beyond: [15] show that for any submodular function f, one can use $\widetilde{O}(n/\varepsilon^3)$ exact comparison queries to the function f construct an approximate comparison oracle $\hat{f}$ that returns $\hat{f}(a) > \hat{f}(b)$ for any a, b if $f(a) > \sqrt{n} \cdot f(b)$. [97] studied a comparison model for accessing a metric space, where an oracle returns which of two locations j, k is closer to a fixed location i. They showed that the ordering of distances from a fixed location i can be obtained using $\log n$ queries w.h.p. under some expansion conditions on the metric; later work studied a similar access model for classification/regression in Euclidean space [98]. [188] studied binary classification under noisy labeling and access to a pairwise comparison oracle. In a similar vein, [113] studied active classification using label and pairwise comparison queries between the data points to recover a (linear) classification boundary. Indeed, this last result plays a central role in our work on general linear optimization. The techniques of [114] can be used to derandomize the query strategy for general linear optimization.

There is other work on graph reconstruction using cut value queries: see, e.g., [37, 48, 91]. Recently, there is work on solving other optimization problems (e.g., minimal spanning forests) using cut value queries; see, e.g., [14, 101, 129]. While we do not consider these kinds of "improper" optimization problems in this work, it seems an interesting direction for research. Further afield are more general query models, which allow for "independent set" queries, see, e.g., [129] for references.

7.2 Minimum Cut using Cut Comparisons

In this section, we focus on graphs and prove Theorems 1.6.1 and 1.6.2, which we restate for convenience.

Theorem 1.6.1 (Minimum cut)**.** *There is a randomized algorithm that computes the exact minimum cut of a simple graph G with high probability using $\widetilde{O}(n)$ cut comparison queries and $\widetilde{O}(n^2)$ time.*

Theorem 1.6.2 (Graph recovery)**.** *A simple unweighted graph $G \notin \{K_2, \bar{K}_2, K_3, \bar{K}_3\}$ can be recovered using $O(\min\{(m+n)\log n, n^2\})$ cut comparison queries and in $\widetilde{O}(n^2)$ time.*

In order to prove the two results, we draw on the following useful primitives that can be obtained using the cut comparison queries:

Theorem 7.2.1 (Structural Primitives)**.** *When $n \geq 4$, for any vertex $u \in U$:*

(i) Given a set of k vertices $A = \{v_1, \ldots, v_k\} \subseteq U \setminus \{u\}$, the neighbors of u in A can be determined using $k + O(\log n)$ queries and $\widetilde{O}(n+k)$ time.

(ii) Given an ordered subset $T = (v_1, \ldots, v_t) \subseteq U \setminus \{u\}$, either the neighbor $v_i \in N(u)$ of u with the smallest index $i \in [t]$ (or a certificate that $N(u) \cap T = \emptyset$) can be found using $O(\log n)$ queries and $\widetilde{O}(n)$ time.

Remark 7.2.2 (A Note on Runtimes)**.** We assume that the query input (S, T) to the comparison oracle is given as two bit vectors encoding the sets S and T respectively. The running time bounds are the total computation done to process the query information, where oracle calls are assumed to take unit time. The dominant cost is writing down the input bit vectors for queries, which takes $O(n)$ time per query. In addition, we separately account for standard algorithmic overhead, such as binary search or other data structure updates.

7.2.1 Graph Reconstruction

With the power of Theorem 7.2.1 behind us, the graph recovery problem is easily solved. Indeed, using Theorem 7.2.1(i), we can discover the neighborhood of node $u \in V$ using $n-1+O(\log n)$ queries and $\widetilde{O}(n)$ time; summing over all nodes gives us the graph G in at most $O(n^2)$ queries and $\widetilde{O}(n^2)$ time.

To get the better bound of $O(m \log n)$ for sparse graphs, we use Theorem 7.2.1(ii) to prove:

Lemma 7.2.3 (Edge Extraction)**.** *When $n \geq 4$, for any vertex u, a subset $T \subseteq U - u$ of vertices, and any integer k, we can extract $r = \min(k, \partial(u, T))$ edges from $\partial(u, T)$ using $O(r \log n)$ queries and $\widetilde{O}(n)$ time.*

Proof. Start with $T_0 := T$ and extract a neighbor $v_i \in N(u) \cap T_i$ (if any) from $\partial(u, T_i)$ using Theorem 7.2.1(ii); set $T_{i+1} := T_i - v_i$, and repeat. Each step requires $O(\log n)$ queries and $\widetilde{O}(n)$ time. Repeating this for $r = \min(k, \partial(u, T))$ neighbors gives a total of $O(r \log n)$ queries and

$\widetilde{O}(nr)$ time. However, there is an more efficient implementation that gives a running time of $\widetilde{O}(n)$. This version is provided in Section 7.6.1. □

Given this efficient edge-extraction claim, we now prove the claimed bounds for graph recovery.

Proof of Theorem 1.6.2. When $n \geq 4$, using Lemma 7.2.3, we can discover all the neighbors of u using $|\partial(u)| \cdot O(\log n)$ queries and $\widetilde{O}(n)$ time. Summing over all u gives a total of at most $O((m+n)\log n)$ queries and $\widetilde{O}(n^2)$ time. Hence, we start by running this procedure until the number of edges discovered is at least $n^2/\log n$. At this point, we switch to the algorithm using Theorem 7.2.1(i) to discover the remaining edges using at most $O(n^2)$ queries. This guarantees that we use at most $O(\min((m+n)\log n, n^2))$ queries and $\widetilde{O}(n^2)$ time.

When $n \leq 3$ and $G \notin \{K_2, \bar{K}_2, K_3, \bar{K}_3\}$, sorting the degrees of the vertices reveals the graph. □

7.2.2 Finding Minimum Cuts

We now turn to finding minimum cuts. We start off by observing that when $n \leq 4$, we can bruteforce the minimum cut and otherwise, use Theorem 1.6.2 to optimize over min-cuts: we reconstruct the graph using $O(\min((m+n)\log n, n^2))$ queries and $\widetilde{O}(n^2)$ time; we can then optimize over it in $\widetilde{O}(n^2)$ time. However, the number of cut queries incurred this way is quadratic and not near-linear, as promised in Theorem 1.6.1.

However, we can use ideas from [162] to do better. Suppose G' is a contracted graph (i.e., where some nodes in G have been contracted into each other), and $G'(p)$ is the "edge-percolation" graph obtained by sub-sampling each edge of G' independently with probability p. Distilling the results of [162] shows that if, for any contraction G' of G, we can obtain a sample from $G'(p)$ using $\alpha \cdot p \cdot |E(G')|$ queries, then we can compute the min-cut of G using $\alpha \cdot \widetilde{O}(n)$ queries and in $\widetilde{O}(n^2)$ expected runtime. (A proof appears in Section 7.6.1.)

To use this result, we show how to sample percolations of contractions with $\alpha = \widetilde{O}(1)$.

Lemma 7.2.4 (Contracted Graphs)**.** *When $n \geq 4$, for any contraction $G' = (U', E')$ of G, with its vertex set U' represented as a partition of the original vertex set U, we can sample a subgraph $G'(p)$ where each edge in G' is sampled independently with probability p using $|E'| \cdot O(p \log n) + \widetilde{O}(n)$ queries and $\widetilde{O}(n^2)$ time in expectation.*

The sampling can be done more generally for any induced subgraph $G'' = G'[S]$ for some $S \subseteq U'$ with $|E'(S)| \cdot O(p \log n) + \widetilde{O}(n)$ queries and $\widetilde{O}(n^2)$ time.

Proof. For any regular vertex u, if V_u is the super vertex that contains it, we will show how to sample edges from $\partial(u, U - V_u)$ with each edge sampled independently with bias $q \in [0,1]$. Call this random subgraph $\partial_{G'}(u,q)$. Given the primitive to sample $\partial_{G'}(u,q)$, taking the union $\cup_{u \in U} \partial_{G'}(u,q)$ of such samples for each vertex $u \in U$ with $q = 1 - \sqrt{1-p}$ gives $G'(p)$. Because, the probability of any edge $e \in E'$ to be selected is $1 - (1-q)^2 = p$ and the independence is trivial.

To sample $\partial_{G'}(u, q)$, let T be a random subset of $U - V_u$ with each element sampled independently with probability q. Reveal all the edges of u going into T using Lemma 7.2.3. This takes $|\delta(u, T)| \cdot O(\log n)$ queries which is $q \cdot |\delta(u, U - V_u)| \cdot O(\log n)$ in expectation and $\widetilde{O}(n)$ time. Summing over all $u \in U$ gives a total of $|E'| \cdot O(p \log n) + \widetilde{O}(n)$ queries and $\widetilde{O}(n^2)$ time in expectation. An edge $\{u, v\}$ for $v \in U \backslash V_u$ is sampled iff $v \in T$. This implies that each edge is sampled independently with probability q. Simply replacing U with S gives the extension to induced subgraphs. □

Combining Lemma 7.2.4 with the abstraction of [162] discussed above completes our algorithm for min-cuts (Theorem 1.6.1).

7.2.3 Proof of the Structure Primitives Theorem

Finally, we turn to the proof of the Structure Primitives result in Theorem 7.2.1. We will provide a proof sketch here for the query complexity bounds; see Section 7.6.1 for the full proof including the running time bounds.

Proof sketch of Theorem 7.2.1. Recall that we want to identify the edges coming out of u. The key observation is that, for any set S not containing u, the sign of $\partial(S + u) - \partial(S)$ tells us whether less, equal to, or more than than half of the edges incident to u go into S (if the sign is positive, zero, or negative).

Suppose we find a set S such that *just below* half of the edges from u go into S (so that expression has a positive sign)—such an S is called a *median set for* u. Now if we add v to the median set S, and the sign of $\partial(S + v + u) - \partial(S + v)$ changes, we know that v is a neighbor of u. In summary, once we have a median set S, we can find all the neighbors of u not in S.

How do we find a median set for u? Order all the vertices other than u (say this is $v_1, v_2, \ldots, v_{n-1}$, define S_i to be the first i elements. If u has at least one edge incident to it, the query $\partial(S_0 + u) - \partial(S_0)$ has a positive sign, whereas $\partial(S_{n-1} + u) - \partial(S_{n-1})$ has a negative sign (note that we use $S_0 = \emptyset$ and $S_{n-1} + u = U$ here for the sake of exposition, but such trivial cuts are not allowed in the actual comparison model. The full proof in Section 7.6.1 handles this correctly and with full rigor). So we can perform binary search to find a point where the sign changes, which can be used to find a median set in $\log_2 n$ comparisons. (In fact, we can find two disjoint median sets—a prefix and a suffix of this ordering—and use the two to find all the neighbors in some k-sized set with $k + O(\log n)$ queries.)

For part (ii), the argument builds on the use of such a median set S. Given an ordered set $T = \{v_1, \ldots, v_r\}$ where v_j is the first neighbor of u in the ordering, we first compute a median set S that is guaranteed to exclude the prefix $\{v_1, \ldots, v_j\}$. Then, we binary search over the chain

$$S \subseteq S \cup \{v_1\} \subseteq \cdots \subseteq S \cup (T \setminus S)$$

to locate the point at which the marginal changes sign. This transition occurs precisely at the set $S \cup \{v_1, \ldots, v_{j-1}\}$, thus identifying v_j as the first neighbor of u in T. □

7.2.4 Sparsifiers for Heavy Subgraph Problems

The work of [72] considers a broad class of graph optimization problems, which they call *heavy subgraph* problems; these include densest subgraph, densest bipartite subgraph, and d-max cut. For these problems, they show that uniformly sampling $\widetilde{O}(n/\varepsilon^2)$ edges without replacement from the underlying graph G produces a subgraph H such that running any α-approximation algorithm for the problem on H yields (after appropriate rescaling) an $(\alpha - \varepsilon)$-approximate solution for the original graph, with high probability. In other words, the sampled subgraph preserves the structure of the problem up to a small loss in accuracy, and approximate solutions on the sparsifier H translate to approximate solutions on the full graph G. We show how to sample a fixed number of edges from G uniformly using the following lemma:

Lemma 7.2.5 (Sampling Lemma)**.** *When $n \geq 4$, given cut comparison queries, for any $k = \widetilde{\Omega}(1)$, we can sample k edges uniformly without replacement using $\widetilde{O}(n+k)$ queries and $\widetilde{O}(n^2)$ time with high probability.*

Proof. Consider the process that at time step t (starting from $t = 0$) samples $G(p_t)$ for $p_t := 2^t/n^2$ until the sample $G(p_t)$ at $t = r$ has at least k edges. Then, subsample k edges uniformly from the sample $G(p_r)$.

Because edges are sampled i.i.d. from the Bernoulli Ber(p_t) distribution at each step t, the symmetry implies that the subsample of edges is a uniform sample of k edges. Using Lemma 7.2.4, each sample $G(p_t)$ takes $p_t \cdot \widetilde{O}(m) + \widetilde{O}(n)$ queries and $\widetilde{O}(n^2)$ time in expectation. Summing this over $0 \leq t \leq r$ gives an upper bound of $2p_r \cdot \widetilde{O}(m) + \widetilde{O}(nr)$ queries and $\widetilde{O}(n^2 r)$ time. Using the fact that the probability that $G(p)$ has less than k edges for $p = 2k/m$, is at most $e^{-k/4}$, we can conclude that $p_r \leq 4k/m$ with probability at least $1 - e^{-k/4}$. So the expected number of queries is upper bounded by $\widetilde{O}(nr + k) = \widetilde{O}(n + k)$ and the running time by $\widetilde{O}(n^2 r) = \widetilde{O}(n^2)$ using the fact that $r \leq O(\log n)$. □

Using Lemma 7.2.5 with $k = \widetilde{O}(n/\varepsilon^2)$, we can obtain a random subgraph H of the unknown graph G to get such a sample of edges using $\widetilde{O}(n/\varepsilon^2)$ queries, and hence apply existing approximations for all heavy subgraph problems.

7.2.5 Illustrative Examples

Non-Reconstructable Graphs

The unweighted graph pairs $(K_2, \bar{K_2})$ and $(K_3, \bar{K_3})$ are not distinguishable, since essentially the only feasible sets S are singleton sets (recall that by symmetry, $f(S) = f(\bar{S})$), and all of them have the same cut value. However, as we showed in Theorem 1.6.2, we can recover all unweighted simple graphs other than $K_2, \bar{K_2}, K_3, \bar{K_3}$ using comparison queries.

In contrast, many connected weighted graphs cannot be recovered using just comparison queries. This is not surprising: we cannot distinguish between a graph with weights w_e, and those with weight αw_e for all $\alpha > 0$. But there are other kinds of barriers apart from a simple scaling of weights—here is an example. Given any graph $G = (V, E)$ with unit weight edges, add

another spanning tree $T = (V, F)$ on the same vertices, and give the i^{th} edge $e_i \in F$ a weight of $w_T(e_i) = n^2 \cdot 2^i$. This weight function ensures that the F-weights of all cuts are distinct. Moreover, for a set $S \subseteq V$, $w(\partial S) = |\partial_G S| + w_T(\partial_T S)$. Since the w_T weights are scaled by n^2 (and hence much larger than the contribution due to the unit-weight edges), we know that

$$w(\partial S) > w(\partial S') \iff w_T(\partial_T S) > w_T(\partial_T S)$$

and hence we cannot infer the structure of the unit-weight edges.

Separations between Comparison and Value Queries

In the weighted setting, consider taking two disjoint cycles C_1 and C_2 of $k = n/2$ vertices and with unit-weight edges, and adding in a perfect matching of size k between them, with edges of cost ε. This is the *circular ladder graph*, with cross-edge weight ε.

Claim 7.2.6. There exist non-adaptive algorithms making $O(n^2)$ value queries to sets of size one and two that can reconstruct any graph, but for any non-adaptive algorithm in the comparison-query model making queries to sets of constant size (or even $\ll n/2$ size), there exist weighted n-vertex graphs G^+, G^- which cannot find the minimum cut.

Proof. Take the circular ladder graph with $\varepsilon = \frac{2}{k-1} \pm \gamma$ for some tiny $\gamma > 0$; the sign in front of the γ will determine whether the min-cut is a single-vertex cut with cost $2 + \varepsilon$, or the bisection (C_1, C_2) with cost $k\varepsilon$. However, we cannot distinguish these two cases if we compare sets S, T of size $\ll n/2$. (This is true even if we know the edges of the graph and the entire construction, except the sign in front of γ. This is in contrast to the value query setting, where asking the value of $f(S)$ for sets of size 1 and 2 allows us to completely reconstruct the graph. □

Claim 7.2.7. There exist non-adaptive algorithms making $O(n^2)$ value queries to reconstruct the graph, but for any non-adaptive algorithm in the comparison-query model, there exist weighted n-vertex graphs G^+, G^- which require $\exp(n)$ comparison queries to have high success probability.

Proof. Again, take the circular ladder graph, and rename the vertices uniformly at random. Assume that the cross-edge weight is $\varepsilon = \frac{2}{k-1} \pm \gamma$ for some tiny $\gamma > 0$; the sign in front of the γ will determine whether the degree cut or the partition (C_1, C_2) is optimal. Again, for value queries we can non-adaptively reconstruct the graph using $O(n^2)$ queries. But for comparison queries, the only way to learn the sign in front of γ would be to query the correct partition with a degree cut. Now if the queries are non-adaptive, and the vertices of the graph have been uniformly and randomly renumbered, the number of non-adaptive queries would have to be exponential in n in order to have a high probability of success. □

Finding Heaviest Edge Incident to a Vertex

We can show that comparison queries do not allow us to identify, for a vertex v, the heaviest edge adjacent to it. Finding this edge would be useful, e.g., in the min-cut algorithm of [179]. Indeed, consider the 4-vertex graph $\{a, b, c, d\}$ and the 6 edge weight variables. Here are two universes:

Edge Weight	Universe 1 ($w_{ab} > w_{ac}$)	Universe 2 ($w_{ab} < w_{ac}$)
w_{ab}	**1000.01**	**999.99**
w_{ac}	**999.99**	**1000.01**
w_{ad}	**10**	**10**
w_{bc}	100	100
w_{bd}	50	50
w_{cd}	1	1

Table 7.1: Two weight configurations for a 4-vertex graph that produce identical cut orderings but differ in which edge is heaviest incident to vertex a.

