SUBSET SELECTION VIA SPECTRAL OBJECTIVES

A Dissertation
Presented to
The Academic Faculty

By

Adam Brown

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Mathematics
Algorithms Combinatorics and Optimization

Georgia Institute of Technology

May 2025

SUBSET SELECTION VIA SPECTRAL OBJECTIVES

Thesis committee:

Dr. Mohit Singh
Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Lap Chi Lau
School of Computer Science
University of Waterloo

Dr. Greg Blekherman
School of Mathematics
Georgia Institute of Technology

Dr. Sahil Singla
School of Computer Science
Georgia Institute of Technology

Dr. Santanu Dey
Industrial and Systems Engineering
Georgia Institute of Technology

Date approved: May 2025

Je n’ai fait celle-ci plus longue que parce que je n’ai pas eu le loisir de la faire plus courte.

- *Blaise Pascal*

For my grandfathers, as with everything I put my name on.

ACKNOWLEDGMENTS

First, I would like to thank my advisor Mohit Singh who has been a perfect guide throughout this process, and helped maintain some spirit of play even when the work was difficult and technical. I would also like to thank Aditi Laddha and Madhusudhan Pittu for being wonderful collaborators across many projects, although maybe we spent too much time talking to each other at conferences.

TABLE OF CONTENTS

LIST OF FIGURES

SUMMARY

Selecting a small subset of items that captures key properties of a much larger data set is an important problem in a variety of areas, including machine learning, network design, and fair allocation of goods. We use a geometric model for this problem where the items (or their qualities) are represented by d-dimensional vectors. That is, given a large set E, where each element $e \in E$ is represented by a real vector $v_e \in \mathbf{R}^d$, select a small subset $S \subseteq E$ with good properties.

There are many different possible goals for subset selection problems, and we will focus on objectives of the form $f(S) := \Phi\left(\sum_{i \in S} v_i v_i^\top\right)$ where Φ is some function of the eigenvalues of a matrix. For example, Φ may be the determinant, trace, or minimum eigenvalue. While we could simply limit the number of selected items, we can also have more elaborate constraints. We may instead restrict our attention to subsets S that are bases of a specific matroid. This will allow us to model a much broader class of problems.

This work focuses on four problems which can be modeled in the spectral subset selection framework. The first, is determinant maximization under matroid constraints. For this problem, in the case that we number of vectors we are selecting is exactly d, we provide a combinatorial $O(d^d)$-approximation algorithm based on matroid intersection. Next, we consider the special case of the weighted Nash Social Welfare problem. This problem has its own specially-tailored relaxations, which we use to construct a new approximation algorithm. Then, we turn to the minimum eigenvalue problem with general matroid constraints. By modifying the natural convex program, and randomly rounding the optimal solution, we are able to find $(1 - \epsilon)$-approximate solution in polynomial time, when the dimension of the vectors is constant. Finally, we turn to a practical problem in electrical network design called the network reconfiguration problem. After analyzing various convex relaxations for the problem we propose a collection of potential upper and lower bound algorithms, and compare them practically.

CHAPTER 1
INTRODUCTION

Selecting a small subset of items that captures key properties of a much larger data set is an important problem in a variety of areas, including machine learning, network design, and fair allocation of goods. We use a geometric model for this problem where the items (or their qualities) are represented by d-dimensional vectors. That is, given a large set E, where each element $e \in E$ is represented by a real vector $v_e \in \mathbf{R}^d$, select a small subset $S \subseteq E$ with good properties.

There are many different possible goals for subset selection problems, and we will focus on objectives of the form $f(S) := \Phi\left(\sum_{i \in S} v_i v_i^\top\right)$ where Φ is some function of the eigenvalues of a matrix. For example, Φ may be the determinant, trace, or minimum eigenvalue. This model is general enough to capture practical problems such as experimental design [1, 2, 3], and finding a "well-connected" subgraph of a given network [4, 5, 6].

There is also the question of what it means for a set to be small. The simplest example is, of course, to limit the size of the subset S we select, but we can also have more elaborate constraints. We may instead restrict our attention to subsets S that are bases of a specific matroid. This will allow us to model more network design problems and new problems, such as fairness in the allocation of goods [7, 8, 9, 10, 11, 12]. Our goal will be to develop efficient algorithms to find good approximate solutions to these problems with geometric objectives.

1.1 Spectral Objectives

First, let us explore the possibilities for spectral objectives, and gain some intuition for what we can model. For a subset $S \subseteq E$ we construct the matrix $\sum_{e \in S} v_e v_e^\top$. This matrix will always be symmetric and positive semidefinite, so has non-negative real eigenvalues.

Whatever our subset selection goal may be, we hope to phrase the objective in terms of the properties of this matrix and, often, its eigenvalues.

To keep everything else simple, we will start with the case of straightforward cardinality constraints. This still captures the well-studied problem of experimental design [3], which we will use as an illustrative example. In this problem, we want to measure a vector of unknown parameters $\theta^\star \in \mathbf{R}^d$ by taking a set of linear observations of the form $y_i = v_i^\top \theta^\star + \eta_i$ where $v_i \in \mathbf{R}^d$ and η_i is Gaussian noise. We can select the test vector v_i from a set of candidate vectors, $\{v_1, \ldots, v_n\}$, but each observation has some associated cost. The goal of experimental design is to select a subset of k (often $k \ll n$) vectors, S, from the candidate set such that measurements on S maximize the accuracy of estimating $\theta^\star$ for some criterion. Under the Gaussian assumption on the noise, for a set of measurements S, the maximum likelihood estimator for $\theta^\star$ is the ordinary least squares solution on S, i.e., $\hat{\theta} = \operatorname{argmin}_\theta(\sum_{i \in S} |v_i^\top \theta - y_i|^2)$. A simple computation shows that $\hat{\theta} - \theta^\star$ is distributed as a d-dimensional Gaussian with covariance matrix proportional to $\left(\sum_{i \in S} v_i v_i^\top\right)^{-1}$ [3]. For many reasonable spectral functions, minimizing a function of this covariance matrix corresponds to maximizing some function of the matrix $\sum_{i \in S} v_i v_i^\top$, and we will phrase our objectives this way to fit our framework.

In the experimental design literature, many possible spectral functions have been given special names: in E-design the goal is to maximize $\lambda_{\min}(\cdot)$, which corresponds to minimizing the maximum radius of the confidence ellipsoid; in A-design the goal is to minimize $\operatorname{Tr}((\cdot)^{-1})$, which corresponds to minimizing the sum of the axes of the confidence ellipsoid; and in D-design the goal is to maximize $\det(\cdot)^{1/d}$, which corresponds to minimizing the volume of the confidence ellipsoid. With this intuition we can start to see some relationships between different objectives. It should be clear that the minimum eigenvalue objective is often the strongest (and therefore the most challenging algorithmically), since it also implies bounds on the other objectives. We can also expect some HM-GM-type relationship between the trace and determinant objectives.

There is a substantial body of work on these problems, and we have good understanding of how to approximate these objectives once the number of vectors selected, k, is large enough compared to the dimension d.

- Maximize $\lambda_{\min}(\cdot)$ (E-design): $(1+\epsilon)$-approximation when $k = \Omega(d/\epsilon^2)$ [13],
- Minimize $\mathrm{Tr}((\cdot)^{-1})$ (A-design): $(1+\epsilon)$-approximation when $k = \Omega(d/\epsilon)$ [14, 2],
- Maximize $\det(\cdot)^{1/d}$ (D-design): $(1+\epsilon)$-approximation when $k = \Omega(d/\epsilon)$ [14, 2].

The result for $\lambda_{\min}$ also extends to other reasonable spectral functions (including A- and D-design, where it is weaker than the listed results). The approximations are much worse for k close to d. For example a d-approximation for A-design when $k = d$ [1].

1.2 Combinatorial Constraints

So far, we have discussed cardinality constraints, to make the exploration of possible objective easier. Now we will focus on the possibilities for more complicated definitions of what it means for a subset to be small. The next step up from cardinality, is to consider partition constraints.

We start with out ground set E, which is partitioned into parts $\mathcal{P}_1, \ldots, \mathcal{P}_k$. The restriction is that the final set $S \subseteq E$ we select must contain exactly one element from each part $\mathcal{P}_j$. This already allows us to model interesting allocation problems, and will be a common prototype example in later chapters.

In a general allocation problem we are given a collection of k goods $\mathcal{G}$, and d agents $\mathcal{A}$. The goal is to select a function $\sigma : \mathcal{G} \to \mathcal{A}$ which assigns the items to the agents. We can model this as a partition constraint by considering the set of all pairs $E = \{(i,j) : i \in \mathcal{A}, j \in \mathcal{G}\}$, and dividing it into parts

$$\mathcal{P}_j = \{(i,j) : i \in \mathcal{A}\}.$$

Now, given a set $S \subseteq E$ such that $|S \cap \mathcal{P}_j| = 1$ for all $j \in \mathcal{G}$, we know that for each item $j \in \mathcal{G}$ there is a unique agent $i \in \mathcal{A}$ such that $(i, j) \in S$, so we can construct an assignment by setting $\sigma(j) = i$.

In the case of additive valuations, each agent $i \in \mathcal{A}$ has its own valuation $u_{ij} \geq 0$ for each good $j \in \mathcal{G}$. Given an assignment $\sigma : \mathcal{G} \to \mathcal{A}$, we say that the utility received by agent i in assignment σ is $\sum_{j:\sigma(j)=i} u_{ij}$. The objective is usually to maximize some function of the agents' utilities.