In both scenarios, $w_{ad} = 10$ is much less than w_{ab} and w_{ac} (which are both $\sim$1000). However, a calculation shows that the order of the cuts is **identical** in both universes:

$$f(\{a,d\}) > f(\{a\}) > f(\{a,b\}) > f(\{b\}) > f(\{a,c\}) > f(\{c\}) > f(\{d\}).$$

Hence, given this specific ordering of cuts, we cannot distinguish between Case 1 (where (a,b) is the max-weight edge adjacent to a) or Case 2 (where (a,c) is).

7.3 Matroids, Matchings, and Paths

In this section, we give efficient comparison-based algorithms to optimize over matroids, matroid intersections, and shortest paths.

7.3.1 Matroid Bases

We now give our algorithm to compute a minimum-weight basis of a matroid $\mathcal{M} = (U, \mathcal{I})$ with $|U| = n$ elements, where we are only allowed to compare matroid bases. Our main result for matroid bases is the following:

Theorem 1.6.3 (Matroid Bases)**.** *There is an algorithm outputs the minimum-weight basis of a matroid on n elements using $O(n \log n)$ comparison queries in $\widetilde{O}(n^2)$ time.*

Proof. First, let us consider graphical matroids, where the bases are spanning trees of a graph. Given any graph with edge set E, we can first partition the graph into its 2-vertex-connected components $E_1, \ldots, E_k$ (note that these are the edge sets of the components); the edge set of the minimum-weight spanning tree (MST) for E is the union of those for the E_i's, so it suffices to focus on a 2-connected component induced by the edges in some E_i. We claim we can simulate Kruskal's algorithm for MST, which just compares weights of edges: indeed, if this algorithm compares edges $e, e' \in E_i$, we can find a base B containing e, such that $B' := (B \setminus e) \cup e'$ is also a base. (We defer the proof, since it follows from the result below.) Now comparing e, e' has the same answer as comparing B, B'.

The same argument holds for arbitrary matroids, where we recall that the notion of 2-vertex connectivity in graphs extends to matroids [151]:

Lemma 7.3.1. *For any matroid $\mathcal{M} = (\mathcal{I}, U)$, let $\mathcal{B}(\mathcal{M})$ and $\mathcal{C}(\mathcal{M})$ be the collection of all bases and circuits in the matroid.*

1. *If elements $e, e' \in E$ share a circuit $C \in \mathcal{C}(\mathcal{M})$, then there exist bases $B, B' \in \mathcal{B}(\mathcal{M})$ such that $B' = B + e' - e$. Call such pair of elements e, e' **comparable**.*
2. *The binary relation $\gamma_{\mathcal{M}}$ on the elements of the matroid such that $(e, e') \in \gamma_{\mathcal{M}}$ iff either $e = e'$ or e, e' are comparable is an equivalence relation.*

Proof. For the first part, let B be the basis that extends the independent set $C - e'$. This means that the fundamental circuit in $B + e'$ is C implying that $B' := B + e' - e$ is also a basis. The second part is proved in [151, Proposition 4.1.2]. The equivalence classes in Lemma 7.3.1(2) are called the *connected components* of $\mathcal{M}$. □

Now we can run the analog of Kruskal's greedy algorithm for matroids on the connected components of $\mathcal{M}$; this uses $O(|U_i| \log |U_i|)$ element (and hence base) comparisons for each component; summing over all components gives us $O(|U| \log |U|)$, as claimed.

Running time and other details: The running time consists of the time taken to decompose the matroid into its connected components, and obtaining two bases B, B' such that $B \Delta B' = \{x, y\}$ for every pair x, y compared by the algorithm. For this, we use the results from [118] that connect the *basis exchange graph* with the connected components of the matroid. The following are a definition and a lemma:

For any basis B of $\mathcal{M}$, the basis exchange graph $H(B)$ is defined as the bipartite graph with color classes B and $U \backslash B$. For $x \notin B, y \in B$, we have $\{x, y\} \in H$ iff $B + x - y$ is also a basis. The following lemma connects this exchange graph with matroid connectivity:

Lemma 7.3.2 (Section 6, [118])**.** *For any basis $B \in \mathcal{M}$,*

1. *The nodes in the connected components (in the graphic sense) of $H(B)$ are the connected components (in the matroid sense) of matroid $\mathcal{M}$.*
2. *For x, y in the same connected component of $H(B)$ (or equivalently $\mathcal{M}$) such that $x \notin B$ and $y \in B$, if P is the shortest path connecting x and y in $H(B)$, then $B' := B \Delta P$ is a basis such that $B' + y - x$ is a basis.*

Given Lemma 7.3.2, the matroid decomposition step requires $O(n^2)$ matroid independence calls (in fact, the calls are always about whether a set is a basis or not) to build the exchange graph using Lemma 7.3.2 part 1 and $O(n^2)$ time to identify connected components.

Next, to simulate any pairwise comparison between elements x, y, we obtain the two bases $B \ni x, B' \ni y$ such that $B \Delta B' = \{x, y\}$ using part 2 of Lemma 7.3.2. The implementation of Kruskal's algorithm requires sorting the elements in each connected component which requires $O(c_i \log c_i)$ comparisons in each component with c_i elements. Each such comparison takes $O(n)$ time to find the path and write down the bases to input to the comparison oracle. Adding everything, we get the $O(n^2 \log n)$ time bound as claimed. □

7.3.2 Matroid Intersection

We now turn to finding *common independent sets* $S \in \mathcal{I}_1 \cap \mathcal{I}_2$ between two matroids $\mathcal{M}_1 = (U, \mathcal{I}_1)$ and $\mathcal{M}_2 = (U, \mathcal{I}_2)$. Our main result in this case is the following:

Theorem 1.6.4 (Matroid Intersection)**.** *Let $\mathcal{M}_1 = (U, \mathcal{I}_1)$ and $\mathcal{M}_2 = (U, \mathcal{I}_2)$ be two matroids defined on a ground set U containing n elements. Then, there is an algorithm that outputs the minimum-weight set that is in both $\mathcal{I}_1, \mathcal{I}_2$ using $O(n^4)$ comparison queries in $O(n^4)$ time.*

To prove this, we show that Edmond's classical weighted matroid intersection algorithm, as presented in [171, Section 41.3], can be implemented using only comparisons of common independent sets. As in the previous section, we first consider the simpler bipartite matching case, where the shortest augmenting paths algorithm constructs extremal matchings via shortest paths in an exchange graph. Our key observation is that these shortest-path steps—e.g., via Bellman–Ford—can themselves be implemented using only comparisons between matchings.

Formally, let $G = (V, E)$ be a bipartite graph with color classes $V = L \cup R$. Given an *extremal matching* M of size k (i.e., a min-cost matching with k edges), we describe a primitive to obtain an extremal matching M' of size $k + 1$.

The *M-exchange graph* D orients matching edges from R to L and non-matching edges from L to R in G. Let L' and R' be the unmatched vertices in L and R. An *M-augmenting path* is a directed path in D from a vertex in L' to one in R'. Define the edge length function $\ell : E \to \mathbb{R}$ as: $\ell(e) := -w(e)$ if $e \in M$, and $w(e)$ otherwise. The following result is standard (see, e.g., Theorem 3.5, Proposition 1 in Section 3.5 of [168]).

Lemma 7.3.3. *Let M be an extremal matching. Then:*

1. *The exchange graph D has no negative-length cycles.*
2. *If P is a minimum-length M-augmenting path, then $M' := M \triangle P$ is also extremal.*

By iteratively applying Lemma 7.3.3, we construct extremal matchings of increasing size, from size 0 to $n/2$, and return the one with minimum total weight. Moreover, to implement this using only comparisons between matchings, we describe a variant of the Bellman-Ford algorithm. For each $u \in L$, let $p_u^{(k)}$ be the shortest M-alternating path from L' to u using at most k matching edges. For matched $u \in L$, let $u' \in R$ be its partner. We initialize:

$$p_u^{(1)} = \begin{cases} \emptyset & \text{if } u \in L', \\ s \to u' \to u & \text{for } s = \arg\min_{s \in L'} \ell(s \to u' \to u). \end{cases}$$

Moreover, we recursively define $p_u^{(k+1)} = \min_{v \in L} \left(p_u^{(k)},\ p_v^{(k)} \to u' \to u \right)$, where the $\min$ operator returns the path with minimum total length $\ell(p)$, ties broken arbitrarily. Each such path P yields a matching $M \triangle P$, and satisfies $\ell(P) = w(M) - w(M \triangle P)$, so comparing path lengths reduces to comparing weights of the resulting matchings.

To reach a free vertex $t \in R'$, compute: $p_t = \min_{v \in L} \left(p_v^{(n)} \to t \right)$, which again defines an alternating path. Hence, the full algorithm can be implemented using only comparisons between matchings.

This insight extends naturally to the matroid intersection setting, where shortest augmenting paths must also be minimal (i.e., having fewest arcs among paths of equal weight), a property standard algorithms like Bellman–Ford already ensure. We show that the path-finding steps can again be carried out using only comparisons of common independent sets. The details appear in Section 7.6.2.

7.3.3 Shortest paths

Finally, when the combinatorial objects of interest are $s - t$ walks, we show that the Bellman-Ford shortest path algorithm can be used to find the shortest $s - t$ walk (which will be a path) using $O(n^3)$ many $s - t$ walk comparisons.

Theorem 1.6.5 (s-t walks)**.** *There is an algorithm that finds the minimum-length s-t path in a graph G (or a negative cycle, if one exists) using $O(n^3)$ s-t walk comparisons and $O(n^3)$ time.*

Proof. Assume that for every vertex $v \in V$, there exists a path from s to v and from v to t, since otherwise no s-t walk can pass through v, and we may ignore such vertices. For each $v \in V$, fix an arbitrary v-t walk t_v. Let $s_v^{(k)}$ denote the shortest s-v walk with at most k edges, where $s_v^{(0)} = \emptyset$.

Define the recurrence:

$$s_v^{(k+1)} = \min_{u \in V} \left(s_u^{(k)} \circ (u, v) \right), \tag{7.1}$$

where ties are broken arbitrarily. We can compute this because for any two walks $s_{u_1}^{(k)} \circ (u_1, v)$ and $s_{u_2}^{(k)} \circ (u_2, v)$, we can compare them by extending each with t_v and comparing total weights.

If w has no negative cycles, then $s_t^{(n-1)}$ is the shortest s-t path. To detect a negative cycle, check whether $w(s_u^{(n)}) < w(s_u^{(n-1)})$ for any $u \in V$.

Since there are $O(n^2)$ entries $s_u^{(k)}$, and each is computed by comparing $O(n)$ candidate walks, the total number of comparisons is $O(n^3)$. Although each comparison naively takes $O(n)$ time, we can amortize this using shared subwalks, leading to $O(n)$ time per table entry and total runtime $O(n^3)$. □

7.4 Linear Optimization for General Set Systems

Given these algorithms for general classes of set systems, we aim high and ask a very broad abstraction, which we call the *general linear optimization* (GLO) problem: suppose we are given a set of N points $\mathcal{P} = \{x_1, \ldots, x_N\} \subseteq \mathbb{R}^d$, and an unknown weight vector $w^* \in \mathbb{R}^d$. Again, we are only allowed comparison queries—given $x, y \in \mathcal{P}$, the comparison oracle responds with

$$\operatorname{sign}\left(\langle w^*, x - y \rangle\right), \tag{7.2}$$

i.e., the relative ordering of x and y in the ordering on $\mathcal{P}$ induced by w^*. The goal is to identify

$$x^* := \arg\min_{x \in \mathcal{P}} \langle w^*, x \rangle \tag{7.3}$$

using as few queries as possible. The above combinatorial optimization problems are all special cases of this problem where $d = |U|$, and we represent each set S by its (Boolean) indicator vector $\mathbb{1}_S \in \{0,1\}^d$.

There are two natural questions we can ask for GLO:

- **Certification:** Is it possible to certify that a candidate solution x is optimal using only a small number of comparisons?
- **Optimization:** Is it possible to find an optimal solution x^* using only a small number of comparisons? Naturally, this is at least as difficult as the certification problem, since we can compare this x^* to x.

Our results for linear optimization on more general point sets depend on the complexity of these point sets, via the notion of the *conic dimension*, which we define in Definition 7.4.3. We show how to optimize over point sets $\mathcal{P}$ with low conic dimension.

Theorem 1.6.7 (General Linear Optimization)**.** *There is an algorithm that, for any point set $\mathcal{P} \subseteq \mathbb{R}^d$ with conic dimension k and unknown weights $w \in \mathbb{R}^d$, returns the minimizer $x^* = \arg\min_{x \in \mathcal{P}} \langle w, x \rangle$ using at most $O(k \log k \log |\mathcal{P}|)$ comparisons.*

Since Boolean point sets have small conic dimension [113] (see Section 7.6.3 for a proof), we get our main theorem for Boolean optimization:

Theorem 1.6.6 (Boolean Linear Optimization)**.** *For any family $\mathcal{F} \subseteq 2^U$ and unknown weight function $w : U \to \mathbb{R}$, we can solve $\arg\min_{S \in \mathcal{F}} \sum_{e \in S} w_e$ using $O(n \log n \cdot \log |\mathcal{F}|) = \widetilde{O}(n^2)$ comparison queries, where $n = |U|$.*

As mentioned earlier, for the setting of *maximum cut* on a graph $G = (V, E)$, this result implies an upper bound of $\widetilde{O}(|V|^3)$ queries; moreover, our results can be made deterministic using the techniques of [114]. The work of [154] shows a lower bound of $\Omega(|V|^2)$ queries in the more informative *value query* model, thereby showing that a gap of only $O(|V|)$.

7.4.1 Warm-up: Query Complexity for Bounded Weights

Theorem 7.1.1 (Boolean Linear Optimization: Bounded Weights)**.** *For any family $\mathcal{F} \subseteq 2^U$ and unknown integer weight function satisfying $|w_e| \leq W$, we can sort all sets $S \in \mathcal{F}$ by their corresponding weight $\sum_{e \in S} w_e$ using $O(nW \log nW)$ comparison queries, where $|U| = n$.*

Proof. We can represent the feasible sets in $\mathcal{F}$ as points in the boolean hypercube $\{0,1\}^n$. Let this set of points be $\mathcal{P}$. The algorithm we describe not only finds the minimum-weight point but sorts all points in $\mathcal{P}$ by their weight.

First, we present a simpler algorithm with a query complexity of $O(n^2 W \log(nW))$ and then show how to improve it. The algorithm processes points from $\mathcal{P}$ in an arbitrary order, maintaining a sorted collection of the points seen so far.

At each step, we maintain a partition the set of points processed thus far as disjoint lists $\{L_v\}$, where each list L_v contains points that share the same objective function value v. The possible values for v lie in the range $[-nW, nW]$. Although we do not know the actual values v, we maintain the relative order of the lists.

For each list L_v, we maintain the affine subspace spanned by its points. This is done by storing a representative point $x_v \in L_v$ and a basis for the vector subspace spanned by the difference vectors $\{x - x_v \mid x \in L_v\}$. When a new point x_{new} is considered, we first check if its value can be determined without a query. For each existing list L_v, we test if x_{new} lies in the affine subspace spanned by L_v. If $x_{new} \in \text{span}(L_v)$, then $f(x_{new}) = v$, and we add x_{new} to L_v, updating the basis if necessary.

If x_{new} does not lie in any of the existing affine subspaces, we must use comparison queries to determine its value relative to the existing values. We perform a binary search over the sorted list of values. This either places x_{new} into an existing list L_v (if we find $f(x_{new}) = v$) or establishes a new list for a previously unseen value.

The complexity is determined by the total number of times a binary search is required. A binary search, which costs $O(\log(nW))$ queries, is performed only when a new point x cannot be placed into an existing list L_v by checking the affine span. This happens in two cases: (1) x has a value that has not been seen before, or (2) x has a value corresponding to a list L_v, but x increases the dimension of the affine subspace for L_v.

The number of distinct values is at most $2nW + 1$. For each of the $O(nW)$ possible values, the dimension of its corresponding subspace can increase at most $n - 1$ times. Therefore, the total number of events that trigger a binary search is bounded by $O(nW + n \cdot nW) = O(n^2 W)$. This leads to a total query complexity of $O(n^2 W \log(nW))$.

To improve this bound, we can make a simple but crucial observation. Instead of maintaining a separate subspace for each list L_v, we can maintain a single vector subspace A. This subspace A is spanned by all difference vectors encountered so far, i.e., $A = \text{span}\{x_i - x_v \mid x_i \in L_v, \forall v\}$. All these difference vectors lie in the subspace orthogonal to the true weight vector w, since $w^\top (x_i - x_v) = f(x_i) - f(x_v) = v - v = 0$.

When a new point x_{new} arrives, we check for each known representative x_v whether the difference $x_{new} - x_v$ lies in A. If $x_{new} - x_v \in A$, then $f(x_{new}) = f(x_v) = v$, and no query is needed. Otherwise, we use binary search on the values. The total number of dimensions we can learn for A is at most $n - 1$. The number of new values we can discover is at most $2nW + 1$. Therefore, the total number of queries is bounded by the cost of binary searches for each new value and the queries needed to build the basis for A. This gives a total query complexity of $O((n + nW) \log(nW)) = O(nW \log(nW))$. □

Remark 7.4.1. The proof relies on the fact that the total number of possible objective function

values is $O(nW)$. If there are B possible distinct values, the same argument yields a query complexity of $O((n+B)\log B)$. More crucially, the proof only uses a specific type of implication: if $x', x_1, \ldots, x_k$ all have the same value, then $f(x_i - x') = 0$. The existence of a linear combination $x - x' = \sum_i \alpha_i(x_i - x')$ then implies $f(x) = f(x')$. One can achieve better bounds by using inequalities and conic combinations rather than equalities and linear combinations, which is what subsequent extensions explore.

7.4.2 Envelopes and the conic dimension of Point Sets

Let us begin by defining the *envelope* and *conic dimension* of a point set. Recall that a cone of a point set is the collection of all positive linear combinations of these points.

Definition 7.4.2. The *envelope* of a *sequence* of points $\sigma = (y_1, \ldots, y_n) \in (\mathbb{R}^d)^n$ is

$$\mathrm{env}(\sigma) := \mathrm{cone}(\{y_j - y_i\}_{1 \le i < j \le n}) = \mathrm{cone}(\{y_{i+1} - y_i\}_{1 \le i < j \le n}). \tag{7.4}$$

The idea behind the envelope is simple: if σ is the total order induced by w^* on the points $\{y_1, \ldots, y_n\}$ (i.e., if $\langle w^*, y_i - y_j \rangle \ge 0$ for $i < j$, then taking the envelope of σ gives us all the "implications" of this ordering, and allows us to derive implications without performing any further calculations.

Definition 7.4.3. The *conic dimension* of a *set* $\mathcal{Y} \subseteq \mathbb{R}^d$ denoted by $\mathrm{ConicDim}(\mathcal{Y})$ is the largest integer k for which there exists a length-k sequence $\sigma = (y_1, \ldots, y_k) \in \mathcal{Y}^k$ of points from $\mathcal{Y}$, such that for each $t \in \{2, \ldots, k\}$ we have

$$y_t - y_1 \notin \mathrm{env}(\sigma^{1:t-1}), \tag{7.5}$$

where $\sigma^{1:t}$ denotes the prefix of the first t elements of sequence σ.

In other words, each $\langle w^*, y_t - y_1 \rangle \ge 0$ is not implied by the comparisons in the prefix. Note that the conic dimension is defined for a set, whereas the notion of an envelope is defined for a sequence. A note on the name: [113] defined the *inference dimension* as being the size of a largest "inference-independent" set where none of the inequalities are implied by the others. Our definition is very similar to theirs, with two minor difference: (a) we restrict to linear optimization, and (b) we consider sequences where each comparison is not implied by the previous ones. Since implications using linear inequalities has a natural geometric visualization as a cone, we call it *conic dimension*.

Given these definitions, it is possible to show that if a point set $\mathcal{P}$ has conic dimension k, then for the optimal point $y^* \in \mathcal{P}$ there exists proofs of optimality of size at most $k-1$. In the next section, we give an algorithm for the optimization problem, which finds an optimal point y^* and also gives a proof of size $O(k \log k \log |\mathcal{P}|)$.

7.4.3 An Algorithm for Finding a Minimizer

We now show the proof of the main result for general linear optimization:

Theorem 1.6.7 (General Linear Optimization)**.** *There is an algorithm that, for any point set $\mathcal{P} \subseteq \mathbb{R}^d$ with conic dimension k and unknown weights $w \in \mathbb{R}^d$, returns the minimizer $x^* = \arg\min_{x \in \mathcal{P}} \langle w, x \rangle$ using at most $O(k \log k \log |\mathcal{P}|)$ comparisons.*

For point set $\mathcal{P} = \{x_1, \ldots, x_N\}$, consider the following iterative algorithm, which distills the approach of [113]:

Algorithm 9 Iterative Sieving Algorithm

1: **while** $|\mathcal{P}|$ is more than $O(k)$ **do**
2: Independently include each point of $\mathcal{P}$ in a subset $\mathcal{Y}$ with probability $2k/N$
3: $\sigma \leftarrow$ result of sorting $\mathcal{Y}$ using $O(k \log k)$ expected comparisons
4: $y \leftarrow$ first (smallest) element of σ
5: Eliminate all points $x \in \mathcal{P} \setminus \{y\}$ such that $x - y \in \text{env}(\sigma)$
6: **end while**
7: Sort $\mathcal{P}$ using a brute-force algorithm

The proof of the algorithm follows immediately from the next lemma, which shows that each iteration reduces the number of points in $\mathcal{P}$ by half in expectation. We defer the proof to Section 7.6.3.

Lemma 7.4.4. *The expected number of points that are eliminated from $\mathcal{P}$ at step 3 is at least $|\mathcal{P}|/2$.*

We observe that the algorithm optimizes the number of queries, and not the runtime, since both steps 2 and 3 cannot be efficiently executed in general; it remains an interesting direction to make some version of this algorithm efficient for special classes of problems.

7.5 Future Directions

Our work initiates the study of combinatorial optimization under perhaps the weakest interesting model of information: the ability to compare two potential solutions. In this pairwise-comparison model, we show how to solve the graph min-cut problem for simple graphs (and give results also for graph reconstruction, which is non-trivial in this setting). We also show how to optimize over matroids, matroid intersections, and s-t paths using just pairwise comparisons. Finally, we show that all Boolean point sets (and indeed, all point sets of bounded precision) have small query complexity, even though their computational complexity may be large.

Many intriguing open questions arise from this work. We highlight some of the most pressing directions below.

- **Weighted Graphs.** A natural next step is to extend our min-cut algorithm from simple graphs to weighted graphs. Finding an efficient comparison-based algorithm for the weighted case is a primary open problem.

- **Submodular Minimization.** Can we optimize objectives beyond modular functions? Determining if submodular functions can be minimized with a polynomial number of comparison queries—even with exponential computation time—would be a significant generalization of our results.
- **Characterizing Efficient Problem Classes.** What general properties of a set system permit efficient optimization? Our results for specific structures like matroids are encouraging. A central challenge is to characterize the properties that make the general framework of Section 7.4 efficient for broader classes of combinatorial problems.
- **Noisy Comparisons.** How can we handle noisy queries? If each query is correct only with a probability of $\frac{1}{2}+\varepsilon$, standard binary search approaches fail. While some techniques from the value query model might be adaptable [75], this direction remains largely unexplored.
- **Adaptive vs. Non-adaptive Gaps.** What is the query complexity gap between comparison and value queries in the **adaptive setting**? While a simple separation exists in the non-adaptive case, a deeper understanding of the power of adaptivity in the comparison model is needed.

7.6 Appendix to Chapter 7

7.6.1 Proofs from Section 7.2

We begin with a simple observation allowing us to test, for a vertex u and any non-empty set S that is a proper subset of $U-u$, whether a majority of u's neighbors lie inside S.

Observation 7.6.1. Suppose we perform a cut comparison query on the pair S and $S+u$. Then the result of the query is:

$$\text{sign}(|\partial(S)| - |\partial(S+u)|) = \text{sign}\left(|\partial(u,S)| - \frac{1}{2}|\partial(u)|\right). \tag{7.6}$$

Proof. Indeed,

$$\begin{aligned}|\partial(S)| - |\partial(S+u)| &= |\partial(u,S)| - |\partial(u, U\setminus(S+u))| \\ &= 2|\partial(u,S)| - |\partial(u)|.\end{aligned}$$

Therefore, the sign of the query $(S, S+u)$ indicates whether a majority of u's edges go into S. □

The above observation allows us to reason about neighborhood structure. We now formalize how to use this to isolate individual neighbors.

Lemma 7.6.2 (tipping point)**.** *For $u \in U$, consider a set $\emptyset \neq S \subseteq U \setminus \{u\}$, and a sequence $v_1, v_2, \ldots, v_t$ of vertices in $U \setminus (S \cup \{u\})$. Define $S_i = S \cup \{v_1, \ldots, v_i\}$. Suppose $S_t \neq U - u$*

and the comparisons of the queries (S_0, S_0+u) and (S_t, S_t+u) differ. Then there exists an index $i \in [t]$ such that

$$\text{sign}(|\partial(S_{i-1})| - |\partial(S_{i-1}+u)|) < \text{sign}(|\partial(S_i)| - |\partial(S_i+u)|). \tag{7.7}$$

The corresponding vertex v_i must be a neighbor of u, and can be identified using $O(\log t)$ queries and $\widetilde{O}(n)$ time via binary search.

Proof. By Observation 7.6.1, the $\text{sign}(|\partial(S)| - |\partial(S+u)|)$ is monotonic in S, since the quantity $|\partial(u, S)|$ increases as S grows. Therefore, if the sign changes between S_0 and S_t, then by monotonicity, it must strictly increase at some index i, which proves (7.7). Moreover, using the equality in (7.6) again, (7.7) can be rewritten as

$$|\partial(u, S_{i-1})| < |\partial(u, S_i)|, \tag{7.8}$$

and hence u is adjacent to the vertex v_i. Finally, a binary search over the chain finds such an i using $O(\log t)$ queries. The binary search performs $O(\log t)$ queries, each requiring $O(n)$ time to write bit vectors, for a total of $O(n \log t) = \widetilde{O}(n)$. The additional binary search logic takes $O(\log t)$ time. □

We now apply these ideas to construct balanced sets around each vertex.

Lemma 7.6.3 (median sets)**.** *For any vertex u with $\partial(u) \neq \emptyset$, and an ordering $v_1, \ldots, v_{n-1}$ of the vertices in $U - u$, there exist disjoint sets $S_u^-, S_u^+ \subseteq U \setminus \{u\}$ that are a prefix and a suffix of the sequence such that*

$$|\partial(u, S_u^-)| = |\partial(u, S_u^+)| = \left\lceil \frac{|\partial(u)|}{2} \right\rceil - 1. \tag{7.9}$$

These sets can be found using $O(\log n)$ queries and $\widetilde{O}(n)$ time.

Proof. Let $L_j = \{v_1, v_2, \ldots\ldots, v_j\}$ and $R_j = \{v_j, v_{j+1}, \ldots, v_{n-1}\}$ for $j \in [n-1]$. First, check if $\partial(v_1) > \partial(v_1 + u)$ in which case, we can conclude $S_u^- = \emptyset, S_u^+ = \{v_2, v_3, \ldots, v_{n-1}\}$ because $\{u, v_1\}$ has to be the only edge using Observation 7.6.1. Similarly for v_{n-1}. So from now on, assume that $\partial(v_1) \leq \partial(v_1+u)$ and $\partial(v_{n-1}) \leq \partial(v_{n-1}+u)$. Equivalently, this can also be written as $\partial(R_2) \geq \partial(R_2 + u), \partial(L_{n-2}) \geq \partial(L_{n-2} + u)$.

In what follows, we will first describe how to compute S_u^- and then use a similar argument to compute S_u^+. If $\partial(v_1) = \partial(v_1 + u)$, then again we can simply conclude $S_u^- = \emptyset$ because this implies that $|\partial(u)| = 2$ and $\{u, v_1\}$ is an edge. Otherwise, if $\partial(v_1) < \partial(v_1+u)$, we can search for S_u^- as a tipping point for the chain defined by the sequence $v_1, v_2, \ldots, v_{n-2}$ using Lemma 7.6.2 because $\partial(v_1) < \partial(v_1 + u)$ and $\partial(v_1, \ldots, v_{n-2}) \geq \partial(v_1, \ldots, v_{n-2} + u)$. By the arguments in Lemma 7.6.2, there exists $i \in [n-2]$ such that $v_i \in N(u)$ and

$$|\partial(u, L_{i-1})| < \tfrac{1}{2}|\partial(u)|, \quad |\partial(u, L_i)| \geq \tfrac{1}{2}|\partial(u)|. \tag{7.10}$$

This implies that $S_u^- := L_{i-1}$ satisfies $|\partial(u, S_u^-)| = \lceil \frac{|\partial(u)|}{2} \rceil - 1$. We can use the same argument to get S_u^+. Moreover, since S_u^- and S_u^+ are prefix and suffix of the same ordering, and contain no more than half the neighbors of u, they must be disjoint. The proof makes two calls to Lemma 7.6.2, one for each ordering, with $O(n)$ preprocessing. Each call takes $\widetilde{O}(n)$ time, so the total running time is $\widetilde{O}(n)$. □

Proof of Structural Primitives Lemma

We can now prove the main structural result Theorem 7.2.1, which we restate for convenience:

Theorem 7.2.1 (Structural Primitives)**.** *When $n \geq 4$, for any vertex $u \in U$:*

(i) Given a set of k vertices $A = \{v_1, \ldots, v_k\} \subseteq U \setminus \{u\}$, the neighbors of u in A can be determined using $k + O(\log n)$ queries and $\widetilde{O}(n + k)$ time.

(ii) Given an ordered subset $T = (v_1, \ldots, v_t) \subseteq U \setminus \{u\}$, either the neighbor $v_i \in N(u)$ of u with the smallest index $i \in [t]$ (or a certificate that $N(u) \cap T = \emptyset$) can be found using $O(\log n)$ queries and $\widetilde{O}(n)$ time.

Before we prove the theorem, we first prove a primitive that shows how to identify if a vetrex is isolated using constant queries when $n \geq 4$:

Lemma 7.6.4 (Home Alone)**.** *For any three vertices $u, v, w \in U$, vertex u is isolated iff*

$$|\delta(v)| = |\delta(u + v)|, |\delta(w)| = |\delta(u + w)|, |\delta(v + w)| = |\delta(u + v + w)|. \tag{7.11}$$

Proof. The three equalities are trivial when u is isolated. Given the equalities, we observe that they can be written equivalently as

$$|\delta(u, v)| = \tfrac{1}{2}|\delta(u)|, |\delta(u, w)| = \tfrac{1}{2}|\delta(u)|, |\delta(u, v + w)| = \tfrac{1}{2}|\delta(u)|. \tag{7.12}$$

Using the fact that $|\delta(u, v + w)| = |\delta(u, v)| + |\delta(u, w)|$ gives $|\delta(u)| = 0$. □

Proof of Theorem 7.2.1. We first check if $\partial(u) = \emptyset$ using Lemma 7.6.4. If $\partial(u) = \emptyset$, there are no neighbors of u in A in part (i) and we can certify that $N \cap T = \emptyset$ in part (ii). Assume $\partial(u) \neq \emptyset$ from here on.

In proving part (i), and determining which of $v_1, \ldots, v_k \in U \setminus \{u\}$ are neighbors of u, we begin by computing the disjoint sets $S_u^-, S_u^+ \subseteq U \setminus \{u\}$ with exactly $\lceil \frac{|\partial(u)|}{2} \rceil - 1$ neighbors of u; this takes $O(\log n)$ queries and $\widetilde{O}(n)$ time due to Lemma 7.6.3.

For each v_i we choose S to be one of S_u^- or S_u^+ in which v_i does not appear—say $S := S_u^+$. Since $|\partial(u, S)| < \frac{1}{2}|\partial(u)|$, we know from Observation 7.6.1 that $\text{sign}(|\partial(S)| - |\partial(S + u)|) = -1$. We then add v_i to S and ask for $\text{sign}(|\partial(S + v_i)| - |\partial(S + v_i + u)|)$. Since $\partial(S, u)$ is just one edge short of reaching half the degree of u, this sign increases if and only if $v_i \in N(u)$. Thus, each test requires one query, and the total query complexity is $k + O(\log n)$. After computing the median sets S_u^- and S_u^+ in $\widetilde{O}(n)$ time, we partition the vertices $v_1, \ldots, v_k$ into two groups based on whether each v_i lies in S_u^- or S_u^+; this takes $O(k)$ time using bit vector lookups. For each

group, we fix the corresponding set $S := S_u^+$ or S_u^- and write its bit vector once in $O(n)$ time. Each query $(S+v_i, S+v_i+u)$ is then constructed in $O(1)$ time via bit flipping. The total running time is $\widetilde{O}(n+k)$.

This proves Theorem 7.2.1(i).

For part (ii) of the lemma, consider the ordered sequence $T = (v_1, \ldots, v_t) \subseteq U \setminus \{u\}$ and extend it to a total ordering of $U \setminus \{u\}$ by appending the remaining vertices arbitrarily, forming the sequence $(v_1, \ldots, v_t, v_{t+1}, \ldots, v_{n-1})$. Define suffixes $R_k := \{v_k, \ldots, v_{n-1}\}$ for each k; by Lemma 7.6.3, there exists some k such that

$$|\partial(u, R_k)| = \left\lceil \tfrac{|\partial(u)|}{2} \right\rceil - 1, \tag{7.13}$$

and such a suffix $S := R_k$ can be found using $O(\log n)$ queries and $\widetilde{O}(n)$ time via Lemma 7.6.3. To verify that $|\partial(u, T)| > 0$, consider the sequence $(v_{t+1}, \ldots, v_{n-1}, v_1, \ldots, v_t)$ with T placed all the way at the end. The suffix median of this ordering computed using Lemma 7.6.3 is the same as S iff $|\partial(u, T)| > 0$. Otherwise, define the nested sequence

$$S_0 := S, \tag{7.14}$$
$$S_i := S \cup \{v_1, \ldots, v_i\} \quad \text{for } i = 1, \ldots, \min(t, k). \tag{7.15}$$

(Recall that $|T| = t$ and $|S| = n - k + 1$.) We have:

$$|\partial(u, S_0)| = |\partial(u, S)| = \left\lceil \tfrac{|\partial(u)|}{2} \right\rceil - 1, \tag{7.16}$$
$$|\partial(u, S_{\min(t,k)})| = |\partial(u, S)| + |\partial(u, T \setminus S)| \geq \tfrac{1}{2}|\partial(u)|. \tag{7.17}$$

To justify the inequality in (7.17): if $|\partial(u, T)| > 0$, then $|\partial(u, T \setminus S)| > 0$, because otherwise if $|\partial(u, T \setminus S)| = 0$, then $N(u) \subseteq S$. But this would imply $|\partial(u, S)| = |\partial(u)|$, contradicting the fact that less than half the edges incident to u go into S (by the choice of S). Hence, $T \setminus S$ must contain a neighbor of u.