We can combine the partition constraints with different spectral objectives to get a variety of allocation problems. The Nash social welfare objective is to maximize the geometric mean of the agents' valuations, i.e. $\max_{\sigma:\mathcal{G}\to\mathcal{A}} \prod_{i\in\mathcal{A}} \left(\sum_{j:\sigma(j)=i} u_{ij}\right)^{1/d}$. This problem can be modeled in the spectral-objective framework using $\Phi(\cdot) = \det(\cdot)^{1/d}$. For each $(i, j) \in E$ let $v_{ij} = \sqrt{u_{ij}}\mathbb{1}_i \in \mathbf{R}^d$. Then $v_{ij}v_{ij}^\top$ is a $d \times d$ matrix, where the (i, i) entry is exactly u_{ij} and all other entries are zero. Given a basis S corresponding to an assignment σ, the (i, i) entry of the matrix $\sum_{(i,j)\in S} v_{ij}v_{ij}^\top$ will be exactly $\sum_{j:\sigma(j)=i} u_{ij}$, and all off-diagonal entries are zero. Hence the d-th root of the determinant will be exactly the original Nash social welfare objective.

By selecting different spectral objectives, the same reduction can model other well know problems in fair allocation. If we select $\Phi(\cdot) = \mathrm{Tr}(\cdot)$ then we get the maximum utility problem, while if we select $\Phi(\cdot) = \lambda_{\min}(\cdot)$ we get the Santa Claus problem [15, 16, 17, 18, 19].

More generally, we can have matroid constraints. This can model both cardinality, and partition constraints using uniform and partition matroids respectively. This will also allow us to impose new constraints, such as selecting a spanning tree of a graph, or a linearly independent set of vectors.

We will often consider convex relaxation of our spectral selection problems. If our constraining matroid is $\mathcal{M}$ on the ground set E, we let $\mathcal{P}(M) \subseteq \mathbf{R}^E$ denote the corresponding matroid polytope. Then a natural relaxation for the spectral selection problem

with objective $f(S) = \Phi\left(\sum_{i \in S} v_i v_i^\top\right)$ is as follows:

$$\begin{aligned} \max / \min \quad & \Phi(X) \\ & X = \textstyle\sum_{i \in E} x_e \cdot v_e v_e^\top, \\ & x \in \mathcal{P}(M). \end{aligned}$$

This formulation will be a routine starting point in later sections. In many cases it is actually a quite poor relaxation, and we consider adding additional constraints to make it stronger.

1.3 Summary of Results

1.3.1 Determinant Maximization

For the determinant maximization problem our objective is $\Phi(\cdot) = \det(\cdot)$ (or the product of the k largest eigenvalues in the case that $k < d$). We will show how to obtain an $O(d^{O(d)})$-approximation when the number of vectors selected is equal to the dimension.

Theorem 1 *There is a polynomial time algorithm which, given a collection of vectors $v_1, \ldots, v_n \in \mathbf{R}^d$ and a matroid $\mathcal{M} = ([n], \mathcal{I})$ of rank d, returns a set $S \in \mathcal{I}$ such that*

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

In [20] we provided an $O(k^{O(k)})$-approximation algorithm in the case that the rank of the matroid is $k \leq d$, while in [21] we adapted the analysis to get an $O(d^{O(d)})$-approximation in that case that $k \geq d$. The best known estimation algorithms for this problem, which only approximately estimate the optimal value and do not provide an optimal solution (although they can be converted to approximation algorithms with slightly worse guarantees), are an $O(e^k)$-estimation for $k \leq d$ [22, 23, 24, 25], and an $O(d^{O(d)})$-estimation when $k \geq d$ [25]. For $k \geq d$, this approximation ratio matches the previous best

known estimation algorithms for the problem .

Previous approximation algorithms [22, 23, 24, 25] used convex relaxations and their relationship to stable polynomials. In contrast, the algorithm for the case that $r \leq d$ is entirely combinatorial, and for $r \geq d$ we only need some facts about the sparsity of the corresponding convex relaxation. The combinatorial approach is based on the classical matroid intersection algorithm, where we construct an exchange graph and search for short cycles to iteratively improve our approximation.

Note that even the problem of determining if there is a basis $B \subseteq E$ of $\mathcal{M}$ such that $\det\left(\sum_{i \in S} v_i v_i^\top\right) > 0$ is itself a matroid intersection problem; it is equivalent to finding a basis B such that the vectors $\{v_i : i \in B\}$ are linearly independent.

1.3.2 Weighted Nash Social Welfare

It was remarked that the determinant maximization problem with partition constraints can model the Nash Social Welfare (NSW) problem.

$$\max_{\sigma:\mathcal{G}\to\mathcal{A}} \prod_{i\in\mathcal{A}} \left(\sum_{j:\sigma(j)=i} u_{ij} \right)^{1/d}.$$

The unweighted Nash Social Welfare function first appeared as the solution to an arbitration scheme proposed by Nash for two-person bargaining games. Informally, if a game satisfies the following axioms: (1) each individual wishes to maximize the utility to themselves, (2) if we restrict the set of allowable divisions without removing the optimal division, then the optimal division does not change, and (3) the individuals have equal bargaining skill; then the optimal assignment will maximize the NSW objective. This idea was later generalized to multiple players [26, 27]. 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 [28] 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. Optimizing the Nash Social Welfare objective also implies notions of fairness, such as *envy-free* allocation in an approximate sense [29, 30].

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:

$$\prod_{i \in \mathcal{A}} \left(\sum_{j \in \sigma^{-1}(i)} v_{ij} \right)^{w_i}. \tag{1.1}$$

The Nash Social Welfare function with weights (also referred to as asymmetric or non-symmetric Nash Social Welfare) was first studied in the seventies [31, 32] in the context of two-person bargaining games, and can also be obtained through an axiomatic approach. Starting from Nash's three original axioms, if we throw out the third (which states that players have equal bargaining skill) then [32] showed that there must be weights such that the optimal bargain maximizes the weighted NSW objective. Due to the added flexibility of assigning agents different weights, problems in many diverse domains can be modeled using the weighted objective, including bargaining theory [28, 33], water resource allocation [34, 35], and climate agreements [36]. From a context of indivisible goods, the study of this problem has been much more recent [37, 38, 39]. There have also been attempts to extend the ideas of envy-freeness to the weighted setting [40, 41], where the situation is more complicated than in the unweighted setting.

We begin by building our understanding of the unweighted problem. There are many approaches, but we focus on two, which both achieve their approximation algorithms by rounding a convex relaxation for the problem.

On the surface, the two programs, (LogConcave-Unweighted) and (CVX-Unweighted) (see Section 4.4.1 for full details), and their corresponding rounding algorithms are quite

different: [42] uses intuition from economics and market equilibrium to both arrive at (CVX-Unweighted) and also to round it, while [43] uses the properties of log-concave polynomials to round (CVX-Unweighted). However, our next result shows that these two convex programs indeed optimize the same objective.

Theorem 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.*

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, (NCVX-Weighted). 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). This program is still convex in the weighted case and, by repeating the same transformations used to prove Theorem 2, we can start from (LogConcave-Weighted) and get an equivalent convex program (CVX-Weighted).

Theorem 3 (NCVX-Weighted)*,* (LogConcave-Weighted)*, and* (CVX-Weighted) *are relaxations of the weighted Nash Social Welfare problem. Moreover, the convex programs* (LogConcave-Weighted)*, and* (CVX-Weighted) *are equivalent.*

Both of these relaxations will be useful in designing an approximation algorithm for the weighted Nash Social Welfare problem. At a high level, we solve the convex program (CVX-Weighted), and then convert the optimal solution to a solution to the non-convex relaxation NCVX-Weighted. After some adjustments, the same rounding used by [42] also works in the non-convex program. The main loss is in the conversion from the convex to non-convex program, and the end result is the following.

Theorem 4 *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 6) 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 $D_{KL}(\mathbf{w}||\mathbf{u}) = \log n - \sum_{i\in\mathcal{A}} w_i \log \frac{1}{w_i}$.*

This result is stated in the log-scale. Using the standard weighted geometric mean form of the objective, this is equivalent to an approximation ratio of

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

When all the weights are the same, this gives a constant factor approximation.

1.3.3 Minimum eigenvalue in constant dimension

In the minimum eigenvalue the objective is $\Phi(\cdot) = \lambda_{\min}(\cdot)$. This is quite general; any approximation for the minimum eigenvalue problem automatically leads to some approximation for determinant maximization and many other spectral objectives. Even in the case of a partition matroid and vectors which align with the standard basis this setup can model the max-min allocation problem from fair allocation using the same reduction discussed earlier for the case of Nash social welfare.

An instance of the max-min allocation problem is given by a collection of goods $\mathcal{G}$ and agents $\mathcal{A}$, where agent i receives additive utility u_{ij} when assigned item j. The goal is to find an assignment $\sigma : \mathcal{G} \to \mathcal{A}$ which maximizes

$$\min_{i\in\mathcal{A}} \sum_{j:\sigma(j)=i} u_{ij}.$$

We define the ground set $E = \{(i,j) : i \in \mathcal{A}, j \in \mathcal{G}\}$, and divide it into parts $\mathcal{P}_j = \{(i,j) : i \in \mathcal{A}\}$. For each $(i,j) \in E$ let $v_{ij} = \sqrt{u_{ij}}\mathbb{1}_i \in \mathbf{R}^d$. Then $v_{ij}v_{ij}^\top$ is a $d \times d$ matrix, where the (i,i) entry is exactly u_{ij} and all other entries are zero. Given a basis S corresponding to an assignment σ, the (i,i) entry of the matrix $\sum_{(i,j)\in S} v_{ij}v_{ij}^\top$ will be exactly $\sum_{j:\sigma(j)=i} u_{ij}$, and all off-diagonal entries are zero. Hence the minimum eigenvalue of the matrix is exactly the minimum utility received among all agents.

For the minimum eigenvalue problem with general matroid constraints we have shown a $(1-\epsilon)$ approximate solution can be found efficiently by rounding the natural convex relaxation, when the dimension d of the vectors is constant.