By Lemma 7.6.2, there exists an index i such that the sign of the query $(S_{i-1}, S_{i-1}+u)$ is strictly smaller than that of $(S_i, S_i + u)$. The corresponding vertex v_i is the first vertex in T adjacent to u, and binary search over $i \in [\min(t, k)]$ finds it using $O(\log n)$ queries and $\widetilde{O}(n)$ time. □

Lemma 7.2.3 (Edge Extraction)**.** *When $n \geq 4$, for any vertex u, a subset $T \subseteq U - u$ of vertices, and any integer k, we can extract $r = \min(k, \partial(u, T))$ edges from $\partial(u, T)$ using $O(r \log n)$ queries and $\widetilde{O}(n)$ time.*

Proof. We assume that $|\partial(u, T)| \leq \left\lfloor \frac{1}{2}|\partial(u)| \right\rfloor + 1$; otherwise, we split T into $T_+ := T \cap S_u^+$ and $T_- := T \cap S_u^-$ using the median sets from Lemma 7.6.3, and recursively extract edges from each. Both subsets satisfy the same degree bound, and the total query and runtime complexity remains the same. Let S be a median set disjoint from T, which exists by construction and can be found using $O(\log n)$ queries and $\widetilde{O}(n)$ time. Let $T = \{v_1, \ldots, v_t\}$, and define the nested sets $S_j := S \cup \{v_1, \ldots, v_j\}$. We maintain two bit vectors: one for the fixed set S, and one for the

growing prefix of T. In each iteration, we find the smallest index j such that the sign of the query $(S_{j-1}, S_{j-1}+u)$ is strictly smaller than that of (S_j, S_j+u), indicating that $v_j \in N(u)$. We locate this index using the doubling version of binary search. Once v_j is found, we report the edge $\{u, v_j\}$, remove $v_1, \ldots, v_j$ from T, and repeat. Since we extract at most r edges, we perform at most r such searches, each requiring $O(\log n)$ queries, for a total of $O(r \log n)$ queries overall.

Running time. Constructing the median set S takes $\widetilde{O}(n)$ time. To construct the query sets, we reuse the bit vector for S and build each query by appending a prefix $\{v_1, \ldots, v_i\} \subseteq T$. In each iteration, the doubling search explores prefixes of geometrically increasing size until it finds a neighbor at index j. The total size of all prefixes queried in that iteration is $O(j \log t)$, so the time spent constructing queries is $O(j \log t)$. Deleting $v_1, \ldots, v_j$ from T also takes $\widetilde{O}(j)$ time. The values $j_1, \ldots, j_r$ across all r iterations sum to at most n as these are gaps between the edge indices, so the overall total time taken is $O\left(\sum_{i=1}^{r} j_i \log n\right) = \widetilde{O}(n)$. □

Cut sparsifiers using $\widetilde{O}(n)$ queries

Using Lemma 7.2.4, we show that we can implement the sparsifier construction from [162, Algorithm 3.4] and the min-cut computation from [162, Algorithm 4.1] using $\widetilde{O}(n/\varepsilon^2)$ cut comparison queries. The mentioned algorithms and relevant theorems from [162] are mentioned verbatim below:

Algorithm 10 (Approximating Edge Strengths and Sampling a Sparsifier H)

Input: An accuracy parameter ε, and a cut-query oracle for graph G.

1. Initialize an empty graph H on n vertices and $G_0 \leftarrow G$.
2. For $j = 0, \ldots, \log n$, set $\kappa_j = n2^{-j}$ and:
   (a) Subsample G'_j from G_j by taking each edge of G_j with probability

$$q_j = \min\left(\frac{100 \cdot 40 \cdot \ln n}{\kappa_j}, 1\right).$$

   (b) In each connected component of G'_j:
      i. While there exists a cut of size $\leq q_j \cdot \frac{4}{5}\kappa_j$, remove the edges from the cut. Let the connected components induced by removing the cut edges be $C_1, \ldots, C_r$.
      ii. For every $i \in [r]$ and every edge (known or unknown) with both endpoints in C_i, set the approximate edge strength $k'_e := \frac{1}{2\kappa_j}$ (alternatively, subsample every edge in $C_i \times C_i$ with probability $2q_j/\varepsilon^2$ and add it to H with weight $\varepsilon^2/2q_j$).
      iii. Set G_{j+1} as the graph obtained by contracting C_i for each $i \in [r]$ in G_j.

Output: The edge strength approximators $\{k'_e\}$ (or the sparsifier H).

Theorem 7.6.5 (Theorem 3.5 from [162])**.** *For each edge $e \in G$, the approximate edge strength given in Algorithm 10 satisfies $\frac{1}{4}k_e \leq k'_e \leq k_e$. Furthermore, the algorithm requires $\widetilde{O}(n/\varepsilon^2)$ oracle queries to produce a sparsifier H with the following properties:*

- *H has $O(n \log n/\varepsilon^2)$ edges.*

- *The maximum weight of any edge e in H is $O(\varepsilon^2 k_e / \log n)$.*
- *Every cut in H approximates the corresponding cut in G within a $(1 \pm \varepsilon)$ factor.*

Algorithm 11 Simpler global Min Cut with $\widetilde{O}(n)$ oracle queries

---

Input: Oracle access to the cut values of an unweighted simple graph G.

1. Compute all of the single-vertex cuts.
2. Compute a sparsifier H of G using Algorithm 10 with small constant ε.
3. Find all non-singleton cuts of size at most $(1 + 3\varepsilon)$ times the size of the minimum cut in H, and contract any edge which is not in such a cut. Call the resulting graph G'.
4. If the number of edges between the super-vertices of G' is $O(n)$, learn all of the edges between the super-vertices of G' and compute the minimum cut.

Output: Return the best cut seen over the course of the algorithm.

Theorem 7.6.6 (Theorem 4.2 from [162])**.** *Algorithm 11 uses $\widetilde{O}(n)$ queries and finds the exact minimum cut in G with high probability.*

Proof of Theorem 1.6.1. For each $0 \leq j \leq \log n$, step 2(a) of Algorithm 10 can be implemented using Lemma 7.2.4 with $O(|E_j| \cdot q_j \cdot \log n) + \widetilde{O}(n)$ comparison queries in expectation; which is shown to be $\widetilde{O}(n)$ in the proof of Theorem 7.6.5. The runtime bound of the same step is $\widetilde{O}(n^2)$. Step 2(b)(ii) of Algorithm 10 also takes $\widetilde{O}(n/\varepsilon^2)$ queries and $\widetilde{O}(n^2)$ time because the number of edges sampled and added to H is shown to be $\widetilde{O}(n/\varepsilon^2)$ (see proof of Theorem 4.2, [162]) and this sub-sampling can be done again using Lemma 7.2.4. In Algorithm 11, since step (1) is only used to compare the singleton cuts to output the minimum cut seen by the algorithm, we can implement this using comparison in $O(n)$ queries and time. In step (3), all the approximate cuts can be computed in $\widetilde{O}(n^2)$ time using [103]. In step (4), all the edges of G' can be learned using Lemma 7.2.4 with $p = 1$. The proof of Theorem 7.6.6 shows that the number of edges in G' is at most $O(n)$ with high probability. This implies that all the steps in the algorithms take $\widetilde{O}(n/\varepsilon^2)$ queries and $\widetilde{O}(n^2)$ time. There are at most $\log n$ steps in any algorithm which proves the desired bounds. □

7.6.2 Proofs from Section 7.3

Generalization to Matroid Intersection

We now extend the concepts from bipartite matching to matroid intersection. Let $\mathcal{M}_1 = (E, \mathcal{I}_1)$ and $\mathcal{M}_2 = (E, \mathcal{I}_2)$ be matroids over a common ground set E, and let $Y \in \mathcal{I}_1 \cap \mathcal{I}_2$ be a common independent set. Define the *exchange graph* $H(\mathcal{M}_1, \mathcal{M}_2, Y)$ as a directed bipartite graph with color classes Y and $E \setminus Y$. For $y \in Y$ and $x \in E \setminus Y$:

$$(y, x) \in H \iff Y - y + x \in \mathcal{I}_1,$$
$$(x, y) \in H \iff Y - y + x \in \mathcal{I}_2.$$

Let $H(\mathcal{M}_1, Y), H(\mathcal{M}_2, Y)$ be the subgraphs of H with edges corresponding to $\mathcal{M}_1, \mathcal{M}_2$ respectively.

We know the following lemma (see [168, Section 10.4]):

Lemma 7.6.7 (Matching lemmas)**.** *For $Y \in \mathcal{I}_1$, let $H_1 = H(\mathcal{M}_1, Y)$.*

(a) If $Z \in \mathcal{I}_1$ with $|Y| = |Z|$, then H_1 contains a perfect matching on $Y \Delta Z$.
(b) If $Z \subseteq E$ such that H_1 contains a unique perfect matching on $Y \Delta Z$, then $Z \in \mathcal{I}_1$.

Let $X_1 := \{x \in E \setminus Y : Y + x \in \mathcal{I}_1\}$ and $X_2 := \{x \in E \setminus Y : Y + x \in \mathcal{I}_2\}$. Given a weight function $w : E \to \mathbb{R}$, define the length of any path (or cycle) P in H by:

$$\ell(P) := \sum_{e \in P \cap Y} w_e - \sum_{e \in P \setminus Y} w_e.$$

A common independent set $Y \in \mathcal{I}_1 \cap \mathcal{I}_2$ is *extreme* if it minimizes weight among all common independent sets of the same size. We know the following lemmas (see Theorem 10.7, Theorem 10.11, Theorem 10.12 in [168])

Lemma 7.6.8. *The subset $Y \in \mathcal{I}_1 \cap \mathcal{I}_2$, is a maximum cardinality common independent set iff there is no directed path from X_1 to X_2 in H.*

Lemma 7.6.9. *Consider a common independent set $Y \in \mathcal{I}_1 \cap \mathcal{I}_2$.*

(i) Y is extreme if and only if $H(\mathcal{M}_1, \mathcal{M}_2, Y)$ contains no directed cycle of negative length.
(ii) If Y is extreme, and P is a shortest (minimal-hop) path from X_1 to X_2 in H, then $Y \triangle P$ is also extreme.

To mimic the Bellman–Ford algorithm, we generalize the notion of augmenting paths using *minimal paths and cycles*, defined with respect to both weight and hop-count.

Minimal Cycles and Paths. A cycle C in H is *minimal* if no proper subcycle C' satisfies $\ell(C') \leq \ell(C)$. Similarly, for $y \in Y$, define $\mathcal{P}(y, t)$ to be the set of directed paths from X_1 to y in H using at most $2t - 1$ arcs. A path $P \in \mathcal{P}(y, t)$ is *minimal* if it is lexicographically minimal with respect to length and number of arcs:

$$\text{For all } P' \in \mathcal{P}(y, t), \quad (\ell(P'), |P'|) \succeq (\ell(P), |P|).$$

In the following lemma, we prove our main observation that augmenting an extreme common independent by a minimal path gives a common independent set.

Lemma 7.6.10 (Minimal paths and cycles)**.** *If Y is extreme:*

1. *For any minimal cycle C in H, we have $Y \triangle C \in \mathcal{I}_1 \cap \mathcal{I}_2$.*
2. *For any minimal path $P \in \mathcal{P}(y, t)$, we have $Y \triangle P \in \mathcal{I}_1 \cap \mathcal{I}_2$.*

Proof of Lemma 7.6.10. For the proof of part (1), suppose $Z := Y \Delta C \notin \mathcal{I}_1 \cap \mathcal{I}_2$, assume by symmetry that $Z \notin \mathcal{I}_1$. Let N, M be the matching edges in C corresponding to $\mathcal{M}_1$ and $\mathcal{M}_2$ respectively. Using Lemma 7.6.7(a), we know that there exists a different matching $N' \neq N$ between the vertices $VC \cap Y$ and $VC \setminus Y$ with edges corresponding to $\mathcal{M}_1$.

Consider the multiset union of edges $N \cup N' \cup 2M$ which takes two copies of every edge from M and a copy of every edge from N and N'. Since these set of edges enter and leave every vertex exactly twice, the set of edges can be decomposed into a collection of cycles $C_1, \ldots, C_p$ for some $p \geq 2$ such that some of the cycles have strictly fewer vertices than C. We also have $\sum_{i \in p} \ell(C_i) = 2\ell(C)$. Using Lemma 7.6.9(i), we know $\ell(C_i) \geq 0$ and since $p \geq 2$, there exists a cycle C_i with lesser weight or the same weight with lesser number of vertices as C contradicting the minimality of C.

The proof of part (2) proceeds in a few steps:

(2a) (**No re-entry**) First, we prove that a minimal path in $\mathcal{P}(y, t)$ never re-enters X_1 again. If $P = (x_0, y_1, x_1, \ldots, y_k, x_k, \ldots, x_{t-1}, y_t)$ with $x_k \in X_1$, then observe that there is an edge from y_k to x_0 because $Y + x_0 \in \mathcal{I}_1$ as $x_0 \in \mathcal{I}_1$ which implies $Y + x_0 - y_k \in \mathcal{I}_1$ and hence the edge. If we let $C = (x_0, y_1, x_1, \ldots, y_k, x_0)$, and $P' = (x_k, y_{k+1}, \ldots, x_{t-1}, y_t)$, we have $\ell(P) = \ell(C) + \ell(P')$ implying that P' has smaller length and lesser number of arcs than P as $\ell(C) \geq 0$ from Lemma 7.6.9(i) contradicting the minimality of P. An identical argument can be used to show that vertices are not revisited again in minimal paths.

(2b) ($Y \Delta P \in \mathcal{I}_1$) Next, we show that $Y \Delta P \in \mathcal{I}_1$. Let $Z = \{y_1, x_1, \ldots, y_{t-1}, x_{t-1}\}$. We first show that $Y \Delta Z \in \mathcal{I}_1$. Let N be the set of matching edges (y_i, x_i) corresponding to $\mathcal{M}_1$ for $1 \leq i \leq t-1$ and M be the set of matching edges (x_{j-1}, y_j) corresponding to $\mathcal{M}_2$ for $1 \leq j \leq t$. If $Y \Delta Z \notin \mathcal{I}_1$, then there is a different matching N' between $Z \backslash Y$ and $Z \cap Y$ using Lemma 7.6.7(a). Consider the multiset union of edges $N \cup N' \cup 2M \cup 2(y_k, x_0)$.

Since these set of edges enter and leave every vertex exactly twice, the set of edges can be decomposed into a collection of cycles $C_1, \ldots, C_p$ for some $p \geq 2$ such that some of the cycles have strictly fewer vertices than C. We also have $\sum_{i \in p} \ell(C_i) = 2\ell(C)$. Using Lemma 7.6.9(i), we know $\ell(C_i) \geq 0$. Consider the two different cycles (say) C_1, C_2 that each contain a copy of the edge (y_t, x_0). We know that one of these cycles (say) C_1 should have $\ell(C_1) \leq \ell(C)$ and $|C_1| < |C|$. Removing the (y_t, x_0) edge from C_1 gives a path (say) P_1 such that $(\ell(P), |P|) \succeq (\ell(P_1), |P_1|)$ contradicting the minimality of P.

Now we will show that $Y \Delta Z \in \mathcal{I}_1 \Rightarrow Y \Delta P \in \mathcal{I}_1$. First, observe that $r_{\mathcal{M}_1}(Y \cup Z) = |Y|$ because $r_{\mathcal{M}_1}(Y + z) = r_{\mathcal{M}_1}(Y) = |Y|$ for every $z \in Z$ as we showed $Z \cap X_1 = \emptyset$ in Item (2a). On the other hand, the rank $r_{\mathcal{M}_1}(Y \cup P) \geq |Y| + 1$ because $Y \cup P \supseteq Y + x_0 \in \mathcal{I}_1$. Since $Y \Delta Z \subseteq Y \cup P$, there is an element in $\{x_0, y_1, y_2, \ldots, y_{t-1}\}$ that extends $Y \Delta Z$. This has to be x_0 because we know $r_{\mathcal{M}_1}(Y \cup Z) = |Y|$. This implies that

$$(Y \Delta Z) + x_0 = (Y \Delta P) + y_t \in \mathcal{I}_1 \Rightarrow (Y \Delta P) \in \mathcal{I}_1.$$

(2c) ($Y \Delta P \in \mathcal{I}_2$) Finally, we show that $Y \Delta P \in \mathcal{I}_2$. Let N be the set of matching edges (y_i, x_i) corresponding to M_1 for $1 \leq i \leq t-1$ along with (y_t, x_0). Let M be the set of matching edges (x_{j-1}, y_j) corresponding to $\mathcal{M}_2$ for $1 \leq j \leq k$. If $Y \Delta P \notin \mathcal{I}_2$, then there is an alternate matching M' from vertices in $P \backslash Y$ to $Y \backslash P$. Consider the multiset union of edges $M \cup M' \cup 2N$. Similar to the argument in parts (1) and (2a) above, we find a cycle

(say) C_1 containing the edge (y_t, x_0) such that $\ell(C_1) \leq \ell(C)$ and $|C_1| < |C|$. Deleting the edge (y_t, x_0) gives a path P_1 that contradicts the minimality of P. □

Lemma 7.6.10 allows us to compare lengths of minimal paths using only comparisons of matroid intersections:

$$\ell(P_1) - \ell(P_2) = w(Y \triangle P_2) - w(Y \triangle P_1),$$

when P_1 and P_2 are minimal paths or cycles. In order to optimize over minimal paths, we modify the Bellman-Ford algorithm in the natural way.

Modified Bellman-Ford Algorithm. Let $p_y^{(t)}$ denote the minimal path in $\mathcal{P}(y, t)$. We compute it recursively:

- **Initialization:** For each $y \in Y$,

$$p_y^{(1)} = \min_{x \in X_1}(x \to y).$$

- **Update:** For $t \geq 1$,

$$p_y^{(t+1)} = \min\left\{p_y^{(t)},\ p_z^{(t)} \to x \to y\right\},$$

  where $(z, x) \in Y \times (E \setminus Y)$ satisfies:

  1. (z, x) and (x, y) are arcs in H, and
  2. $Y \triangle p_z^{(t)} + x - y \in \mathcal{I}_1 \cap \mathcal{I}_2$.

- **Shortest Augmenting path:** After computing paths $p_y^{(t)}$ for all $y \in Y$ and $t = |Y|$, we extract a minimal shortest path P from X_1 to X_2 such that $Y \triangle P \in \mathcal{I}_1 \cap \mathcal{I}_2$, and update $Y \leftarrow Y \triangle P$.

Proof of Theorem 1.6.4. Let Y_t be the extreme common independent set of size exactly t starting with $Y_0 = \emptyset$. Use the modified Bellman-Ford Algorithm to find the shortest augmenting path P_t in the exchange graph $H(\mathcal{M}_1, \mathcal{M}_2, Y_t))$. Using Lemma 7.6.9 part (i), the exchange graph has no negative cycles. The shortest path computation takes $O(n^3)$ comparison queries and time (see Section 7.3.3). After every update, we know that $Y_{t+1} := Y_t \Delta P_t$ is an extreme set using Lemma 7.6.9 part (ii). We continue for n steps as long as there is a directed path from X_1 to X_2 in the exchange graph. Using Lemma 7.6.8, we know that the maximum cardinality intersection is reached when an augmenting path is not present anymore. Finally, we can compare the weights of all the extreme sets obtained to output the minimum weight common intersection $O(n^4)$ comparisons and $O(n^4)$ time. □

7.6.3 Omitted Content from Section 7.4

The conic dimension of the Boolean Point Sets

An important special case is that of point sets consisting of Boolean vectors in $\{0, 1\}^d$, i.e., of subsets of the hypercube. For completeness, we present the bound on the conic dimension from [113].

Lemma 7.6.11 ([113])**.** *The conic dimension of any subset of points of* $\{0,1\}^d$ *is at most* $O(d \log d)$*. In fact, it is the smallest* k *such that* $2^k > (2k+1)^d$*.*

Proof. Let us first consider the case where $\mathcal{P} = \{0,1\}^d$. For any sequence $(y_1, \ldots, y_{k+1}) \subseteq \{0,1\}^d$, consider the set

$$\left\{ \sum_{i \in [k]} \beta_i (y_{i+1} - y_i) : \beta_i \in \{0,1\} \right\}. \tag{7.18}$$

These vectors lie in the cube $\{-k, \ldots, k\}^d$, which has $(2k+1)^d$ elements. If $2^k > (2k+1)^d$, by the pigeonhole principle, two distinct $\beta \neq \gamma$ yield the same sum.

Let t_0 be the largest index where $\beta_{t_0} \neq \gamma_{t_0}$, and assume $\beta_{t_0} = 0$, $\gamma_{t_0} = 1$. Then:

$$\sum_{i \in [k]} (\beta_i - \gamma_i)(y_{i+1} - y_i) = 0. \tag{7.19}$$

Add $\sum_{i \in [t_0]} (y_{i+1} - y_i) = y_{t_0+1} - y_1$ to both sides:

$$y_{t_0+1} - y_1 = \sum_{i \in [t_0]} (\beta_i - \gamma_i + 1)(y_{i+1} - y_i) \tag{7.20}$$

$$= \sum_{i \in [t_0 - 1]} (\beta_i - \gamma_i + 1)(y_{i+1} - y_i) \tag{7.21}$$

Since $\beta_i, \gamma_i \in \{0,1\}$, the coefficients are non-negative, so $y_{t_0+1} - y_1 \in \mathrm{env}((y_1, \ldots, y_{t_0}))$. This proves that the conic dimension of the hypercube is at most $k = O(d \log d)$. Now we can use the fact the conic dimension of any subset is no larger, to infer the same property for any subset of the cube. □

Beyond the Hypercube

In this section, we consider extensions of the above claim for the Boolean hypercube: to the setting of hypergrids, and to getting ε-approximate solutions for other bounded sets.

Hypergrids and Bounded-Precision Solutions A more general version of Lemma 7.6.11 when points belong to $\{0, 1, \ldots, N-1\}^d$ shows that the conic dimension is shown to be at most $O(d \log(Nd))$. The proof is very similar to that of Lemma 7.6.11 above; see [113, Lemma 4.2].

ε-Approximate Solutions for Bounded Sets For bounded sets $\mathcal{P} \subset \mathbb{B}_d = \{x \in \mathbb{R}^d; \|x\|_2 \leq 1\}$ it is possible to bound an approximate version of the conic dimension, which leads to an algorithm that returns an ε-approximate point. Given a comparison oracle, the algorithm returns $\widehat{x} \in \mathcal{P}$ such that $\langle w^*, \widehat{x} \rangle \leq \langle w^*, x \rangle + \varepsilon$ for all $x \in \mathcal{P}$.

For the following definition, we will use the notion of the distance: given a set $S \subset \mathbb{R}^d$ and a point $x \in \mathbb{R}^d$, define $\mathrm{dist}(x, S) = \min_{y \in S} \|x - y\|_2$.

Definition 7.6.12. The *ε-approximate conic dimension* of a *set* $\mathcal{Y} \subseteq \mathbb{R}^d$ denoted by $\mathrm{ConicDim}_\varepsilon(\mathcal{Y})$ is the largest integer k for which there exists a length-k sequence $\sigma = (y_1, \ldots, y_k) \in (\mathcal{Y})^k$ of points from $\mathcal{Y}$, such that for each $t \in \{2, \ldots, k\}$ we have

$$\mathrm{dist}(y_t - y_1, \mathrm{env}(\sigma^{1:t-1})) \geq \varepsilon. \tag{7.22}$$

All the notions previously defined extend to the approximate conic dimension with the natural changes. We can extend the notion of a basic subsequence $B_\varepsilon(\sigma)$ following the same Kruskal-like procedure: we add an element to the basic subsequence if its distance to the previously spanned elements is at least ε.

If $k = \mathrm{ConicDim}_\varepsilon(\mathcal{Y})$ we can apply Algorithm 9 with two modifications: after the first iteration, always include the minimum point from the previous iteration in the sample; in step 2 eliminate all points $x \in \mathcal{P} \setminus \{y\}$ such that $\mathrm{dist}(x - y, \mathrm{env}(\sigma)) < \varepsilon$ obtaining the following results:

Corollary 7.6.13. *If $\|w^*\| \leq 1$ and $k = \mathrm{ConicDim}_\varepsilon(\mathcal{Y})$, the the variant of Algorithm 9 described above returns a point $\hat{x}$ such that $\langle w^*, \hat{x}\rangle \leq \langle w^*, x\rangle + \varepsilon$ for all $x \in \mathcal{P}$ using $O(k \log k \log |\mathcal{P}|)$ comparisons.*

Proof. Observe that if y is the smallest point in the sampled subsequence then if $\mathrm{dist}(x - y, \mathrm{env}(\sigma)) < \varepsilon$ then let $z \in y + \mathrm{env}(\sigma)$ be a point such that $\|x - z\|_2 \leq \varepsilon$. We know that

$$\langle w^*, y\rangle \leq \langle w^*, z\rangle = \langle w^*, x\rangle + \langle w^*, z - x\rangle \leq \langle w^*, x\rangle + \varepsilon$$

Hence we only eliminate points that are at no better than the point y by more than ε. The modification in step 1 guarantees that the points y selected in each iteration as increasingly better. The remainder of the proof works without any change. □

Finally we bound the approximate conic dimension:

Lemma 7.6.14. *The ε-approximate conic dimension of any set of points $\mathcal{P} \subset \mathbb{B}_d$ is the smallest k such that $2^k(\varepsilon/2)^d > (2k+1)^d$.*

Proof. The proof follows by replacing the pigeonhole argument in Lemma 7.6.11 by a volumetric argument. We again consider the set:

$$\left\{ \sum_{i \in [k]} \beta_i (y_{i+1} - y_i) : \beta_i \in \{0, 1\} \right\} \subseteq 2k \cdot \mathbb{B}_d \tag{7.23}$$

i.e, the ball of radius $2k$ in $\mathbb{R}^d$, which has volume $(2k)^d \cdot \mathrm{Vol}(\mathbb{B}_d)$. Now, for each vector $\beta \in \{0, 1\}^k$, consider the ball of radius $\varepsilon/2$ around $\sum_{i \in [k]} \beta_i (y_{i+1} - y_i)$, which have total volume $2^k(\varepsilon/2)^d \cdot \mathrm{Vol}(\mathbb{B}_d)$. Each of those balls is contained in the ball of radius $2k + 1$ around zero. If $2^k(\varepsilon/2)^d > (2k+1)^d$ then two of the smaller balls must intersect, so there exist vectors $\beta, \gamma \in \{0, 1\}^k$ such that:

$$\sum_{i \in [k]} (\beta_i - \gamma_i)(y_{i+1} - y_i) = z, \tag{7.24}$$

with $\|z\|_2 < \epsilon$. Let t_0 be the largest index where $\beta_{t_0} \neq \gamma_{t_0}$, and assume $\beta_{t_0} = 0$, $\gamma_{t_0} = 1$. Doing the same manipulations as in Lemma 7.6.11 we obtain:

$$z + y_{t_0+1} - y_1 = \sum_{i \in [t_0-1]} (\beta_i - \gamma_i + 1)(y_{i+1} - y_i) \tag{7.25}$$

Since $\beta_i, \gamma_i \in \{0, 1\}$, the coefficients are non-negative, so

$$\operatorname{dist}(y_{t_0+1} - y_1, \operatorname{env}(y_1, \ldots, y_{t_0})) \leq \|z\|_2 < \varepsilon$$

which shows that that ε-approximate conic dimension is at most k. □

The Certification Problem

Before addressing the optimization problem, we ask whether it is even possible to certify optimality efficiently. Suppose we wish to prove that some point $y \in \mathcal{P}$ is the optimal solution, i.e., $y = \arg\min_{x \in \mathcal{P}} \langle w^*, x \rangle$, or equivalently, that

$$\langle w^*, x - y \rangle \geq 0 \quad \forall x \in \mathcal{P}. \tag{7.26}$$

We want to use as few comparisons as possible.

1. A *y-certificate* is a collection $\mathcal{C}$ of index pairs (i, j) such that for all $w \in \mathbb{R}^d$, we have that
$$(\langle w, x_i - x_j \rangle \geq 0 \quad \forall (i, j) \in \mathcal{C}) \Rightarrow (\langle w, x - y \rangle \geq 0 \quad \forall x \in \mathcal{P}) \tag{7.27}$$
2. A certificate $\mathcal{C}$ is *valid* if $\langle w^*, x_i - x_j \rangle \geq 0$ for all $(i, j) \in \mathcal{C}$.

Hence a valid y-certificate implies that y is a minimizer of $\langle w^*, x \rangle$ among points in $\mathcal{P}$. In fact, the minimum number of queries needed to find an optimal solution is lower bounded by the minimum size of a valid certificate.

Recall that given a set $V \subseteq \mathbb{R}^d$ of vectors, the cone generated by V is the set of all vectors that can be written as positive linear combinations of vectors from V; this is denoted by $\operatorname{cone}(V)$. Formally,

$$\operatorname{cone}(V) = \left\{ x \in \mathbb{R}^d : x = \sum_{v \in V} \alpha_v \, v, \ \alpha_v \geq 0 \right\}.$$

The following lemma gives a characterization of y-certificates based only on the geometry of the point set $\mathcal{P}$.

Lemma 7.6.15. *A set $\mathcal{C} \subseteq [N] \times [N]$ is a y-certificate iff $\mathcal{P} \subseteq y + \operatorname{cone}\left(\{x_i - x_j\}_{(i,j) \in \mathcal{C}}\right)$.*

Proof. Suppose $\mathcal{C}$ is a y-certificate. Then, the definition implies that the system:

$$\begin{aligned} &\langle w, x_i - x_j \rangle \geq 0 \quad \forall (i, j) \in \mathcal{C} \\ &\langle w, x - y \rangle < 0 \end{aligned} \tag{7.28}$$

is infeasible for any $x \in \mathcal{P}$. By Farkas' lemma, this implies that $x - y \in \operatorname{cone}\left(\{x_i - x_j\}_{(i,j) \in \mathcal{C}}\right)$.

Conversely, if $x - y \in \operatorname{cone}\left(\{x_i - x_j\}_{(i,j) \in \mathcal{C}}\right)$ and $\langle w, x_i - x_j \rangle \geq 0$ for all $(i, j) \in \mathcal{C}$, then $\langle w, x - y \rangle \geq 0$ follows immediately. □

Definition 7.6.16. For a sequence $\sigma = (x_{i_1}, x_{i_2}, \ldots, x_{i_k})$, the *induced certificate* is defined as $\mathcal{C}_\sigma = \{(i_\ell, i_{\ell+1}) : 1 \le \ell \le k-1\}$

Basic Subsequences

For a set $\mathcal{Y} \subseteq \mathbb{R}^d$ with $\mathrm{ConicDim}(\mathcal{Y}) = k$ and any sequence $\sigma = (y_1, \ldots, y_n) \in \mathcal{Y}^n$, consider a process that maintains a subsequence π_t of σ for each $0 \le t \le n$. We start with the empty sequence $\pi_0 = \langle\rangle$, and at step $t+1$, we update

$$\pi_{t+1} = \begin{cases} \pi_t \circ y_{t+1}, & \text{if } y_{t+1} - y_1 \notin \mathrm{env}(\pi_t) \\ \pi_t, & \text{otherwise.} \end{cases}$$

An analogy can be made to Kruskal's algorithm, where we add the next element in the sequence precisely when it is not conically spanned by the envelope of the current set. We observe:

1. the sequence π_t is a prefix of π_{t+1}, and moreover, π_{t+1} contains at most one more element.
2. $|\pi_n| \le k$.

The first observation is immediate from the construction, and the second follows by contradiction: if $|\pi_n| > k$, then some subsequence of size $k+1$ contradicts the definition of conic dimension. Let $B(\sigma) := \pi_n$ be the subsequence of σ obtained by this process; we call this the "basis" of σ.

Lemma 7.6.17. *Given a sequence $\sigma = (y_1, \ldots, y_n) \in \mathcal{Y}^n$, we have:*

$$\mathrm{cone}(\{y_j - y_1\}_{1 \le j \le n}) \subseteq \mathrm{env}(B(\sigma)) \subseteq \mathrm{env}(\sigma). \tag{7.29}$$

Proof. For the rightmost inclusion, suppose $B(\sigma) = (y_{i_1}, \ldots, y_{i_k})$. Then,

$$\mathrm{env}(B(\sigma)) = \mathrm{cone}(\{y_{i_{\ell+1}} - y_{i_\ell}\}_{1 \le \ell \le k-1}) \subseteq \mathrm{cone}(\{y_j - y_i\}_{1 \le i \le j \le n}) = \mathrm{env}(\sigma). \tag{7.30}$$

For the leftmost inclusion, let $\pi_t := B(\sigma^{1:t})$ denote the basis of the prefix at time t. Consider any $y_t \in \sigma$:

- If $y_t \notin \pi_n = B(\sigma)$, then it was excluded by the base construction algorithm, meaning:

$$y_t - y_1 \in \mathrm{env}(\pi_{t-1}) \subseteq \mathrm{env}(\pi_n) = \mathrm{env}(B(\sigma)).$$

- If $y_t \in \pi_n$, then it is one of the elements of the basis, and we can write:

$$y_t - y_1 = \sum_{\ell : i_\ell \le t} (y_{i_{\ell+1}} - y_{i_\ell}) \in \mathrm{cone}(\{y_{i_{\ell+1}} - y_{i_\ell}\}_{1 \le \ell \le k-1}) = \mathrm{env}(B(\sigma)).$$

Hence, in both cases, $y_t - y_1 \in \mathrm{env}(B(\sigma))$, completing the proof. □

We can now relate the size of valid x^*-certificates to the conic dimension of the underlying point set.