Theorem 5 *For any $\epsilon > 0$ there is an $O\left(n^{O(d\log(d)/\epsilon^2)}\right)$-time algorithm which, given a collection of vectors $v_1, \ldots, v_n \in \mathbf{R}^d$ and a matroid $\mathcal{M} = ([n], \mathcal{I})$ returns a set $B \in \mathcal{I}$ such that with probability at least $1 - d^{-4}$*

$$\lambda_{\min}\left(\sum_{i\in B} v_i v_i^\top\right) \geq (1-\epsilon) \cdot \max_{B^\star \in \mathcal{I}} \lambda_{\min}\left(\sum_{i\in B^\star} v_i v_i^\top\right).$$

The algorithm is based on first solving the natural convex relaxation, and then rounding to a basis. If our constraints come from a partition matroid then it is enough to round independently within each part, but for a general matroid it is not so clear. Instead, we use a dependent rounding scheme know as pipage rounding [44] which has been shown to have good matrix concentration properties. By finding a suitable change of basis, we are able to use this technique to get a $(1-\epsilon)$-approximate solution in time $O\left(n^{O(d\log(d)/\epsilon^2)}\right)$.

If we round starting from the convex programming solution $L = \sum_{e\in E} x_e \cdot v_e v_e^\top$, then not only do we get a solution with $\lambda_{\min}\left(\sum_{i\in B} v_i v_i^\top\right) \geq (1-\epsilon)\lambda_{\min}(L)$, we actually find a solution such that $\sum_{i\in B} v_i v_i^\top \succeq (1-\epsilon)\cdot L$. This allows us to generalize the result to other objectives.

Theorem 6 *Suppose we have a collection of vectors $\mathcal{V} = (v_1, \ldots, v_n) \in \mathbf{R}^d$, and a matroid $\mathcal{M} = ([n], \mathcal{I})$. Let $f : \mathbb{S}_d^+ :\to \mathbb{R}$ be a concave, monotone, and homogeneous function given*

with a value and first order oracle. For any $\epsilon > 0$, there is an $O\left(n^{O(d\log(d)/\epsilon^2)}\right)$-time randomized algorithm, which takes $(\mathcal{V}, \mathcal{M}, f)$ as input and returns a set $B \in \mathcal{I}$ such that with probability at least $1 - d^{-4}$,

$$f\left(\sum_{i\in B} v_i v_i^\top\right) \geq (1-\epsilon)\cdot \max_{B^\star\in\mathcal{I}} f\left(\sum_{i\in B^\star} v_i v_i^\top\right).$$

Even the simpler case of partition matroids is powerful enough to allow us to solve discrepancy-style problems like the algorithmic form of the Kadison-Singer problem introduced in [45]. By modeling it as a minimum eigenvalue problem our approach achieves the same high-probability approximation guarantees as [45] with a mild improvement in the runtime.

1.3.4 Network Reconfiguration

Power distribution networks are often built as highly connected networks of lines and busses, but are usually operated only along tree subgraphs (called radial networks) since this makes fault detection and overload protection easier [46]. In the network reconfiguration problem we are given a fixed graph $G = (V, E)$ where V is the set of busses (nodes) and E is the set of lines (edges). We will have a single substation $s \in V$ (source node) and all other nodes will be load busses (demand nodes) with given active power demands d_v for all $v \in V\backslash\{s\}$. Let f_e represent the power flow along line e. The goal is to find a spanning tree rooted at the substation s which minimizes the total energy loss [47, 48, 49].

With a simplifications to the physical model for electrical networks, we can formulate the reconfiguration problem in a much simpler mathematical way. Given a candidate solution spanning tree T, the power flow on each line $e \in E$ will be equal to the total demand of the busses in the subtree of descendant of that edge. If we denote by T_e the subtree not containing s in $T\backslash\{e\}$ then

$$f_e = \sum_{v\in T_e} d_v.$$

In terms of standard flow problems, the flow f is the unique flow on T which satisfies all the demands. We will also assume that every line has unit resistance to simplify the presentation. In this case, the loss of line e, denoted L_e, is exactly

$$L_e = f_e^2 = \left(\sum_{v \in T_e} d_v \right)^2 .$$

Given a spanning tree T we can take the sum over all edges in the tree, and write the loss minimization reconfiguration problem as follows.

Problem 1 *Let $G = (V, E)$ be a graph with a source node $s \in V$ and demands d_v at each $v \in V \backslash \{s\}$. Then the network reconfiguration problem is*

$$\min_T \sum_{e \in T} \left(\sum_{v \in T_e} d_v \right)^2 ,$$

where the minimum is over all spanning trees T.

This problem is called reconfiguration as opposed to just configuration, since the assumption is that we are starting with a network which has already been configured, and we want to open some switches and close others in order to balance load or minimize power loss. The above formulation was motivated by power loss reduction, but a simple change in the interpretation of the variables also allows it to model a load balancing objective for the power network. Thus, working with this more abstract mathematical formulation can give ways to solve multiple distinct reconfiguration problems for power networks.

The network reconfiguration problem can be reformulated to match our general subset selection framework. For each $e \in E$ with $e = (u, v)$ let $v_e = \mathbb{1}_u - \mathbb{1}_v$ be a signed incidence vector for the edge e. For a tree T let B_T be the $|V| \times |E|$ matrix whose columns are v_e for $e \in T$. Then the unique flow on T satisfying the demands d is the solution to the linear system $B_T f^T = d$. Thus, we can write $f^T = B_T^+ d$, where B_T^+ is the pseudo-inverse. Then

the energy of f^T is

$$\|f^T\|^2 = \|B_T^+ d\| = \left\langle dd^\top, \left(\sum_{e \in T} v_e v_e^\top\right)^+ \right\rangle.$$

While not directly a function of the eigenvalues of the Laplacian, this objective is similar to taking $\Phi(\cdot) = \mathrm{Tr}\left((\cdot)^{-1}\right)$ in the spectral-objective framework, and using the graphic matroid of G as our constraint. By replacing the matrix $dd^\top$ in the objective with other positive semidefinite matrices we can also model additional combinatorial problems; by using the identity matrix we get the problem of finding a minimum routing cost tree [50, 51] and by using the Laplacian of the underlying graph G we get the problem of finding a low-stretch spanning tree [52]. This problem also generalizes A-design [1, 53], and we show that a straightforward rounding algorithm achieves and $O(n)$-approximation, matching the best know bound for the network reconfiguration problem [48].

We compare this formulation with previously used quadratic programs, and theoretically shown that the spectral formulation is stronger.

We then provide a list of potential upper and lower bound heuristics for the problem; the two lower bounds come from solving fractional relaxations, while the two upper bounds are combinatorial algorithms which return spanning trees. We provide a complete description of these algorithms, and compare them in practice.

One natural and commonly-used algorithm for the network reconfiguration problem is local search. Starting from any feasible spanning tree, add an unused edge from the graph and then delete an edge from the cycle that would create. Iterating this process for as long as we can improve the objective, we may hope to get close to the optimal solution. Unfortunately, we can show that local search can end quite far from the optimal solution.

Theorem 7 *There is a graph G on n vertices and a spanning tree T of G such that T is locally optimal for the network reconfiguration objective, and the gap between the objective*

of T and the optimal solution is

$$\Omega\left(\frac{n}{(\log\log n)^2}\right).$$

CHAPTER 2
GENERAL PRELIMINARIES AND BACKGROUND

In this chapter we review the classical results which will form the foundation for much of the work presented in this thesis.

First, in Section 2.1 we review the theory of matroid intersection. In the traditional approach to matroid weighted intersection we are given two matroids and take a "greedy" approach; we start from an empty set and iteratively grow a common basis of the two matroids. We present an alternative "local-search" approach where we start from a common basis, and make updates to improve the weight while maintaining a basis in each iteration. The same approach was also outlined in [54], and we give a complete analysis here for completeness. This perspective on matroid intersections will be an essential component of our approximation algorithm for determinant maximization in Chapter 3.

Next, in Section 2.2 we turn from the combinatorial to the polyhedral theory of matroids. We will discuss key properties of matroid polytopes, before giving an overview of pipage rounding. This dependent rounding technique gives a way to round a fractional solution to a linear or convex program with matroid constraints, and output a basis of the matroid. We will briefly explain what guarantees are possible for spectral objectives. This technique is a key component of our algorithm for the minimum eigenvalue problem, which is discussed in Chapter 5.

Finally, in Section 2.3 we will give a brief overview of electrical flow problems on graphs, and the required background material from linear algebra. The problem of computing the effective resistance of a network can be formulated as a linear-algebraic problem in a standard way. This perspective forms the foundation of our approach to the network reconfiguration problem in Chapter 6.

2.1 Matroid Intersection

In this section we will review the basics of matroid theory, and give a "local-search" approach to matroid intersection. Most of the material can be found in Chapter 39 of [55], but we will use notation that matches Chapter 3.

Definition 1 *A pair $(E, \mathcal{I})$ is called a matroid if E is a finite set and $\mathcal{I}$ is a nonempty collection of subsets of E satisfying*

(i) if $I \in \mathcal{I}$ and $J \subseteq I$ then $J \in \mathcal{I}$,

(ii) if $I, J \in \mathcal{I}$ and $|I| < |J|$ then there is $z \in J \backslash I$ such that $I + z \in \mathcal{I}$.

This is the independent-set definition of a matroid. Given a matroid $\mathcal{M} = (E, \mathcal{I})$, a subset I of E is called independent if $I \in \mathcal{I}$, and dependent otherwise. A subset B of E is called a basis of $\mathcal{M}$ if B is an inclusion-wise maximal independent set. The second property in the above definition implies that all bases have the same cardinality. We can similarly define a basis for any subset U of E. The size of the largest independent set in a subset U of E is called the rank of U, and is denoted $r_{\mathcal{M}}(U)$ (we suppress the $\mathcal{M}$ index when it is clear from context). The concept of a matroid can also be defined directly in terms of either its bases, or the rank function.

Theorem 8 (Theorem 39.6 in [55]) *Let E be a set and $\mathcal{B}$ be a nonempty collection of subsets of E. The following are equivalent*

(i) $\mathcal{B}$ is the set of bases of a matroid;

(ii) if $B, B' \in \mathcal{B}$ and $x \in B' \backslash B$ then there is $y \in B \backslash B'$ such that $B' - x + y \in \mathcal{B}$;

(iii) if $B, B' \in \mathcal{B}$ and $x \in B' \backslash B$ then there is $y \in B' \backslash B$ such that $B + x - y \in \mathcal{B}$

While this result provides a complete characterization of matroids, we can actually get a stronger version of the basis exchange properties in Theorem 8.

Theorem 9 (Theorem 39.12 in [55]) *Let $\mathcal{M} = (E, \mathcal{I})$ be a matroid. Let B_1 and B_2 be bases and let $x \in B_1 \backslash B_2$. Then there exists a $y \in B_2 \backslash B_1$ such that both $B_1 - x + y$ and $B_2 - y + x$ are bases.*

This will be a key insight for matroid intersection. First, we construct the (directed) exchange graph for a single matroid. For any matroid $\mathcal{M} = (E, \mathcal{I})$ and independent set $I \in \mathcal{I}$, define $D_{\mathcal{M}}(I)$ to be the bipartite graph with vertices E partitioned into I and $E \backslash I$, and an edge from $y \in I$ to $z \notin I$ if $I - y + z \in \mathcal{I}$.

By repeatedly applying Theorem 9 we get the following

Theorem 10 (Theorem 39.12a in [55]) *Let $\mathcal{M} = (E, \mathcal{I})$ be a matroid and let $I, J \in \mathcal{I}$ with $|I| = |J|$. Then $D_{\mathcal{M}}(I)$ contains a perfect matching on the vertices in the symmetric difference $I \triangle J$.*

There is also a partial converse to the above theorem, originally due to Krogdahl [56].

Theorem 11 ([56], Theorem 39.13 in [55]) *Let $\mathcal{M} = (E, \mathcal{I})$ be a matroid and $I \in \mathcal{I}$. Let $J \subseteq S$ be such that $|I| = |J|$ and such that $D_{\mathcal{M}}(I)$ contains a unique perfect matching on $I \triangle J$, then J is independent.*

With these results we are ready to state and then solve the matroid intersection problem. First, we start with the unweighted version. Given two matroids $\mathcal{M}_1 = (E, \mathcal{I}_1)$ and $\mathcal{M}_2 = (E, \mathcal{I}_2)$, the goal is to find the largest set $I \in \mathcal{I}_1 \cap \mathcal{I}_2$. Edmonds showed that this problem can be solved in polynomial time.

Theorem 12 (Theorem 41.4 in [55]) *A maximum-size common independent set in two matroids can be found in polynomial time.*

We will use this algorithm as a black-box to initialize our algorithm for the weighted version of the problem. In weighted matroid intersection we are again given two matroids

$\mathcal{M}_1 = (E, \mathcal{I}_1)$ and $\mathcal{M}_2 = (E, \mathcal{I}_2)$, but also a weight function $w : E \to \mathbb{Q}$, and the goal is to find a set $I \in \mathcal{I}_1 \cap \mathcal{I}_2$ which maximizes $w(I) := \sum_{e \in I} w_e$.

To solve this problem, we define the exchange graph $G_{\mathcal{M}_1,\mathcal{M}_2}(I)$ for both matroids $\mathcal{M}_1$ and $\mathcal{M}_2$. The graph $G_{\mathcal{M}_1,\mathcal{M}_2}(I)$ is a bipartite graph with parts I and $E \backslash I$ and consists of the edges of $D_{\mathcal{M}_1}(I)$ and the reverse of the edges in $D_{\mathcal{M}_2}(I)$. That is, there is an edge from $y \in I$ to $z \in E \backslash I$ if $I - y + z \in \mathcal{I}_1$ and an edge from $z \in E \backslash I$ to $y \in I$ if $I - y + z \in \mathcal{I}_2$. Moreover, we define a new weight function $l : E \to \mathbb{Q}$ where we give each vertex $y \in I$ a weight $l_y := w_y$ and each vertex $z \in E \backslash I$ a weight of $l_z := -w_z$.

The plan will be to find a negative-weight cycle C in the exchange graph and then update $I \leftarrow I \triangle C$ until there are no more negative-weight cycles. Not just any negative-weight cycle will do fo this update. To ensure that $I \triangle C$ is independent it is important that we select a inclusion-wise *minimal* negative-weight cycle with respect to its vertex set.

Lemma 1 ([56], Equivalent to Lemma 41.5α in [55]) *If C is a minimal negative-weight cycle in $G_{\mathcal{M}_1,\mathcal{M}_2}(I)$ then $J = I \triangle C$ is independent in both $\mathcal{M}_1$ and $\mathcal{M}_2$.*

Proof By symmetry it suffices to show that $J \in \mathcal{I}_1$, so suppose this is not the case. Let N_1 and N_2 be the sets of arcs of C belonging to $D_{\mathcal{M}_1}(I)$ and $D_{\mathcal{M}_2}(I)$ respectively. The edges N_1 form a perfect matching on $I \triangle J$, so since J is not independent in $\mathcal{M}_1$, Theorem 11 implies that there is another perfect matching N_1' on $I \triangle J$ such that $N_1' \neq N_1$. Consider the directed multi-graph $D = (V(C), A)$ formed by taking all the arcs in N_1 and N_1', with two copies of the edges in $N_1 \cap N_1'$, plus two parallel copies of each of the arcs in N_2.

Each vertex of $V(C)$ has exactly two incoming and two outgoing arcs in D. Moreover, since $N_1' \neq N_1$ there is an edge of N_1 which creates a chord across the copy of C in D, and so there is a directed circuit C_1 with $V(C_1) \subsetneq V(C)$. As D is Eulerian we can extend this to a decomposition of its arcs into directed circuits $C_1, \ldots, C_k$. Then, since $V(C_1) \neq V(C)$ we know that at most one cycle C_j can have $V(C_j) = V(C)$. If this is the case for some index j, then $\sum_{i \neq j} l(V(C_i)) = l(V(C)) < 0$, so there exists $i \neq j$ such that $l(V(C_i)) < 0$,

and $V(C_i)$ is a proper subset of $V(C)$, which is a contradiction. Otherwise, $V(C_j)$ is a proper subset of $V(C)$ and $\sum_j l(V(C_j)) = 2 \cdot l(V(C)) < 0$, so again there is a cycle C_i such that $l(V(C_i)) < 0$. □

This implies that we can iteratively update on negative weight cycles and obtain the maximum weight common independent set.

Theorem 13 ([56], Equivalent to Theorem 41.5 in [55]) *Let $I \in \mathcal{I}_1 \cap \mathcal{I}_2$ be a maximum cardinality common independent set. I is a maximum weight common basis if and only if $G_{\mathcal{M}_1,\mathcal{M}_2}(I)$ has no negative-weight directed cycle.*

Proof Suppose that $G_{\mathcal{M}_1,\mathcal{M}_2}(I)$ has a directed circuit of negative length. Then choose C to be a negative-weight cycle with $|V(C)|$ minimum. Then $J := I \triangle V(C)$ has greater weight than I, and by the previous lemma we know that J is independent in both matroids.

Conversely, consider another maximum cardinality common independent set $J \in \mathcal{I}_1 \cap \mathcal{I}_2$. By Theorem 10 both $D_{\mathcal{M}_1}(I)$ and $D_{\mathcal{M}_2}(I)$ contain a perfect matching on $I \triangle J$. Together, these two matchings from a vertex-disjoint union of directed circuits $C_1, \ldots, C_k$. Then

$$w(I) - w(J) = \sum_{i=1}^{k} l(V(C_i)) \geq 0,$$

implying that $w(I) \geq w(J)$. □

2.2 Matroid Polytopes and Pipage Rounding

In this section, we provide the necessary background on matroids, as well as the lower tail versions of lemmas from [44], which let us prove Lemma 5.

Given a matroid $\mathcal{M}$, the matroid base polytope is the convex hull of indicator vectors of the bases of $\mathcal{M}$, and is denoted $\mathcal{P}(\mathcal{M})$. The base polytope has the following linear

description

$$\begin{aligned}\mathcal{P}(\mathcal{M}) &= \operatorname{conv}\{\chi(B) : B \text{ a basis of } \mathcal{M}\} \\ &= \left\{ x \in \mathbf{R}^E : \sum_{e\in E} x_e = r(E), \sum_{e\in U} x_e \le r(U)\ \forall U \subseteq E, x \ge 0 \right\}.\end{aligned}$$

Cunningham [57] showed that given $x \in \mathbf{R}_+^E$, it is possible to find a violated constraint for $\mathcal{P}(\mathcal{M})$ in strongly polynomial time using only an independence oracle for the matroid $\mathcal{M}$.

The main idea behind pipage rounding is to iteratively transform a point $x \in \mathcal{P}(\mathcal{M})$ to a basis of $\mathcal{M}$ while ensuring that the some desired failure probability does not increase. We say that a function $g : \mathcal{P}(\mathcal{M}) \to \mathbf{R}$ is concave under swaps if for all $a, b \in E$ and $x \in \mathcal{P}(\mathcal{M})$ the map $z \mapsto g(x + z(e_a - e_b))$ is concave. Pipage rounding works well when the probability of the failure case can be bounded by a function which is concave under swaps.

If x is not an extreme point of $\mathcal{P}(\mathcal{M})$, then there exist $a, b \in [n]$ and $\epsilon > 0$ such that $x \pm \epsilon(e_a - e_b) \in \mathcal{P}(\mathcal{M})$. Let $l = \min\{z : x + z(e_a - e_b) \in \mathcal{P}(\mathcal{M})\}$ and $u = \max\{z : x + z(e_a - e_b) \in \mathcal{P}(\mathcal{M})\}$.

With this, we can define $x^l = x + l(e_a - e_b)$ and $x^u = x + u(e_a - e_b)$. Since $g(x + z(e_a - e_b))$ is concave as a function of z, we know that either $g(x^l) \le g(x)$ or $g(x^u) \le g(x)$. Moreover, both x^l and x^u are on a lower dimensional face than the initial point x. Thus, for any initial point $x_0 \in \mathcal{P}(\mathcal{M})$, a total of m iterations suffice to find an extreme point with $\hat{x}$ with $g(\hat{x}) \le g(x_0)$.