Lemma 7.6.18. *If some point set $\mathcal{Y} \subseteq \mathbb{R}^d$ has conic dimension k, and $y^* = \arg\min_{y \in \mathcal{Y}} \langle w^*, y \rangle$, then there exists a valid y^*-certificate $\mathcal{C}$ of length $|\mathcal{C}| \leq k - 1$.*

Proof. Order the points in $\mathcal{Y}$ as the sequence $\sigma = \langle y^* = y_1, y_2, \ldots, y_n \rangle$ such that $\langle w^*, y_i \rangle \leq \langle w^*, y_{i+1} \rangle$ for all i, and let $B(\sigma) = (y_{i_1}, \ldots, y_{i_k})$ be the basic subsequence corresponding to σ. We claim that the induced certificate $\mathcal{C} := \mathcal{C}_{B(\sigma)}$ is a valid y^*-certificate. Notice that $\mathcal{C}$ is defined such that

$$\mathrm{env}(B(\sigma)) = \mathrm{cone}(\{y_i - y_j\}_{(i,j) \in \mathcal{C}}).$$

Using Lemma 7.6.17, we have $\mathcal{Y} \subseteq y_1 + \mathrm{cone}(\{y_i - y_j\}_{(i,j) \in \mathcal{C}})$, which means that $\mathcal{C}$ is a valid certificate using Lemma 7.6.15. Moreover, by construction, $(i, j) \in C$ means $\langle w^*, y_j - y_i \rangle \geq 0$, which in turn means that the certificate is valid. □

Proof of Lemma 7.4.4

Proof. For the analysis, consider a sequence $(x_1, \ldots, x_N)$ be such that $\langle w^*, x_i \rangle \leq \langle w^*, x_{i+1} \rangle$ for all $1 \leq i \leq N - 1$. Note that σ is distributed as a random subsequence of $(x_1, \ldots, x_N)$, where each element is selected independently with probability $2k/N$.

Consider a slightly modified, more lenient version of the algorithm that eliminates a point $x_t \in \mathcal{P} \setminus \{y\}$ in Step 3 only if $x_t - y \in \mathrm{env}(B(\sigma_{t-1}))$ where σ_t is the prefix of σ restricted to elements $\{x_i\}_{1 \leq i \leq t}$; recall the definition of the basic subsequence $B(\cdot)$ from Section 7.6.3. This modification ensures that elimination of x_t depends only on the coin tosses at indices in $[t]$ and the sequence $(x_1, \ldots, x_t)$. This modified algorithm eliminates no more points than the original algorithm because $\mathrm{env}(B(\sigma_{t-1})) \subseteq \mathrm{env}(\sigma_{t-1}) \subseteq \mathrm{env}(\sigma)$—each inequality just uses that the cone of a smaller set of vertices is smaller. Hence, it suffices to prove that the expected number of points eliminated by the modified algorithm is at least $N/2$.

We now use the principle of deferred randomness to analyze this version of the algorithm. Let E_t be the "evolving set" of elements in $\{x_i\}_{1 \leq i \leq t}$ that are not eliminated by the modified algorithm after it has seen the elements $x_1, x_2 \ldots, x_t$ in that order. Remember that σ_t is the subsequence of $(x_1, \ldots, x_t)$ corresponding to elements that are sampled. We can simulate the process of generating the sets E_t and σ_t in the following way:

1. Start with $E_0, \sigma_0 = \langle\rangle$.
2. At step $1 \leq t \leq N$, toss a biased coin with success probability $2k/N$. Update

$$\sigma_t = \begin{cases} \sigma_{t-1} \circ x_t, & \text{if coin flip succeeds at t} \\ \sigma_{t-1}, & \text{otherwise.} \end{cases}$$

and

$$E_t = \begin{cases} E_{t-1} \circ x_t, & \text{if } x_t - y \notin \mathrm{env}(B(\sigma_{t-1})) \\ E_{t-1}, & \text{otherwise.} \end{cases}$$

where y is the first element sampled.

Observing that

$$\begin{aligned}|B(\sigma_t)| - |B(\sigma_{t-1})| &= \mathbb{1}[x_t - y \notin \mathrm{env}(B(\sigma_{t-1})) \wedge \text{coin flip succeds at } t] \\ &= (|E_t| - |E_{t-1}|) \cdot \mathbb{1}[\text{coin flip succeds at } t].\end{aligned}$$

Taking expectations on both sides gives

$$\mathbb{E}[|E_t| - |E_{t-1}|] = \frac{N}{2k} \cdot \mathbb{E}[|B(\sigma_t)| - |B(\sigma_{t-1})|].$$

Summing this over $1 \leq t \leq N$ gives

$$\mathbb{E}[|E_N|] = \frac{N}{2k} \cdot \mathbb{E}[|B(\sigma)|] \leq \frac{N}{2k} \cdot k = \frac{N}{2}. \qquad \square$$

Bibliography

[1] Scott Aaronson and Alex Arkhipov. The computational complexity of linear optics. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*, STOC '11, page 333–342, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450306911. doi: 10.1145/1993636.1993682. URL https://doi.org/10.1145/1993636.1993682. 3.1

[2] Deeksha Adil, Rasmus Kyng, Richard Peng, and Sushant Sachdeva. Fast algorithms for ℓ_p-regression. *J. ACM*, 71(5), October 2024. ISSN 0004-5411. doi: 10.1145/3686794. URL https://doi.org/10.1145/3686794. 5.4.2

[3] Zeyuan Allen-Zhu, Yuanzhi Li, Aarti Singh, and Yining Wang. Near-optimal discrete optimization for experimental design: A regret minimization approach. *arXiv preprint arXiv:1711.05174*, 2017. 1.2, 2.1, 2.1, 2.1.4

[4] Jason Altschuler, Aditya Bhaskara, Gang Fu, Vahab Mirrokni, Afshin Rostamizadeh, and Morteza Zadimoghaddam. Greedy column subset selection: New bounds and distributed algorithms. In *International Conference on Machine Learning*, pages 2539–2548, 2016. 5.1, 5.1.1

[5] Nima Anari and Shayan Oveis Gharan. A generalization of permanent inequalities and applications in counting and optimization. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 384–396. ACM, 2017. 1.2, 1.2.1, 2.1, 2.1, 2.1.2, 2.1.4

[6] Nima Anari, Shayan Oveis Gharan, Amin Saberi, and Mohit Singh. Nash social welfare, matrix permanent, and stable polynomials. *In Proceedings of Conference on Innovations in Theoretical Computer Science*, 2016. 1.2, 2.1, 4, 2.1, 2.1.4

[7] Nima Anari, Shayan Oveis Gharan, Amin Saberi, and Mohit Singh. Nash Social Welfare, Matrix Permanent, and Stable Polynomials. In Christos H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, volume 67 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 36:1–36:12, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-95977-029-3. doi: 10.4230/LIPIcs.ITCS.2017.36. URL http://drops.dagstuhl.de/opus/volltexte/2017/8148. 1.3.1, 4.1.2, 4.1.2, 4.1.2, 4.1.2, 4.1.3, 4.2, 4.6.2, 4.6.3, 4.6.2

[8] Nima Anari, Shayan Oveis Gharan, and Cynthia Vinzant. Log-concave polynomials, entropy, and a deterministic approximation algorithm for counting bases of matroids. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 35–46. IEEE, 2018. 1.2, 1.2.1, 2.1, 2.1, 2.1.2

[9] Nima Anari, Tung Mai, Shayan Oveis Gharan, and Vijay V Vazirani. Nash social welfare for indivisible items under separable, piecewise-linear concave utilities. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2274–2290. SIAM, 2018. 1.3.1, 2.1.4

[10] Nima Anari, Kuikui Liu, Shayan Oveis Gharan, and Cynthia Vinzant. Log-concave polynomials ii: high-dimensional walks and an fpras for counting bases of a matroid. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 1–12. ACM, 2019. 2.1.4

[11] Simon Apers, Yuval Efron, Pawel Gawrychowski, Troy Lee, Sagnik Mukhopadhyay, and Danupon Nanongkai. Cut Query Algorithms with Star Contraction . In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 507–518, Los Alamitos, CA, USA, November 2022. IEEE Computer Society. doi: 10.1109/FOCS54457.2022.00055. URL `https://doi.ieeecomputersociety.org/10.1109/FOCS54457.2022.00055`. 7.1, 7.1.2

[12] Sepehr Assadi and Aditi Dudeja. A simple semi-streaming algorithm for global minimum cuts. In Hung Viet Le and Valerie King, editors, *4th Symposium on Simplicity in Algorithms, SOSA 2021, Virtual Conference, January 11-12, 2021*, pages 172–180. SIAM, 2021. doi: 10.1137/1.9781611976496.19. URL `https://doi.org/10.1137/1.9781611976496.19`. 7.1.2

[13] Megasthenis Asteris, Dimitris Papailiopoulos, and Alexandros Dimakis. Nonnegative sparse pca with provable guarantees. In *International Conference on Machine Learning*, pages 1728–1736. PMLR, 2014. 5.1

[14] Arinta Auza and Troy Lee. On the query complexity of connectivity with global queries. *arXiv preprint arXiv:2109.02115*, 2021. 7.1.2

[15] Maria-Florina Balcan, Ellen Vitercik, and Colin White. Learning combinatorial functions from pairwise comparisons. In *Conference on Learning Theory*, pages 310–335. PMLR, 2016. 7.1.2

[16] Frank Ban, Vijay Bhattiprolu, Karl Bringmann, Pavel Kolev, Euiwoong Lee, and David P. Woodruff. A PTAS for ℓ_p-low rank approximation. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 747–766. SIAM, 2019. 5.1.1

[17] Frank Ban, David P. Woodruff, and Qiuyi (Richard) Zhang. Regularized weighted low rank approximation. *CoRR*, abs/1911.06958, 2019. 5.1.1

[18] Sayan Bandyapadhyay, Fedor V. Fomin, Petr A. Golovach, William Lochet, Nidhi Purohit, and Kirill Simonov. How to find a good explanation for clustering? In *Thirty-Sixth*

AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022, pages 3904–3912. AAAI Press, 2022. URL https://ojs.aaai.org/index.php/AAAI/article/view/20306. 1.5.1, 6.1, 6.1.1, 6.1.2

[19] Julius B. Barbanel and Alan D. Taylor. *The Geometry of Efficient Fair Division*. Cambridge University Press, 2005. doi: 10.1017/CBO9780511546679. 1.3, 4.1

[20] Siddharth Barman, Sanath Kumar Krishnamurthy, and Rohit Vaish. Finding fair and efficient allocations. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, pages 557–574. ACM, 2018. 1.3, 2.1.4, 4.1, 4.1.3

[21] Siddharth Barman, Sanath Kumar Krishnamurthy, and Rohit Vaish. Greedy algorithms for maximizing nash social welfare. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 7–13. International Foundation for Autonomous Agents and Multiagent Systems, 2018. 2.1.4

[22] William Barnett. The modern theory of consumer behavior: Ordinal or cardinal? *The Quarterly Journal of Austrian Economics*, 6(1):41–65, 2003. 7.1

[23] Saugata Basu, Richard Pollack, and Marie-Françoise Roy. On the combinatorial and algebraic complexity of quantifier elimination. *J. ACM*, 43(6):1002–1045, 1996. 5.4.8

[24] Luca Becchetti, Marc Bury, Vincent Cohen-Addad, Fabrizio Grandoni, and Chris Schwiegelshohn. Oblivious dimension reduction for *k*-means: beyond subspaces and the Johnson-Lindenstrauss lemma. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 1039–1050. ACM, 2019. doi: 10.1145/3313276.3316318. URL https://doi.org/10.1145/3313276.3316318. 1

[25] Aditya Bhaskara, Sepideh Mahabadi, Madhusudhan Reddy Pittu, Ali Vakilian, and David P. Woodruff. Guessing Efficiently for Constrained Subspace Approximation. In Keren Censor-Hillel, Fabrizio Grandoni, Joël Ouaknine, and Gabriele Puppis, editors, *52nd International Colloquium on Automata, Languages, and Programming (ICALP 2025)*, volume 334 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 29:1–29:20, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISBN 978-3-95977-372-0. doi: 10.4230/LIPIcs.ICALP.2025.29. URL https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ICALP.2025.29. 4, 1.4.1, 1.8

[26] Christos Boutsidis, Michael W Mahoney, and Petros Drineas. An improved approximation algorithm for the column subset selection problem. In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, pages 968–977, 2009. 5.1

[27] Christos Boutsidis, Petros Drineas, and Malik Magdon-Ismail. Sparse features for pca-like linear regression. *Advances in Neural Information Processing Systems*, 24, 2011. 5.1.1

[28] Christos Boutsidis, Petros Drineas, and Malik Magdon-Ismail. Near-optimal column-based matrix reconstruction. *SIAM Journal on Computing*, 43(2):687–717, 2014. 5.1

[29] Christos Boutsidis, Anastasios Zouzias, Michael W Mahoney, and Petros Drineas. Randomized dimensionality reduction for k-means clustering. *IEEE Transactions on Information Theory*, 61(2):1045–1062, 2014. 5.1, 5.1.1

[30] Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952. 7.1

[31] Steven J. Brams and Alan D. Taylor. *Fair Division: From Cake-Cutting to Dispute Resolution*. Cambridge University Press, 1996. doi: 10.1017/CBO9780511598975. 1.3, 4.1

[32] Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D.Editors Procaccia. *Handbook of Computational Social Choice*. Cambridge University Press, 2016. doi: 10.1017/CBO9781107446984. 1.3, 4.1

[33] L. M. Bregman. Some properties of nonnegative matrices and their permanents. *Sov. Math., Dokl.*, 14:945–949, 1973. ISSN 0197-6788. 3.1

[34] Adam Brown, Aditi Laddha, Madhusudhan Reddy Pittu, and Mohit Singh. *Approximation Algorithms for the Weighted Nash Social Welfare via Convex and Non-Convex Programs*, pages 1307–1327. doi: 10.1137/1.9781611977912.52. URL `https://epubs.siam.org/doi/abs/10.1137/1.9781611977912.52`. 2, 1.8

[35] Adam Brown, Aditi Laddha, Madhusudhan Pittu, and Mohit Singh. Efficient determinant maximization for all matroids, 2022. URL `https://arxiv.org/abs/2211.10507`. 1, 1.8

[36] Adam Brown, Aditi Laddha, Madhusudhan Pittu, Mohit Singh, and Prasad Tetali. Determinant maximization via matroid intersection algorithms. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 255–266, 2022. doi: 10.1109/FOCS54457.2022.00031. 1, 1.8

[37] Nader H Bshouty and Hanna Mazzawi. On parity check (0, 1)-matrix over $\mathbb{Z}_p$. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1383–1394. SIAM, 2011. 7.1.2

[38] Niv Buchbinder, Joseph (Seffi) Naor, and Roy Schwartz. Simplex partitioning via exponential clocks and the multiway-cut problem. *SIAM J. Comput.*, 47(4):1463–1482, 2018. doi: 10.1137/15M1045521. URL `https://doi.org/10.1137/15M1045521`. 6.1.1

[39] Jaroslaw Byrka, Thomas W. Pensyl, Bartosz Rybicki, Aravind Srinivasan, and Khoa Trinh. An improved approximation for k-median and positive correlation in budgeted optimization. *ACM Trans. Algorithms*, 13(2):23:1–23:31, 2017. doi: 10.1145/2981561. URL `https://doi.org/10.1145/2981561`. 6.1.2

[40] Jorge Cadima and Ian T Jolliffe. Loading and correlations in the interpretation of principle compenents. *Journal of applied Statistics*, 22(2):203–214, 1995. 5.1.1

[41] Ioannis Caragiannis, David Kurokawa, Hervé Moulin, Ariel D. Procaccia, Nisarg Shah, and Junxing Wang. The unreasonable fairness of maximum nash welfare. In *Proceedings of the 2016 ACM Conference on Economics and Computation*, EC '16, pages 305–322, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-3936-0. doi: 10.1145/2940716.2940726. URL http://doi.acm.org/10.1145/2940716.2940726. 2.1.4

[42] Ioannis Caragiannis, David Kurokawa, Hervé Moulin, Ariel D Procaccia, Nisarg Shah, and Junxing Wang. The unreasonable fairness of maximum nash welfare. *ACM Transactions on Economics and Computation (TEAC)*, 7(3):1–32, 2019. 1.3, 4.1

[43] Suchan Chae and Hervé Moulin. Bargaining among groups: An axiomatic viewpoint. *International Journal of Game Theory*, 39, 02 2004. doi: 10.1007/s00182-009-0157-6. 1.3, 4.1, 4.1

[44] Mithun Chakraborty, Ayumi Igarashi, Warut Suksompong, and Yair Zick. Weighted envy-freeness in indivisible item allocation. *ACM Trans. Econ. Comput.*, 9(3), August 2021. ISSN 2167-8375. doi: 10.1145/3457166. URL https://doi.org/10.1145/3457166. 4.1

[45] Moses Charikar and Lunjia Hu. Near-optimal explainable k-means for all dimensions. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 2580–2606. SIAM, 2022. doi: 10.1137/1.9781611977073.101. URL https://doi.org/10.1137/1.9781611977073.101. 1.5.1, 1.5.1, 6.1, 6.1, 6.1.2

[46] Xi Chen, Paul N Bennett, Kevyn Collins-Thompson, and Eric Horvitz. Pairwise ranking aggregation in a crowdsourced setting. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 193–202, 2013. 7.1

[47] Ashish Chiplunkar, Sagar Kale, and Sivaramakrishnan Natarajan Ramamoorthy. How to solve fair k-center in massive data models. In *International Conference on Machine Learning*, pages 1877–1886, 2020. 5.1.1

[48] Sung-Soon Choi. Polynomial time optimal query algorithms for finding graphs with arbitrary real weights. In *Conference on Learning Theory*, pages 797–818. PMLR, 2013. 7.1.2

[49] Ali Civril and Malik Magdon-Ismail. Column subset selection via sparse approximation of SVD. *Theoretical Computer Science*, 421:1–14, 2012. 5.1

[50] Kenneth L. Clarkson and David P. Woodruff. Numerical linear algebra in the streaming model. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, page 205–214, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585062. doi: 10.1145/1536414.1536445. URL https://doi.org/10.1145/1536414.1536445. 5.3.8

[51] Kenneth L. Clarkson and David P. Woodruff. Input sparsity and hardness for robust subspace approximation. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 310–329, 2015. doi: 10.1109/FOCS.2015.27. 5.1, 5.2