In randomized pipage starting at $x \in \mathcal{P}(\mathcal{M})$, our next iterate x' of the rounding procedure will be x^l with probability $\frac{u}{u-l}$ and x^u with probability $\frac{-l}{u-l}$. This ensures that $\mathbf{E}(x') = x$, and the concavity under swaps guarantees that $\mathbf{E}[g_\epsilon(x')] \le g_\epsilon(x)$ by Jensen's in the variable z. If we start at a point $x_0 \in \mathcal{P}(\mathcal{M})$ and iterate this random procedure m times, we get an extreme point $\hat{x}$ which satisfies $\mathbf{E}[\hat{x}] = x_0$ and $\mathbf{E}[g(\hat{x})] \le g(x_0)$.

This intuition leads to algorithm 1.

Algorithm 1 Randomized Pipage Rounding

Input: Point $x \in \mathcal{P}(\mathcal{M})$, where $\mathcal{P}(\mathcal{M})$ is a matroid base polytope

while x is not integral **do**

  $a, b \leftarrow$ distinct elements of $[n]$ s.t. $\exists \epsilon > 0$ with $x \pm \epsilon(e_a - e_b) \in \mathcal{P}(\mathcal{M})$

  $\ell \leftarrow \min\{y \geq 0 : x - y(e_a - e_b) \in \mathcal{P}(\mathcal{M})\}$

  $h \leftarrow \max\{y \geq 0 : x + y(e_a - e_b) \in \mathcal{P}(\mathcal{M})\}$

  $$x \leftarrow \begin{cases} x - \ell(e_a - e_b) & \text{w.p. } \ell/(\ell + h) \\ x + h(e_a - e_b) & \text{w.p. } h/(\ell + h) \end{cases}$$

end while

Return basis $B \in \mathcal{P}(\mathcal{M})$ with indicator vector x

Theorem 14 *[44] There is a randomized polynomial time algorithm that, given $x_0 \in \mathcal{P}(\mathcal{M})$, outputs an extreme point $\hat{x}$ of $\mathcal{P}(\mathcal{M})$ with $\mathbf{E}[\hat{x}] = x_0$ and such that for any g concave under swaps $\mathbf{E}[g(\hat{x})] \leq g(x)$.*

We will mainly make use of this theorem through the following claim. The conditions of the claim come from pessimistic estimators, but not all of them are strictly necessary. For a point $x \in [0,1]^n$, let $D(x)$ represent the corresponding product distribution over $\{0,1\}^n$ with marginals given by x.

Lemma 2 *[44] Let $\mathcal{E} \subseteq \{0,1\}^n$ and $g : \mathcal{P}(\mathcal{M}) \to \mathbf{R}$ satisfy*

$$\mathbf{P}_{x \sim D(x)}[x \in \mathcal{E}] \leq g(x), \text{ and}$$
$$\min\{g(x - x_i e_i), g(x + (1 - x_i)e_i)\} \leq g(x)$$

for all $x \in [0,1]^n$, and g be concave under swaps. If pipage rounding is started at an initial point $x_0 \in P$ and $\hat{x}$ is the random extreme point, then $\mathbf{P}[\hat{x} \in \mathcal{E}] \leq g(x_0)$.

Essentially, this lemma says that if we have a pessimistic estimator which is concave

under swaps, then pipage rounding has the same type of concentration behavior as independent rounding, but will actually return a vertex of the matroid polytope.

For our particular application, we will be choosing the function g to be an estimator for matrix concentration due to Tropp [58].

Theorem 15 *[58, Theorem 5.1.1] Let $M_1, \ldots, M_n$ be symmetric matrices with*

$$\max_{i \in [n]} \lambda_{\max}(M_i) \leq R \quad \textit{and let} \quad \mu_{\min} = \lambda_{\min}\left(\mathbf{E}_{x \sim \mathcal{D}(x)}\left[\sum_{i \in [n]} x_i M_i\right]\right).$$

For $t \in \mathbb{R}$, we have the bound

$$\mathbf{P}_{x \sim D(x)}\left[\lambda_{\min}\left(\sum_{i \in [n]} x_i M_i\right) \leq t\right] \leq \inf_{\theta < 0} g_{t,\theta}(x)$$

where $g_{t,\theta}(x) = e^{-\theta t} \cdot \operatorname{tr} \exp\left(\sum_{i \in [n]} \log \mathbf{E}\, e^{\theta x_i \cdot M_i}\right)$. Furthermore, for $t = (1-\epsilon)\mu_{\min}$,

$$g_{t,\theta}(x) \leq d \cdot \left(\frac{e^{-\epsilon}}{(1-\epsilon)^{1-\epsilon}}\right)^{\mu_{\min}/R}.$$

This is the lower-tail version of the same concentration inequality which was used in [44]. In that paper, they provide an upper-tail version of Lemma 5 using a new generalization of Lieb's concavity theorem, stated below. We use $\mathbb{S}_d$ to denote the set of symmetric $d \times d$ matrices, $\mathbb{S}_d^+$ for the subset of positive semidefinite matrices, and $\mathbb{S}_d^{++}$ for the subset of positive definite matrices.

Lemma 3 *[44] Let $L \in \mathbb{S}_d$, $C_1, C_2 \in \mathbb{S}_d^{++}$, and $K_1, K_2 \in \mathbb{S}_d^+$. Then the univariate function*

$$z \to \operatorname{tr} \exp\left(L + \log(C_1 + zK_1) + \log(C_2 - zK_2)\right)$$

is concave in a neighborhood of 0.

The logarithm of a positive definite matrix is usually defined in terms of its spectral decomposition. For $M \in \mathbb{S}_d^{++}$, if

$$M = \sum_{i=1}^{d} \lambda_i \cdot v_i v_i^\top,$$

where $\lambda_i > 0$ are the eigenvalues of M, and v_i the corresponding eigenvectors, then

$$\log(M) := \sum_{i=1}^{d} \log(\lambda_i) \cdot v_i v_i^\top.$$

As a consequence, we get the following lemma.

Lemma 4 *For $\theta < 0$, all $x \in [0,1]^m$, the function*

$$g_{t,\theta}(x) = e^{-\theta t} \cdot \operatorname{tr} \exp\left(\sum_{i \in [m]} \log \mathbf{E}_{x \sim \mathcal{D}(x)} e^{\theta x_i \cdot M_i} \right)$$

is concave under swaps.

Proof Let $C_i := \mathbf{E}_{x \sim \mathcal{D}(x)}[e^{\theta x_i \cdot M_i}] = x_i \cdot e^{\theta M_i} + (1 - x_i) \cdot I \succ 0$, and for any $i \in [n]$

$$\mathbf{E}_{x \sim \mathcal{D}(x + z e_i)}[e^{\theta x_i \cdot M_i}] = (x_i + z) e^{\theta M_i} + (1 - x_i - z) \cdot I = C_i - z \cdot (I - e^{\theta M_i}).$$

Then $\forall a, b \in [n]$,

$$\begin{aligned} g_{t,\theta}(x + z(e_a - e_b)) &= e^{-\theta t} \cdot \operatorname{tr} \exp\Bigg(\sum_{i \in [n] \setminus \{a,b\}} \log C_i + \log\left(C_b + z \cdot (I - e^{\theta M_b})\right) \\ &\qquad + \log\left(C_a - z \cdot (I - e^{\theta M_a})\right) \Bigg) \\ &= e^{-\theta t} \cdot \operatorname{tr} \exp\left(L + \log\left(C_b + z \cdot K_b\right) + \log\left(C_a - z \cdot K_a\right)\right), \end{aligned}$$

where $K_a = (I - e^{\theta M_a})$, $K_b = (I - e^{\theta M_b})$, and $L = \sum_{i \in [n] \setminus \{a,b\}} \log C_i \in \mathbb{S}_d$. If $\theta \leq 0$, then $K_a \succeq 0$ and $K_b \succeq 0$. Using Lemma 3, $z \to g_{t,\theta}(x + z(e_a - e_b))$ is concave in z, and the result follows. □

Combining Lemma 4 and Theorem 15 with Lemma 2, we obtain the following result, which we will use crucially in Chapter 5.

Lemma 5 *Let $\mathcal{P}(\mathcal{M})$ be a matroid base polytope and $x \in \mathcal{P}(\mathcal{M})$. Let $M_1, \ldots, M_m$ be self-adjoint matrices that satisfy $\lambda_{\max}(M_i) \leq R$. Let $\mu = \lambda_{\min}\left(\sum_{i\in[n]} x_i M_i\right)$. If randomized pipage rounding (Algorithm 1) starts at x and outputs the extreme point $\hat{x} = \chi(B)$ of $\mathcal{P}(\mathcal{M})$, then we have*

$$\mathbf{P}\left[\lambda_{\min}\left(\sum_{i\in B} M_i\right) \leq (1-\epsilon)\cdot\mu\right] \leq d \cdot \exp\left(\frac{-\epsilon^2\mu}{2R}\right).$$

2.3 Electrical Flows and Linear Algebra

In this section we review some basics from the theory of electrical flows on graphs, and a few related results from linear algebra which will be useful in later sections.

We start by defining unit (s,t)-flows on an undirected graph $G = (V, E)$. We fix an orientation of the edges arbitrarily, and for each edge construct a vector $v_e \in \mathbf{R}^V$ so that $v_e = \mathbb{1}_u - \mathbb{1}_v$ if edge e was oriented from u to v. Let B be the $V \times E$, matrix whose columns are the vectors v_e. A unit $(s-t)$-flow is a function $f : E \to \mathbf{R}$ which satisfies the flow-conservation constraints

$$\sum_{e=uv} f_e - \sum_{u=vu} f_e = \begin{cases} 1 & \text{if } v = t, \\ -1 & \text{if } v = s, \\ 0 & \text{otherwise.} \end{cases}$$

Equivalently, the flow vector satisfies $Bf = \mathbb{1}_t - \mathbb{1}_s$.