[52] Kenneth L Clarkson and David P Woodruff. Low-rank approximation and regression in input sparsity time. *Journal of the ACM (JACM)*, 63(6):1–45, 2017. 5.4.4, 5.4.4

[53] Michael B Cohen, Sam Elder, Cameron Musco, Christopher Musco, and Madalina Persu. Dimensionality reduction for k-means clustering and low rank approximation. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 163–172, 2015. 5.1, 5.1, 5.1.1, 5.1.1, 1, 5.4.4

[54] Vincent Cohen-Addad, Philip N. Klein, and Claire Mathieu. Local search yields approximation schemes for k-means and k-median in euclidean and minor-free metrics. In Irit Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 353–364. IEEE Computer Society, 2016. doi: 10.1109/FOCS.2016.46. URL https://doi.org/10.1109/FOCS.2016.46. 6.1.2

[55] Vincent Cohen-Addad, David Saulpic, and Chris Schwiegelshohn. A new coreset framework for clustering. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, page 169–182, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450380539. doi: 10.1145/3406325.3451022. URL https://doi.org/10.1145/3406325.3451022. 5.1.1

[56] Vincent Cohen-Addad, Hossein Esfandiari, Vahab S. Mirrokni, and Shyam Narayanan. Improved approximations for euclidean *k*-means and *k*-median, via nested quasi-independent sets. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 1621–1628. ACM, 2022. doi: 10.1145/3519935.3520011. URL https://doi.org/10.1145/3519935.3520011. 6.1.2

[57] Vincent Cohen-Addad, Karthik C. S., and Euiwoong Lee. Johnson coverage hypothesis: Inapproximability of k-means and k-median in ℓ_p-metrics. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 1493–1530. SIAM, 2022. doi: 10.1137/1.9781611977073.63. URL https://doi.org/10.1137/1.9781611977073.63. 6.1.2

[58] Vincent Cohen-Addad, Fabrizio Grandoni, Euiwoong Lee, and Chris Schwiegelshohn. Breaching the 2 LMP approximation barrier for facility location with applications to *k*-median. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 940–986. SIAM, 2023. doi: 10.1137/1.9781611977554.ch37. URL https://doi.org/10.1137/1.9781611977554.ch37. 6.1.2

[59] Vincent Cohen-Addad, Kasper Green Larsen, David Saulpic, Chris Schwiegelshohn, and Omar Ali Sheikh-Omar. Improved coresets for euclidean k-means. In *Proceedings of the*

36th International Conference on Neural Information Processing Systems. Curran Associates Inc., 2024. ISBN 9781713871088. 5.1.1, 1b

[60] Richard Cole and Vasilis Gkatzelis. Approximating the nash social welfare with indivisible items. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 371–380, 2015. 1.3.1, 1.3.1, 4, 2.1.4, 4.1.2, 4.1.2, 4.1.2, 4.1.2, 4.1.3, 4.3

[61] Richard Cole, Nikhil Devanur, Vasilis Gkatzelis, Kamal Jain, Tung Mai, Vijay V Vazirani, and Sadra Yazdanbod. Convex program duality, fisher markets, and nash social welfare. In *Proceedings of the 2017 ACM Conference on Economics and Computation*, pages 459–460, 2017. 1.3.1, 1.3.1, 2.1.4, 4.1.2, 4.1.2, 4.1.2, 4.1.2, 4.1.3, 4.3

[62] Sanjoy Dasgupta, Nave Frost, Michal Moshkovitz, and Cyrus Rashtchian. Explainable k-means and k-medians clustering. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 7055–7065. PMLR, 2020. URL `http://proceedings.mlr.press/v119/moshkovitz20a.html`. (document), 1.5, 1.1, 1.5.1, 1.5.1, 6.1, 6.1, 6.1, 6.1, 6.1.1

[63] Sanjoy Dasgupta, Nave Frost, and Michal Moshkovitz. Framework for evaluating faithfulness of local explanations. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 4794–4815. PMLR, 2022. URL `https://proceedings.mlr.press/v162/dasgupta22a.html`. 6.1

[64] Damek Davis and Dmitriy Drusvyatskiy. Stochastic model-based minimization of weakly convex functions. *SIAM Journal on Optimization*, 29(1):207–239, 2019. doi: 10.1137/18M1178244. URL `https://doi.org/10.1137/18M1178244`. 4.6.3

[65] Alberto Del Pia. Sparse pca on fixed-rank matrices. *Mathematical Programming*, 198(1): 139–157, 2023. doi: 10.1007/s10107-022-01769-9. URL `https://doi.org/10.1007/s10107-022-01769-9`. 5.1, 5.1.1, 5.4.5, 2, 5.4.5, 5.4.18

[66] Amit Deshpande and Luis Rademacher. Efficient volume sampling for row/column subset selection. In *2010 ieee 51st annual symposium on foundations of computer science*, pages 329–338, 2010. 5.1

[67] Amit Deshpande, Madhur Tulsiani, and Nisheeth K. Vishnoi. Algorithms and hardness for subspace approximation. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, page 482–496, USA, 2011. Society for Industrial and Applied Mathematics. 5.1

[68] Michel Marie Deza and Monique Laurent. *Geometry of cuts and metrics*, volume 15 of *Algorithms and Combinatorics*. Springer-Verlag, Berlin, 1997. ISBN 3-540-61611-X. 6.2.2, 6.8.1

[69] Irit Dinur and David Steurer. Analytical approach to parallel repetition. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 624–633. ACM, 2014. 6.3

[70] Petros Drineas, Alan Frieze, Ravi Kannan, Santosh Vempala, and Vishwanathan Vinay. Clustering large graphs via the singular value decomposition. *Machine learning*, 56:9–33, 2004. 5.1

[71] Javad B Ebrahimi, Damian Straszak, and Nisheeth K Vishnoi. Subdeterminant maximization via nonconvex relaxations and anti-concentration. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1020–1031. Ieee, 2017. 2.1.4

[72] Hossein Esfandiari, MohammadTaghi Hajiaghayi, and David P. Woodruff. Applications of uniform sampling: Densest subgraph and beyond, 2015. URL `https://arxiv.org/abs/1506.04505`. 7.1.1, 7.2.4

[73] Hossein Esfandiari, Vahab S. Mirrokni, and Shyam Narayanan. Almost tight approximation algorithms for explainable clustering. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 2641–2663. SIAM, 2022. doi: 10.1137/1.9781611977073.103. URL `https://doi.org/10.1137/1.9781611977073.103`. 1.5.1, 1.5.1, 6.1, 6.1, 6.1, 6.1.1, 6.1.2

[74] Uriel Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998. 6.1.1, 6.3, 2

[75] Uriel Feige, Prabhakar Raghavan, David Peleg, and Eli Upfal. Computing with noisy information. *SIAM Journal on Computing*, 23(5):1001–1018, 1994. 7.5

[76] Dan Feldman, Morteza Monemizadeh, and Christian Sohler. A ptas for k-means clustering based on weak coresets. In *Proceedings of the twenty-third annual symposium on Computational geometry*, pages 11–18, 2007. 5.1.1, 5.1, 5.1.1

[77] Yuda Feng and Shi Li. A note on approximating weighted nash social welfare with additive valuations, 2024. URL `https://arxiv.org/abs/2404.15607`. 4.1.3, 4.5

[78] Yuda Feng, Yang Hu, Shi Li, and Ruilong Zhang. Constant approximation for weighted nash social welfare with submodular valuations, 2024. URL `https://arxiv.org/abs/2411.02942`. 4.1.3, 4.5

[79] Dean Foster, Howard Karloff, and Justin Thaler. Variable selection is hard. In *Proceedings of The 28th Conference on Learning Theory*, volume 40 of *Proceedings of Machine Learning Research*, pages 696–709. PMLR, 2015. 5.5, 5.5.3

[80] Zachary Friggstad, Mohsen Rezapour, and Mohammad R. Salavatipour. Local search yields a PTAS for k-means in doubling metrics. In Irit Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016,*

Hyatt Regency, New Brunswick, New Jersey, USA, pages 365–374. IEEE Computer Society, 2016. doi: 10.1109/FOCS.2016.47. URL https://doi.org/10.1109/FOCS.2016.47. 6.1.2

[81] Nave Frost, Michal Moshkovitz, and Cyrus Rashtchian. Exkmc: Expanding explainable k-means clustering. *CoRR*, abs/2006.02399, 2020. URL https://arxiv.org/abs/2006.02399. 6.1.2

[82] Hu Fu, Robert Kleinberg, and Ron Lavi. Conditional equilibrium outcomes via ascending price processes with applications to combinatorial auctions with item bidding. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, EC '12, page 586, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450314152. doi: 10.1145/2229012.2229055. URL https://doi.org/10.1145/2229012.2229055. 1.3, 4.1

[83] Buddhima Gamlath, Xinrui Jia, Adam Polak, and Ola Svensson. Nearly-tight and oblivious algorithms for explainable clustering. pages 28929–28939, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/f24ad6f72d6cc4cb51464f2b29ab69d3-Abstract.html. 1.5.1, 1.5.1, 6.1, 6.1, 6.1, 6.4.2

[84] Jugal Garg, Martin Hoefer, and Kurt Mehlhorn. Approximating the nash social welfare with budget-additive valuations. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2326–2340. SIAM, 2018. 2.1.4

[85] Jugal Garg, Pooja Kulkarni, and Rucha Kulkarni. Approximating nash social welfare under submodular valuations through (un) matchings. In *Proceedings of the fourteenth annual ACM-SIAM symposium on discrete algorithms*, pages 2673–2687. SIAM, 2020. 1.3, 4.1

[86] Jugal Garg, Edin Husić, and László A Végh. Approximating nash social welfare under rado valuations. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1412–1425, 2021. 1.3, 2.1.4, 4.1, 4.1.3

[87] Jugal Garg, Edin Husić, Wenzheng Li, László A Végh, and Jan Vondrák. Approximating nash social welfare by matching and local search. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 1298–1310, 2023. 1.3, 1.3.1, 4.1, 4.1.3

[88] Dongdong Ge, Simai He, Yinyu Ye, and Jiawei Zhang. Geometric rounding: a dependent randomized rounding scheme. *J. Comb. Optim.*, 22(4):699–725, 2011. doi: 10.1007/s10878-010-9320-z. URL https://doi.org/10.1007/s10878-010-9320-z. 6.1.1

[89] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, 4th edition, 2013. ISBN 978-1-4214-0794-4. 5.2.1, 5.2.2

[90] Andrei Graur, Tristan Pollner, Vidhya Ramaswamy, and S. Matthew Weinberg. New query lower bounds for submodular function minimization. In Thomas Vidick, editor,

11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA, volume 151 of *LIPIcs*, pages 64:1–64:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi: 10.4230/LIPICS.ITCS.2020.64. URL https://doi.org/10.4230/LIPIcs.ITCS.2020.64. 7.1.2

[91] Vladimir Grebinski and Gregory Kucherov. Optimal reconstruction of graphs under the additive model. *Algorithmica*, 28(1):104–124, 2000. doi: 10.1007/S004530010033. URL https://doi.org/10.1007/s004530010033. 7.1.2

[92] Anupam Gupta, Madhusudhan Reddy Pittu, Ola Svensson, and Rachel Yuan. The price of explainability for clustering. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1131–1148, 2023. doi: 10.1109/FOCS57990.2023.00067. 3, 1.8

[93] Aparna Gupte and Vinod Vaikuntanathan. The fine-grained hardness of sparse linear regression. *arXiv preprint arXiv:2106.03131*, 2021. 5.5

[94] Venkatesan Guruswami and Ali Kemal Sinop. Optimal column-based low-rank matrix reconstruction. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1207–1214, 2012. 5.1

[95] Venkatesan Guruswami, Prasad Raghavendra, Rishi Saket, and Yi Wu. Bypassing ugc from some optimal geometric inapproximability results. *ACM Trans. Algorithms*, 12(1), feb 2016. ISSN 1549-6325. doi: 10.1145/2737729. URL https://doi.org/10.1145/2737729. 5.1, 5.2

[96] Leonid Gurvits and Alex Samorodnitsky. Bounds on the permanent and some applications. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 90–99, 2014. doi: 10.1109/FOCS.2014.18. 3.1

[97] Siavash Haghiri, Debarghya Ghoshdastidar, and Ulrike von Luxburg. Comparison-based nearest neighbor search. In *Artificial Intelligence and Statistics*, pages 851–859. PMLR, 2017. 7.1.2

[98] Siavash Haghiri, Damien Garreau, and Ulrike Luxburg. Comparison-based random forests. In *International Conference on Machine Learning*, pages 1871–1880. PMLR, 2018. 7.1.2

[99] Sariel Har-Peled, Piotr Indyk, and Sepideh Mahabadi. Approximate sparse linear regression. In *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018. 5.5

[100] John C. Harsanyi and Reinhard Selten. A generalized nash solution for two-person bargaining games with incomplete information. *Management Science*, 18(5):P80–P106, 1972. ISSN 00251909, 15265501. URL http://www.jstor.org/stable/2661446. 1.3, 4.1

[101] Nicholas James Alexander Harvey. *Matchings, matroids and submodular functions*. PhD thesis, Massachusetts Institute of Technology, 2008. 7.1.2

[102] Trevor Hastie, Robert Tibshirani, and Martin Wainwright. Statistical learning with sparsity. *Monographs on statistics and applied probability*, 143(143):8, 2015. 5.1.1

[103] Zhongtian He, Shang-En Huang, and Thatchaphol Saranurak. Cactus representation of minimum cuts: Derandomize and speed up, 2024. URL https://arxiv.org/abs/2401.10856. 7.6.1

[104] Sandra Heldsinger and Stephen Humphry. Using the method of pairwise comparison to obtain reliable teacher assessments. *The Australian Educational Researcher*, 37(2):1–19, 2010. 7.1

[105] Sedjro Salomon Hotegni, Sepideh Mahabadi, and Ali Vakilian. Approximation algorithms for fair range clustering. In *International Conference on Machine Learning*, pages 13270–13284. PMLR, 2023. 5.1.1

[106] Harold Houba, Gerard Laan, and Yuyu Zeng. Asymmetric nash solutions in the river sharing problem. *Strateg. Behav. Environ.*, 4, 03 2013. doi: 10.2139/ssrn.2243424. 1.3, 4.1

[107] Lingxiao Huang, Jian Li, and Xuan Wu. On optimal coreset construction for euclidean (k, z)-clustering. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, page 1594–1604, 2024. 5.1.1, 1b

[108] Yan Huo, Heng Liang, Si-Qi Liu, and Fengshan Bai. Computing monomer-dimer systems through matrix permanent. *Phys. Rev. E*, 77:016706, Jan 2008. doi: 10.1103/PhysRevE.77.016706. URL https://link.aps.org/doi/10.1103/PhysRevE.77.016706. 3.1

[109] Suk-Geun Hwang, Arnold R. Kräuter, and T.S. Michael. An upper bound for the permanent of a nonnegative matrix. *Linear Algebra and its Applications*, 281 (1):259–263, 1998. ISSN 0024-3795. doi: https://doi.org/10.1016/S0024-3795(98)10040-X. URL https://www.sciencedirect.com/science/article/pii/S002437959810040X. 3.1

[110] Gabriela Jeronimo, Daniel Perrucci, and Elias Tsigaridas. On the minimum of a polynomial function on a basic closed semialgebraic set and applications. *SIAM Journal on Optimization*, 23(1):241–255, 2013. 5.4.9

[111] Matthew Jones, Huy Nguyen, and Thy Nguyen. Fair k-centers via maximum matching. In *International Conference on Machine Learning*, pages 4940–4949, 2020. 5.1.1

[112] Ehud Kalai. Nonsymmetric nash solutions and replications of 2-person bargaining. *International Journal of Game Theory*, 6:129–133, 1977. 1.3, 4.1

[113] Daniel M. Kane, Shachar Lovett, Shay Moran, and Jiapeng Zhang. Active Classification with Comparison Queries . In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 355–366, Los Alamitos, CA, USA, October 2017. IEEE Computer Society. doi: 10.1109/FOCS.2017.40. URL https:

//doi.ieeecomputersociety.org/10.1109/FOCS.2017.40. 7.1.1, 7.1.1, 7.1.1, 7.1.2, 7.4, 7.4.2, 7.4.3, 7.6.3, 7.6.11, 7.6.3

[114] Daniel M Kane, Shachar Lovett, and Shay Moran. Near-optimal linear decision trees for k-sum and related problems. *Journal of the ACM (JACM)*, 66(3):1–18, 2019. 7.1.1, 7.1.2, 7.4

[115] Mamoru Kaneko and Kenjiro Nakamura. The nash social welfare function. *Econometrica: Journal of the Econometric Society*, pages 423–435, 1979. 1.3, 4.1

[116] Leonid G Khachiyan. Rounding of polytopes in the real number model of computation. *Mathematics of Operations Research*, 21(2):307–320, 1996. 1.2, 2.1, 1, 2.1, 2.1.4

[117] Matthäus Kleindessner, Pranjal Awasthi, and Jamie Morgenstern. Fair k-center clustering for data summarization. In *International Conference on Machine Learning*, pages 3448–3457, 2019. 5.1.1

[118] Stein Krogdahl. The dependence graph for bases in matroids. *Discrete Mathematics*, 19(1):47–59, 1977. ISSN 0012-365X. doi: https://doi.org/10.1016/0012-365X(77)90118-2. URL https://www.sciencedirect.com/science/article/pii/0012365X77901182. 7.3.1, 7.3.2

[119] Alex Kulesza and Ben Taskar. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286, 2012. 1.2, 2.1, 2

[120] Eduardo Sany Laber. The computational complexity of some explainable clustering problems, 2022. 1.5.1, 6.1, 6.1.2