In an electrical flow problem we additionally have edge weights w_e, and we interpret the graph as an electrical network where edge e has conductance w_e. The key criteria that distinguishes electrical flows is that they satisfy *Ohm's law*; there exists potentials $\phi \in \mathbf{R}^V$ such that $f_e = w_e \cdot (\phi(v) - \phi(u))$ is e is oriented from u to v.

We can also reinterpret Ohm's law as a statement about a certain optimization problem over unit (s,t)-flows.

Claim 2.3.1 *Let $G = (V,E)$ be a graph with conductance w_e for each $e \in E$. Then $f \in \mathbf{R}^E$ is an (s,t)-electrical flow if and only if it is the optimal solution to the following optimization problem:*

$$\max \left\{ \sum_{e \in E} \frac{f_e^2}{w_e} : Bf = \mathbb{1}_t - \mathbb{1}_s \right\}.$$

Proof We will show that the optimality conditions for the above optimization problem are equivalent to Ohm's law. The objective is strictly convex and the constraints are linear. To create the Lagrangian relaxation we add variables $\lambda \in \mathbf{R}^V$. The Lagrangian is then

$$L(f;\lambda) = \sum_{e \in E} \frac{f_e^2}{w_e} - \lambda^\top \left(Bf - \mathbb{1}t + \mathbb{1}_s \right).$$

If e is oriented from u to v then he derivative with respect to the flow variables f_e is

$$\frac{\partial}{\partial f_e} L = \frac{2}{w_e} f - \lambda_v + \lambda_u.$$

This is zero exactly when

$$f_e = w_e \cdot \left(\frac{\lambda_v}{2} - \frac{\lambda_u}{2} \right),$$

and when we set $\phi = \lambda/2$ this is exactly Ohm's law. Since this program has a convex quadratic objective, linear constraints, and satisfies Slater's condition the first-order conditions are both necessary and sufficient for optimality. □

We can also extend the concept of electrical flows to arbitrary demand vectors $d \in \mathbf{R}^V$ as long as they satisfy $\sum_{v \in V} d_v = 0$. The same proof above shows that the existence of potentials is equivalent to the optimality conditions of the appropriate optimization problem.

The optimization perspective will be essential for writing electrical flow problems in our spectral selection framework. The problem is a least-squares problem which is very

well studied in linear algebra. First we need to recall the singular value decomposition.

Theorem 16 *Given $A \in \mathbf{R}^{m\times n}$ and assume $n \geq m$. Then A may be written as $U\Sigma V^\top$ where $U \in \mathbf{R}^{m\times m}$ is orthogonal, $V \in \mathbf{R}^{n\times n}$ is orthogonal, and*

$$\Sigma = \begin{bmatrix} \sigma_1 & & & 0 \\ & \ddots & & \vdots \\ & & \sigma_n & 0 \end{bmatrix},$$

where $\sigma_1 \geq \cdots \geq \sigma_k = \sigma_{k+1} = \cdots = \sigma_n = 0$ where k is the rank of A.

The values $\sigma_1, \ldots, \sigma_n$ are called the singular values of A, and they are also equal to the positive square roots of the eigenvalues of $AA^\top$. Using the singular value decomposition we can define the *pseudoinverse* of a rectangular or low-rank matrix.

Definition 2 *Given a matrix $\Sigma \in \mathbf{R}^{m\times n}$ which is partially diagonal, we define its pseudoinverse Σ^+ to be the $n \times m$ matrix whose diagonal consists of the reciprocals of the non-zero diagonal values of Σ and zero otherwise.*

Given a matrix $A \in \mathbf{R}^{m\times n}$ with singular value decomposition $A = U\Sigma V^\top$ the pseudoinverse of A is the matrix $A^+ := V\Sigma^+ U^\top$.

Equivalently, the pseudoinverse can be defined in terms of its algebraic properties.

Theorem 17 *Given a matrix A, the pseudoinverse A^+ is the unique matrix satisfying the following properties:*

1. *$AA^+A = A$,*
2. *$A^+AA^+ = A^+$,*
3. *AA^+ is symmetric, and*
4. *A^+A is symmetric.*

We will mostly take advantage of the first property, which can be interpreted as stating that AA^+ acts as an identity on the column space of A. Thus, given a matrix A and a vector b in its column space, we know that A^+b will be a solution to the linear system $Ax = b$. In fact, it will always be the solution with minimum 2-norm.

Claim 2.3.2 *Given a matrix $A \in \mathbf{R}^{m\times n}$ and a vector $b \in \mathbf{R}^m$ in the column space of A, the solution to $Ax = b$ with minimum 2-norm is A^+b.*

If B is a node-arc incidence matrix for a graph G, then the matrix $L_G := BB^\top$ is called the Laplacian for the graph G. This matrix will be positive semidefinite, and we can equivalently write $L_G = \sum_{e\in E}(\mathbb{1}_u - \mathbb{1}_v)(\mathbb{1}_u - \mathbb{1}_v)^\top$.

The final piece of linear algebra which is helpful in our presentation of graph Laplacians is Cauchy-Binet. This allows us to relate the determinant of a product of two matrices to the determinants of the square submatrices of the the two input matrices.

Theorem 18 (Cauchy-Binet) *Let A and B be $m \times n$ matrices with $m \leq n$. Then*

$$\det(AB^\top) = \sum_{S\subseteq[n]:|S|=m} \det(A_S)\det(B_S),$$

where A_S and B_S are the square submatrices of A and B respectively consisting of only the columns in the set S.

We can apply this theorem to our graph Laplacians to obtain the Matrix-tree Theorem due to Kirchoff.

Theorem 19 *Let $G = (V, E)$ be a graph and L_G is Laplacian. Let $[L_G]_{-i,-i}$ be the submatrix obtained by deleting the ith row and ith column. Then the number of spanning trees of G is exactly $\det([L_G]_{-i,-i})$.*

Proof We can write $[L_G]_{-i,-i} = [B]_{-i}[B]_{-i}^\top$, where $[B]^{-i}$ is obtained from the node-arc incidence matrix by deleting the row corresponding to vertex i. Now, by Cauchy-Binet 18

$$\det([L_G]_{-i,-i}) = \det([B]_{-i}[B]_{-i}^\top) = \sum_{S \subset E : |S| = n-1} \det([B]_{-i,S})^2.$$

Now, the square submatrix $[B]_{-i,S}$ has full rank if and only if the edges S are acyclic, and therefore span a tree. Moreover, in this case $\det([B]_{-i,S}) = \pm 1$. Thus,

$$\det([L_G]_{-i,-i}) = \sum_{S \subset E : |S| = n-1} \det([B]_{-i,S})^2 = \sum_{T \text{ a spanning tree}} 1.$$

□

CHAPTER 3
DETERMINANT MAXIMIZATION

This chapter is based on joint work with Aditi Laddha, Madhusudhan Pittu, Mohit Singh, and Prasad Tetali. It is based on a combination of [20] and [21]. The first appeared in the proceedings of the 63rd IEEE Foundations of Computer Science (FOCS 2022), and the second is available as a preprint.

Determinant maximization problem gives a general framework that models problems arising in as diverse fields as statistics [3], convex geometry [59], fair allocations [60], combinatorics [24], spectral graph theory [1], network design and random processes [61]. In an instance of a determinant maximization problem, we are given a collection of vectors $U = \{v_1, \ldots, v_n\} \subset \mathbf{R}^d$, and a goal is to pick a subset $S \subseteq U$ of given vectors to maximize the determinant of the matrix $\sum_{i \in S} v_i v_i^\top$. The set S of picked vectors must satisfy additional combinatorial constraints such as cardinality constraint ($|S| \leq k$) or matroid constraint (S is a basis of a matroid defined on the vectors).

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 [13, 60, 23, 59].

Applications. Observe that when the number of vectors picked is exactly d, the objective is precisely the square of the volume of the parallelepiped spanned by the selected vectors. The problem of finding the largest volume parallelepiped in a collection of given vectors has been studied [62, 59, 63] for over three decades. Another interesting application is the determinantal point processes [61], where a probability distribution over subsets of vectors is defined. The probability of selecting a subset is defined to be proportional to

the squared volume of the parallelepiped defined by them. These distributions display nice properties of negative correlation. Finding sets with the largest probability mass is exactly the determinant maximization problem. We refer the reader to [1] for applications in experimental design and to [60] for application to fair allocations.

The computational complexity of the determinant maximization depends crucially on the combinatorial set family which constrains the set of feasible collection of vectors. The simplest constraint being the cardinality constraint, wherein the number of vectors is fixed, has been the most widely studied variant. For this, a variety of methods including convex programming based methods [13, 63, 62, 64], combinatorial methods - such as local search and greedy [59, 14, 2] - as well as close relationship to graph sparsification [13] have been exploited to obtain efficient approximation algorithms with very good guarantees. Overall, these results give a very clear understanding of the computational complexity of the problem.

The more general case when the combinatorial constraints are defined by a matroid constraint has recently received extensive focus [22, 60, 23, 24, 25]. This is especially interesting since some of the applications are naturally modeled as matroid constraints, in particular, as partition constraints. Unfortunately, there is a big gap between estimation algorithms and approximation algorithms in this case! Indeed, one can approximately *estimate* the value of an optimal solution with a good guarantee, however, *finding* such a solution is much more challenging, leading to an exponential loss in the approximation factor. For example, even for the special case of the partition matroid, there is an e^d-approximate estimation algorithm but the best known approximation algorithms return a solution with an approximation factor of $e^{O(d^2)}$, an exponential blow-up[1]. A fundamental reason for this gap is the reliance on the relationship between convex programming relaxations for the problem and the theory of stable polynomials and its generalization to strongly log-

[1] Since we are computing the determinant of $d \times d$ matrices, the exponent of d in the approximation factor is appropriate. Indeed, many works even consider the d^{th}-root of the determinant where the approximation factors are also the d^{th}-root of the above bounds.

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.

3.1 Summary of Results

In this work, we introduce new combinatorial methods for determinant maximization under a matroid constraint and give an $O(d^{O(d)})$-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 *There is a polynomial time algorithm which, given a collection of vectors $v_1, \ldots, v_n \in \mathbf{R}^d$ 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 on the $e^{O(d^2)}$-approximation algorithm which relies on the $e^{O(d)}$-estimation algorithm [24, 23, 25]. 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 $r \leq d$. In [20], the result was also generalized to the case 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 $sym_r(M)$ be the r^{th} elementary symmetric function of the eigenvalues of the $d \times d$ matrix M. Thus, our objective is to maximize $sym_r\left(\sum_{i \in S} v_i v_i^\top\right)$.

Theorem 20 *There is a polynomial-time algorithm which, given a collection of vectors $v_1, \ldots, v_n \in \mathbf{R}^d$ and a matroid $\mathcal{M} = ([n], \mathcal{I})$ of rank $r \leq d$, returns a set $S \in \mathcal{I}$ such that*

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

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, in [21], 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 21 *There is a polynomial time algorithm which, given a collection of vectors $v_1, \ldots, v_n \in \mathbf{R}^d$ 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 [25], which only gives an estimate of the optimum value and results in a $d^{O(d^2)}$-approximation algorithm.

3.2 Technical Overview

For intuition, let $vol(S)$ denote the volume of the parallelepiped spanned by the vectors in S. Then $vol(S)^2 = \det\left(\sum_{i \in S} v_i v_i^\top\right)$, for any $S \subseteq U$ with $|S| = d$, so we can think of $vol(S)$ as an equivalent objective function. First, observe that the feasibility problem of checking whether there is a set $S \in \mathcal{I}$ such that $vol(S) > 0$ can be reduced to matroid intersection. Indeed, the feasibility problem is equivalent to checking if there is a common

basis of the matroid $\mathcal{M}$ and the linear matroid defined by the vectors $\{v_1, \ldots, v_n\}$. Since we aim to maximize $vol(S)$ over all independent sets S, a natural approach is to use the weighted matroid intersection algorithm. Unfortunately, our weights are not linear, i.e., $vol(S)$ does not equal $\sum_{i \in S} w_i$ or log-linear $\prod_{i \in S} w_i$ for some weights w on the vectors. Nonetheless, the matroid intersection algorithm forms the backbone of our approach.

Overview of Matroid Intersection. Before we describe our algorithm, let us review a classical algorithm to find a maximum weight common basis of two matroids. We give a complete and detailed explanation in Section 2.1, and only a high-level picture here for brevity. Given $U = \{1, \ldots, n\}$, a weight function $w : U \to \mathbf{R}$ and two matroids $\mathcal{M}_1 = (U, \mathcal{I}_1)$ and $\mathcal{M}_2 = (U, \mathcal{I}_2)$, the goal is to find a common basis S of maximum weight $w(S) := \sum_{e \in S} w_e$. We assume that there exists a common basis of the two matroids. Consider the following simple algorithm that also introduces some of the basic ingredients necessary for our algorithm. The algorithm will take as an input a common basis S and either certify that S is a maximum weight common basis or return a new common basis $\widehat{S}$ of higher weight. To explain the algorithm, we recall the important concept of the *exchange graph*. Given the set S, we construct a directed bipartite graph $G(S)$ with bipartitions given by $U \setminus S$ and S. For any $u \in U \setminus S$ and $v \in S$, $G(S)$ contains an arc from u to v if $S - v + u$ is a basis in $\mathcal{M}_2$ and an arc from v to u if $S - v + u$ is a basis in $\mathcal{M}_1$. For convenience, we use $S - v + u$ to refer to the set $(S \cup \{u\}) \setminus \{v\}$. Moreover, give each vertex $u \in U \setminus S$ a weight $-w_u$ and each vertex $v \in S$ a weight of w_v. A nice fact from matroid theory is that S is a maximum weight basis if and only if there is no negative weight cycle in this directed graph (Chapter 41, Theorem 41.5 [55]). Moreover, if C is a directed negative weight cycle with minimum hops (hops here refers to the number of arcs of the cycle), then $S \Delta C$ forms a common basis of the two matroids whose weight is strictly larger than the weight of S. Thus, the algorithm finds a maximum weight basis by iteratively finding a negative weight cycle in such an exchange graph.

With the above algorithm as a guiding tool, we describe our algorithm. The two matroids are precisely the constraint matroid $\mathcal{M}$ and the linear matroid defined over the vectors. A first challenge is that the objective function $vol(S)^2 = \det\left(\sum_{i\in S} v_i v_i^\top\right)$ is not linear. Thus it is not possible to define the vertex weights as was done above. But a natural function to work with instead is the function $\log vol(S)$, which is known to be submodular. While we do not use submodularity explicitly, our algorithm takes linear approximations of this function at each iteration while searching for improvements as in the matroid intersection algorithm. In the case that $r \leq d$, we use the geometric relationship between vol and $\det$ closely, while we take a more algebraic perspective when $r \geq d$. The first new ingredient in our algorithm is to introduce *arc* weights rather than vertex weights in the exchange graph $G(S)$. Indeed for the forward arcs (u, v) for $u \notin S$ and $v \in S$ that correspond to the linear matroid, we introduce a weight of $-\log \frac{vol(S-v+u)}{vol(S)}$. We also introduce a weight of 0 for the backward arcs, which correspond to the arcs for the constraint matroid $\mathcal{M}$. The crucial observation is the following interpretation of the weight $\log \frac{vol(S-v+u)}{vol(S)}$: write the vector $u \notin S$ in the basis S, i.e. $u = \sum_{v\in S} a_{uv} v$, for some $a_{uv} \in \mathbb{R}$ for each $v \in S$. Then $\frac{vol(S-v+u)}{vol(S)} = |a_{uv}|$ (See Lemma 8). Such relationships between the ratio of volumes and coefficients in expressing the vectors in basis given by S play an important role.

Lemma 6 (Determinant to Cycle) *Let S be a basis of $\mathcal{M}$ and OPT be the basis of $\mathcal{M}$ maximizing $vol(OPT)$. If $vol(OPT) \geq e^{12d\log d} \cdot vol(S)$, then there exists a directed cycle C of 2ℓ hops for some $\ell > 0$ in the exchange graph $G(S)$ such that*

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

We call such a cycle an f-violating cycle. Observe that such a cycle can be found as a negative weight 2ℓ-hop cycle when weights are updated to $w_\ell(u, v) = \frac{1}{\ell}\log f(\ell) - \log|a_{uv}|$ for a forward arc (u, v) where $u \notin S$ and $v \in S$. The lemma relies on the following observation. Abusing notation slightly, let T and S be matrices whose columns are the

vectors in OPT and S, respectively. Writing each vector in OPT in the basis given by S we obtain $T = SA^\top$ for some matrix A. The condition in the lemma implies that $\det(A) \geq e^{12d \log d}$. Also observe that the weight of any (u, v) where $u \in OPT$ and $v \in S$ is exactly $-\log |a_{uv}|$ where a_{uv} is the $(u, v)^{\text{th}}$-entry in A. Combining these facts, we can show there exists a cycle satisfying the conditions of the lemma.

The next step in the algorithm is to find an f-violating cycle C and then update the solution to $T = S \Delta C$. Again, we relate the change in objective $vol(T)$ to the coefficients. While the entries $|a_{uv}|$ of the cycle are large, the objective of the new solution T depends not only on the weight of the edges of the cycle but the weight of all arcs between all vertices in $C \setminus S$ and $C \cap S$. Indeed, consider a square matrix B with rows and columns indexed by $C \setminus S$ and $S \cap C$ respectively with entry (u, v) as a_{uv}. Recall a_{uv} is the coefficient of vector v when u is expressed in basis S. Then $vol(T) = |\det(B)| \cdot vol(S)$ (Lemma 11). Thus it remains to lower bound the determinant of B. The entries on the diagonal of the matrix B exactly correspond to entries that define the weights of the forward arcs on the cycle C. Thus Lemma 6 implies that the product of the diagonal entries of B is large. In the next lemma, we show that if the cycle C is the *minimum hop* f-violating cycle, then we can in fact lower bound the determinant of the matrix.

Lemma 7 (Cycle to Determinant) *If C is a minimum hop f-violating cycle in the exchange graph $G(S)$, then $vol(S \Delta C) \geq 2 \cdot vol(S)$. Moreover, $S \Delta C$ is also a basis of $\mathcal{M}$.*

This lemma crucially uses the fact that C is a minimum hop f-violating cycle as in the case for matroid intersection algorithms. Indeed, off-diagonal entries of the matrix B correspond to arcs that form chords of the cycle C. The minimality of C allows us to show upper bounds on all the off-diagonal entries of the matrix B. A technical calculation then allows us to lower bound the determinant.

3.3 The case that rank equals dimension

In this section we will outline the algorithm which gives the following theorem.

Theorem 22 *There is a polynomial-time algorithm which, given a collection of vectors $v_1, \ldots, v_n \in \mathbf{R}^d$ and a matroid $\mathcal{M} = ([n], \mathcal{I})$ of rank d, returns a set $S \in \mathcal{I}$ such that*

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

We first show the algorithm and the analysis for a partition matroid with rank d. This allows us to show the basic ideas without going into the details of matroid theory. The generalizations to general matroid are quite standard. We detail them in subsection 3.3.3.

Consider a partition matroid $\mathcal{M}$ with d partitions $\mathcal{P}_1, \ldots, \mathcal{P}_d$, where each $\mathcal{P}_i$ contains n_i vectors $v_{i1}, \ldots, v_{in_i} \in \mathbf{R}^d$. Our goal is to find a set S which provides a good approximation to the objective

$$\max\left\{\det\left(\sum_{v\in S} vv^\top\right) : |S| = d, |S \cap \mathcal{P}_i| = 1 \ \forall i\right\}.$$

Let OPT denote the optimal solution set. The following theorem is a specialization of Theorem 1 to the case of partition matroid.