[121] Eduardo Sany Laber and Lucas Murtinho. On the price of explainability for some clustering problems. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 5915–5925. PMLR, 2021. URL http://proceedings.mlr.press/v139/laber21a.html. 1.5.1, 6.1, 6.1.2

[122] Guanghui Lan. *First-order and stochastic optimization methods for machine learning*, volume 1. Springer, 2020. 4.6.3

[123] Annick Laruelle and Federico Valenciano. Bargaining in committees as an extension of nash's bargaining theory. *Journal of Economic Theory*, 132(1):291–305, 2007. ISSN 0022-0531. doi: https://doi.org/10.1016/j.jet.2005.05.004. URL https://www.sciencedirect.com/science/article/pii/S0022053105001249. 1.3, 4.1

[124] Lap Chi Lau and Hong Zhou. A local search framework for experimental design. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1039–1058. SIAM, 2021. 2.1, 2.1.4

[125] Euiwoong Lee. Apx-hardness of maximizing nash social welfare with indivisible items. *Information Processing Letters*, 122:17–20, 2017. 4.1.3

[126] Troy Lee, Tongyang Li, Miklos Santha, and Shengyu Zhang. On the cut dimension of a graph. In Valentine Kabanets, editor, *36th Computational Complexity Conference, CCC 2021, July 20-23, 2021, Toronto, Ontario, Canada (Virtual Conference)*, volume 200 of *LIPIcs*, pages 15:1–15:35. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi: 10.4230/LIPICS.CCC.2021.15. URL https://doi.org/10.4230/LIPIcs.CCC.2021.15. 7.1.2

[127] Wenzheng Li and Jan Vondrák. A constant-factor approximation algorithm for nash social welfare with submodular valuations. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 25–36. IEEE, 2022. 2.1.4, 4.1.3

[128] Heng Liang and Fengshan Bai. An upper bound for the permanent of (0,1)-matrices. *Linear Algebra and its Applications*, 377:291–295, 2004. ISSN 0024-3795. doi: https://doi.org/10.1016/j.laa.2003.09.003. URL https://www.sciencedirect.com/science/article/pii/S0024379503007481. 3.1

[129] Hang Liao and Deeparnab Chakrabarty. Learning spanning forests optimally in weighted undirected graphs with cut queries. In Claire Vernade and Daniel Hsu, editors, *Proceedings of The 35th International Conference on Algorithmic Learning Theory*, volume 237 of *Proceedings of Machine Learning Research*, pages 785–807. PMLR, 25–28 Feb 2024. URL https://proceedings.mlr.press/v237/liao24b.html. 7.1.2

[130] Vivek Madan, Mohit Singh, Uthaipon Tantipongpipat, and Weijun Xie. Combinatorial algorithms for optimal design. In *Conference on Learning Theory*, pages 2210–2258, 2019. 2.1, 2.1.4

[131] Vivek Madan, Aleksandar Nikolov, Mohit Singh, and Uthaipon Tantipongpipat. Maximizing determinants under matroid constraints. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 565–576. IEEE, 2020. 1.2.1, 1.2.1, 2.1, 2.1.2, 2.1.2, 2.1.4, 2.4.3, 2.4.6, 2.4.3

[132] Arvind V. Mahankali and David P. Woodruff. Optimal ℓ_1 column subset selection and a fast PTAS for low rank approximation. *CoRR*, abs/2007.10307, 2020. 5.1.1

[133] Konstantin Makarychev and Liren Shan. Near-optimal algorithms for explainable k-medians and k-means. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 7358–7367. PMLR, 2021. URL http://proceedings.mlr.press/v139/makarychev21a.html. 1.5.1, 6.1, 6.1

[134] Konstantin Makarychev and Liren Shan. Explainable *k*-means: don't be greedy, plant bigger trees! In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 1629–1642. ACM, 2022. doi: 10.1145/3519935.3520056. URL https://doi.org/10.1145/3519935.3520056. 1.5.1, 6.1, 6.1.2

[135] Konstantin Makarychev, Yury Makarychev, and Ilya P. Razenshteyn. Performance

of Johnson-Lindenstrauss transform for k-means and k-medians clustering. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 1027–1038. ACM, 2019. doi: 10.1145/3313276.3316350. URL https://doi.org/10.1145/3313276.3316350. 1

[136] Adam W Marcus, Daniel A Spielman, and Nikhil Srivastava. Interlacing families ii: Mixed characteristic polynomials and the kadison—singer problem. *Annals of Mathematics*, pages 327–350, 2015. 2.1.4

[137] Marvin Marcus and Henryk Minc. Permanents. *The American Mathematical Monthly*, 72(6):577–591, 1965. doi: 10.1080/00029890.1965.11970575. URL https://doi.org/10.1080/00029890.1965.11970575. 3.1

[138] Antonis Matakos, Bruno Ordozgoiti, and Suhas Thejaswi. Fair column subset selection. *arXiv preprint arXiv:2306.04489*, 2023. 1

[139] Henryk Minc. Upper bounds for permanents of $(\{0, 1\})$-matrices. *Bulletin of the American Mathematical Society*, 69:789–791, 1963. URL https://api.semanticscholar.org/CorpusID:117771452. 3.1

[140] Henryk Minc. *Permanents*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1984. 3.1

[141] Ankur Moitra. An almost optimal algorithm for computing nonnegative rank. *SIAM J. Comput.*, 45(1):156–173, 2016. 5.1.1

[142] Dana Moshkovitz. The projection games conjecture and the np-hardness of ln n-approximating set-cover. *Theory Comput.*, 11:221–235, 2015. 6.3

[143] Sagnik Mukhopadhyay and Danupon Nanongkai. Weighted min-cut: sequential, cut-query, and streaming algorithms. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, page 496–509, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450369794. doi: 10.1145/3357713.3384334. URL https://doi.org/10.1145/3357713.3384334. 7.1, 7.1.2

[144] John F Nash Jr. The bargaining problem. *Econometrica: Journal of the econometric society*, pages 155–162, 1950. 1.3, 4.1

[145] Balas Kausik Natarajan. Sparse approximate solutions to linear systems. *SIAM journal on computing*, 24(2):227–234, 1995. 5.5

[146] Nhan-Tam Nguyen, Trung Thanh Nguyen, Magnus Roos, and Jörg Rothe. Computational complexity and approximability of social welfare optimization in multiagent resource allocation. *Autonomous agents and multi-agent systems*, 28(2):256–289, 2014. 4.1.3

[147] Aleksandar Nikolov. Randomized rounding for the largest simplex problem. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, pages 861–870. ACM, 2015. 1, 2.1, 2.1.4

[148] Aleksandar Nikolov and Mohit Singh. Maximizing determinants under partition constraints. In *ACM symposium on Theory of computing*, pages 192–201, 2016. 2.1, 2.1.4

[149] Aleksandar Nikolov, Mohit Singh, and Uthaipon Tao Tantipongpipat. Proportional volume sampling and approximation algorithms for A-optimal design. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1369–1386. SIAM, 2019. 1.2, 2.1, 2.1.4

[150] James B. Orlin. Improved algorithms for computing fisher's market clearing prices: computing fisher's market clearing prices. In *Proceedings of the Forty-Second ACM Symposium on Theory of Computing*, STOC '10, page 291–300, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781450300506. doi: 10.1145/1806689.1806731. URL https://doi.org/10.1145/1806689.1806731. 4.1.2

[151] James Oxley. *Matroid Theory*. Oxford University Press, 02 2011. ISBN 9780198566946. doi: 10.1093/acprof:oso/9780198566946.001.0001. URL https://doi.org/10.1093/acprof:oso/9780198566946.001.0001. 7.3.1, 7.3.1

[152] Dimitris Papailiopoulos, Alexandros Dimakis, and Stavros Korokythakis. Sparse pca through low-rank approximations. In *International Conference on Machine Learning*, pages 747–755. PMLR, 2013. 5.1

[153] Vilfredo Pareto. *Manuale di economia politica con una introduzione alla scienza sociale*, volume 13. Società editrice libraria, 1919. 7.1

[154] Orestis Plevrakis, Seyoon Ragavan, and S. Matthew Weinberg. On the Cut-Query Complexity of Approximating Max-Cut. In Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson, editors, *51st International Colloquium on Automata, Languages, and Programming (ICALP 2024)*, volume 297 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 115:1–115:20, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISBN 978-3-95977-322-5. doi: 10.4230/LIPIcs.ICALP.2024.115. URL https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ICALP.2024.115. 7.1.1, 7.4

[155] Friedrich Pukelsheim. *Optimal design of experiments*. SIAM, 2006. 1.2, 2.1, 3

[156] Ran Raz and Boris Spieker. On the "log rank"-conjecture in communication complexity. *Combinatorica*, 15(4):567–588, 1995. 7.1.2

[157] Ilya Razenshteyn, Zhao Song, and David P. Woodruff. Weighted low rank approximations with provable guarantees. In *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*, page 250–263, 2016. 5.1.1

[158] James Renegar. On the computational complexity and geometry of the first-order theory of the reals. part i: Introduction. preliminaries. the geometry of semi-algebraic sets. the decision problem for the existential theory of the reals. *Journal of symbolic computation*, 13(3):255–299, 1992. 5.4.8

[159] James Renegar. On the computational complexity and geometry of the first-order theory

of the reals. part ii: The general decision problem. preliminaries for quantifier elimination. *Journal of Symbolic Computation*, 13(3):301–327, 1992. 5.4.8

[160] J. Robertson and W. Webb. *Cake-cutting algorithms: Be fair if you can*. CRC Press, 1998. 1.3, 4.1

[161] Jrg Rothe. *Economics and Computation: An Introduction to Algorithmic Game Theory, Computational Social Choice, and Fair Division*. Springer Publishing Company, Incorporated, 1st edition, 2015. ISBN 3662479036. 1.3, 4.1

[162] Aviad Rubinstein, Tselil Schramm, and S. Matthew Weinberg. Computing Exact Minimum Cuts Without Knowing the Graph. In Anna R. Karlin, editor, *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*, volume 94 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 39:1–39:16, Dagstuhl, Germany, 2018. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISBN 978-3-95977-060-6. doi: 10.4230/LIPIcs.ITCS.2018.39. URL https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ITCS.2018.39. 7.1, 7.1.1, 7.1.2, 7.2.2, 7.2.2, 7.6.1, 7.6.5, 7.6.6, 7.6.1

[163] Mark Rudelson and Roman Vershynin. The smallest singular value of a random rectangular matrix, 2009. URL https://arxiv.org/abs/0802.3956. 5.4.2

[164] Samira Samadi, Uthaipon Tantipongpipat, Jamie H Morgenstern, Mohit Singh, and Santosh Vempala. The price of fair PCA: One extra dimension. In *Advances in neural information processing systems*, pages 10976–10987, 2018. 1

[165] Alex Samorodnitsky. An upper bound for permanents of nonnegative matrices. *Journal of Combinatorial Theory, Series A*, 115(2):279–292, 2008. ISSN 0097-3165. doi: https://doi.org/10.1016/j.jcta.2007.05.010. URL https://www.sciencedirect.com/science/article/pii/S0097316507000805. 3.1

[166] Ignacio Santamaria, Javier Vía, Michael Kirby, Tim Marrinan, Chris Peterson, and Louis Scharf. Constrained subspace estimation via convex optimization. In *2017 25th European Signal Processing Conference (EUSIPCO)*, pages 1200–1204. IEEE, 2017. 5.1.1, 5.4.1

[167] Alexander Schrijver. A short proof of minc's conjecture. *J. Comb. Theory A*, 25:80–83, 1978. URL https://api.semanticscholar.org/CorpusID:24610177. 3.1

[168] Alexander Schrijver. A course in combinatorial optimization. 1990. URL https://api.semanticscholar.org/CorpusID:54090543. 7.3.2, 7.6.2, 7.6.2

[169] Alexander Schrijver. Counting 1-factors in regular bipartite graphs. *J. Comb. Theory Ser. B*, 72(1):122–135, January 1998. ISSN 0095-8956. doi: 10.1006/jctb.1997.1798. URL https://doi.org/10.1006/jctb.1997.1798. 3.1

[170] Alexander Schrijver. *Theory of Linear and Integer Programming*. Wiley-Interscience, 2000. 2.2.1

[171] Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*, volume B. 01 2003. 7.3.2

[172] Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2003. 2.1.3, 2.2.3, 2.2.3, 2.7.1

[173] Ankit Sharma and Jan Vondrák. Multiway cut, pairwise realizable distributions, and descending thresholds. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 724–733. ACM, 2014. doi: 10.1145/2591796.2591866. URL https://doi.org/10.1145/2591796.2591866. 6.1.1

[174] Mohit Singh and Weijun Xie. Approximate positive correlated distributions and approximation algorithms for D-optimal design. *In Proceedings of SODA*, 2018. 2.1, 2.1.4

[175] Maurice Sion. On general minimax theorems. *Pacific Journal of Mathematics*, 8(1):171 – 176, 1958. 4.6.2

[176] George W. Solues. Extending the minc-brègman upper bound for the permanent. *Linear and Multilinear Algebra*, 47(1):77–91, 2000. doi: 10.1080/03081080008818633. URL https://doi.org/10.1080/03081080008818633. 3.1

[177] Zhao Song, Ali Vakilian, David Woodruff, and Samson Zhou. On socially fair regression and low-rank approximation. In *Advances in Neural Information Processing Systems*, 2024. 1

[178] George W. Soules. New permanental upper bounds for nonnegative matrices. *Linear and Multilinear Algebra*, 51(4):319–337, 2003. doi: 10.1080/0308108031000098450. URL https://doi.org/10.1080/0308108031000098450. 3.1

[179] Mechthild Stoer and Frank Wagner. A simple min cut algorithm. In Jan van Leeuwen, editor, *Algorithms - ESA '94, Second Annual European Symposium, Utrecht, The Netherlands, September 26-28, 1994, Proceedings*, volume 855 of *Lecture Notes in Computer Science*, pages 141–147. Springer, 1994. 7.2.5

[180] Warut Suksompong. Weighted fair division of indivisible items: A review. *Inf. Process. Lett.*, 187(C), January 2025. ISSN 0020-0190. doi: 10.1016/j.ipl.2024.106519. URL https://doi.org/10.1016/j.ipl.2024.106519. 4.1

[181] Marco Di Summa, Friedrich Eisenbrand, Yuri Faenza, and Carsten Moldenhauer. On largest volume simplices and sub-determinants. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 315–323. Society for Industrial and Applied Mathematics, 2015. 1, 2.1, 2.1.4

[182] Uthaipon Tantipongpipat, Samira Samadi, Mohit Singh, Jamie H Morgenstern, and Santosh Vempala. Multi-criteria dimensionality reduction with applications to fairness. In *Advances in Neural Information Processing Systems*, pages 15135–15145, 2019. 1

[183] Joel A Tropp. Column subset selection, matrix factorization, and eigenvalue optimization.

In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, pages 978–986. SIAM, 2009. 5.1

[184] L.G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, 1979. ISSN 0304-3975. doi: https://doi.org/10.1016/0304-3975(79)90044-6. URL https://www.sciencedirect.com/science/article/pii/0304397579900446. 3.1

[185] Ameya Velingker, Maximilian Vötsch, David P. Woodruff, and Samson Zhou. Fast $(1+\varepsilon)$-approximation algorithms for binary matrix factorization. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023. 5.1.1

[186] David P. Woodruff and Taisuke Yasuda. Nearly linear sparsification of ℓ_p subspace approximation, 2024. URL https://arxiv.org/abs/2407.03262. 5.3.3

[187] David P. Woodruff and Taisuke Yasuda. Ridge leverage score sampling for ℓ_p subspace approximation, 2025. URL https://arxiv.org/abs/2407.03262. 5.3.4

[188] Yichong Xu, Hongyang Zhang, Kyle Miller, Aarti Singh, and Artur Dubrawski. Noise-tolerant interactive learning using pairwise comparisons. *Advances in neural information processing systems*, 30, 2017. 7.1.2

[189] Zhirong Yang and Erkki Oja. Linear and nonlinear projective nonnegative matrix factorization. *IEEE Transactions on Neural Networks*, 21:734–749, 2010. URL https://api.semanticscholar.org/CorpusID:7330912. 5.1.1

[190] H. Peyton Young. *Equity: In Theory and Practice*. Princeton University Press, 1994. ISBN 9780691043197. URL http://www.jstor.org/stable/j.ctv10crfx7. 1.3, 4.1

[191] S. Yu, E. C. Ierland, H.-P. Weikard, and X. Zhu. Nash bargaining solutions for international climate agreements under different sets of bargaining weights. *International Environmental Agreements: Politics, Law and Economics*, 17(5):709–729, October 2017. doi: 10.1007/s10784-017-9351-3. URL https://ideas.repec.org/a/spr/ieaple/v17y2017i5d10.1007_s10784-017-9351-3.html. 1.3, 4.1

[192] Xiao-Tong Yuan and Tong Zhang. Truncated power method for sparse eigenvalue problems. *Journal of Machine Learning Research*, 14(4), 2013. 5.1

[193] Zhijian Yuan and Erkki Oja. Projective nonnegative matrix factorization for image compression and feature extraction. In *Image Analysis: 14th Scandinavian Conference, SCIA 2005, Joensuu, Finland, June 19-22, 2005. Proceedings 14*, pages 333–342. Springer, 2005. 5.1.1

[194] Zhijian Yuan, Zhirong Yang, and Erkki Oja. Projective nonnegative matrix factorization : Sparseness , orthogonality , and clustering. 2009. URL https://api.semanticscholar.org/CorpusID:5396302. 5.1.1

[195] Hui Zou, Trevor Hastie, and Robert Tibshirani. Sparse principal component analysis. *Journal of computational and graphical statistics*, 15(2):265–286, 2006. 5.1.1