Theorem 23 *Given a partition matroid $\mathcal{M}$ with d parts, let OPT be the optimal solution to the determinant maximization problem on $\mathcal{M}$. Then, there is a polynomial-time deterministic algorithm that outputs a feasible set $S \in \mathcal{M}$ such that*

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

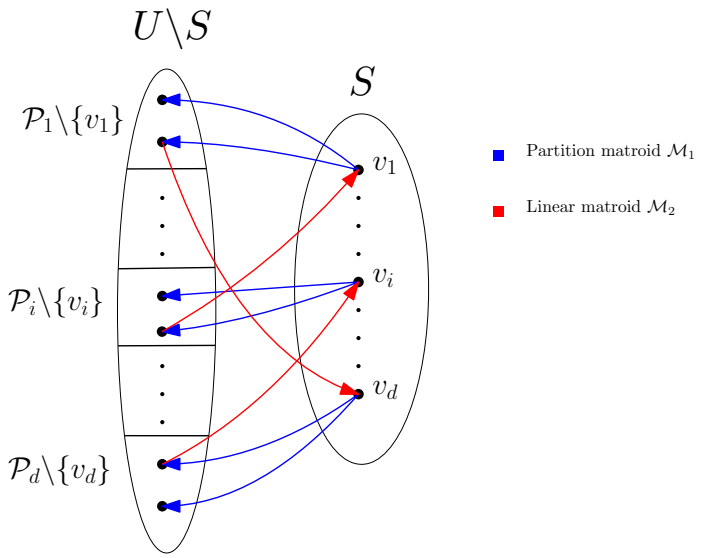


Figure 3.1: The exchange graph $G(S)$

3.3.1 Algorithm

We begin by formally defining the exchange graph, the different weight functions, and then the algorithm which helps establish Theorem 1 for the case of partition matroids.

Definition 3 (Exchange Graph) *For a subset of vectors $S = \{v_1, v_2, \ldots, v_d\}$ with $v_i \in \mathcal{P}_i$ for all i, we define the exchange graph of S, denoted by $G(S)$ as a bipartite graph, where the right-hand side consists of vectors in S, i.e., $R = \{v_1, v_2, \ldots, v_d\}$ and the left-hand side consists of all the vectors $L = \bigcup_{i=1}^{d} \mathcal{P}_i \backslash \{v_i\}$ (See Figure 3.1). Each $v_i \in R$ has an edge to every $u \in \mathcal{P}_i \backslash \{v_i\}$, i.e., all the vectors in the same part as v_i. The vertices on the left-hand side have forward edges to every vertex in S.*

We define a family of weight functions on the exchange graph. The basic weight function will be denoted by $w_0 : A(G(S)) \to \mathbf{R}$ and, in addition, we define weight functions w_i for each $1 \leq i \leq d$. To define these weights, we use the function $f : [d] \to \mathbb{Z}_+$ with $f(1) = 2$, and $f(i) = (i!)^{11}$ for all $i \geq 2$.

Definition 4 (Weight functions on the Exchange graph) *For the basic weight function w_0 all the backward arcs (from any $v_i \in S$ to every $u_j \in \mathcal{P}_i \backslash \{v_i\}$) have weight 0. For*

$u_j \in L$, let $u_j = \sum_{i=1}^{d} a_{ij} \cdot v_i$ be expression for u_j in the basis S where $a_{ij} \in \mathbb{R}$ for each i. Then the forward arc (u_j, v_i) has weight $w_0(u_j, v_i) := -\log(|a_{ij}|)$ for each $i \in [d]$ and each $u_j \in L$.

Now we define the weight function w_ℓ on the arcs for any $1 \leq \ell \leq d$. All backward arcs still have weight 0 but every forward edge (u, v) has weight $w_\ell(u, v) := \frac{\log(f(\ell))}{\ell} + w_0(u, v)$.

The following lemma gives the intuition behind the weight function w_0 defined above. It shows that the weight on arc (u_i, v_j) exactly measures the change in the objective when we replace element v_i with u_j in S. The shifted edge weights w_ℓ will be used to help find the minimal f-violating cycle, as will become clear soon.

Lemma 8 *Let S be a solution with $vol(S) > 0$ and $u \notin S$. Then for any $v \in S$, we have $w_0(u, v) = -\log \frac{vol(S+u-v)}{vol(S)}$.*

Proof Let $S = \{v_1, \ldots, v_d\}$ so that $v = v_d$ and write $u = \sum_{i=1}^{d} a_i v_i$. We can also write $v = v^\perp + \sum_{i=1}^{d-1} b_i v_i$ where $v^\perp$ is orthogonal to $S \backslash \{v\}$. Then $u = a_d v^\perp + \sum_{i=1}^{d-1} (a_d b_i + a_i) v_i$. For $X \subseteq \mathbb{R}^d$ with $|X| = k \leq d$, let $vol(X)$ denote the k-dimensional volume of the parallelepiped spanned by X; i.e. $vol(X) = \sqrt{\det(X^\top X}$, where the matrix $X^\top X$ is the $k \times k$ Gram matrix for the set X. By considering the Gram-Schmidt orthogonalization of S (or equivalently QR-decomposition) we can see that

$$vol(S) = vol(S - v) \cdot \|v^\perp\|.$$

For the same reason

$$vol(S + u - v) = vol(S - v) \cdot |a_1| \|v^\perp\|,$$

since the change in volume from adding a single new vector is proportional to the length of

the component of that vector which is orthogonal to our current set. Thus

$$-\log \frac{vol(S+u-v)}{vol(S)} = -\log \frac{vol(S-v)\cdot |a_{uv}|\|v^{\perp}\|}{vol(S-v)\cdot \|v^{\perp}\|} = -\log |a_{uv}|.$$

□

While we will be specific about which weight function to use, but if it is not specified, then we refer to the weight function w_0.

Definition 5 (Cycle Weight) *The weight of a cycle C in $G(S)$ is defined as $w_0(C) = \sum_{e\in C} w_0(e)$.*

Observe that the weight of a cycle depends only on the weight of the forward edges as backward edges have a weight 0.

We want to move from the current set S to a set with higher volume by exchanging on cycles in $G(S)$. But we want to exchange only on cycles that satisfy certain nice properties. For this purpose, we define f-Violating Cycles and Minimal f-Violating Cycles. The algorithm will always exchange on a Minimal f-Violating Cycle.

Definition 6 (f-Violating Cycle) *A cycle in $G(S)$ is called an f-violating cycle if*

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

where $|C|$ is the number of arcs in C.

We have the following simple observation regarding f-violating cycle.

Observation 1 *If C is a f-violating cycle then*

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

Proof If C is f-violating then $\ell(C) < -\log f(|C|/2)$, where $\ell(C)$ is the sum of the w_0 edge weights in C. By expanding $\ell(C)$ we see that

$$\begin{aligned}\ell(C) &= \sum_{(u,v)\in C: u\in L, v\in R} w_0(u,v) \\ &= -\log\left(\prod_{(u,v)\in C: u\in L, v\in R} |a_{uv}|\right).\end{aligned}$$

Since $\ell(C) < -\log f(|C|/2)$, we can take the exponential to remove the logarithms and attain the desired inequality. □

Definition 7 (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 all cycles C' such that $V(C') \subset V(C)$, C' is not an f-violating cycle.*

Note that finding an f-violating cycle with $2i$ arcs is equivalent to finding a negative cycle with $2i$ arcs in $G(S)$ with weights w_i. We use the following simple algorithm to find a minimal f-violating cycle in $G(S)$ (if one exists), where we iterate on the number of arcs in the cycle.

Algorithm 2 Finding minimal f-violating cycle

```
for i = 1, ..., d do
    if there is a cycle C with exactly 2i arcs in G(S) such that w_i(C) < 0 then
        Return C
    end if
end for
```

The following lemma is immediate.

Lemma 9 *Algorithm 2 finds the minimal f-violating cycle in $G(S)$.*

Proof In the ith iteration of Algorithm 2 we determine if there is a negative cycle in $G(S)$ with weights w_i and $2i$ hops, as follows. For each vertex of $G(S)$, we start an instance of Bellman-Ford (See Chapter 8, Section 8.3, [55]) with that vertex as the root, and proceed for $2i$ iterations. For source u, after $2i$ iterations, we check whether the distance from u to u is negative. If so, we have found a negative cycle with at most $2i$ hops. Note that for weights w_i, any negative cycle with at most $2i$ hops is an f-Violating cycle. Since the $(i-1)$-th iteration of the algorithm ensured that there are no f-Violating cycles with at most $2(i-1)$ hops, a negative cycle in the i-th iteration (if any) must have exactly $2i$ hops.

Suppose there is an f-violating cycle C in $G(S)$, so that $\ell = |C|/2$. Then, with weight w_ℓ, the total weight of the cycle C is

$$w_\ell(C) = \sum_{(u,v)\in C} w_\ell(u,v) = \sum_{(u,v)\in C} \log(f(\ell))/\ell + w_0(u,v) = \log f(\ell) + w_0(C).$$

Since C is f-violating we know that $\log f(\ell) < -w_0(C)$, so the above calculation shows that C has negative total weight with weights w_ℓ. This guarantees that Algorithm 2 will return an f-violating cycle whenever one exists.

Now suppose that C is the cycle returned by Algorithm 2 and we must show that C is minimal f-violating. Let C' be another cycle such that $V(C') \subset V(C)$. Then C' has fewer hops than C, but it was not returned in iteration $|C'|/2$, so we know that C' must not be f-violating. Thus C is indeed minimal. □

After finding a minimal f-violating cycle, C, we modify the current set S to $S\Delta C$ and repeat. Observe that $S\Delta C$ is always a feasible set as it will pick exactly one element from each part. The main idea is that if $vol(S)$ is small compared to $vol(OPT)$, i.e., $vol(S) < vol(OPT) \cdot e^{-\Omega(d\log(d))}$, then there is always an f-violating cycle in $G(S)$ (see Lemma 10). Moreover, if C is a minimal f-violating cycle, then $vol(S\Delta C) \geq 2 \cdot vol(S)$ (see Lemma 12). If we initialize S to any solution with non-zero determinant, then the ratio $vol(OPT)/vol(S)$ is at most $2^{4\sigma}$ where σ is the encoding length of our problem

input (Chapter 3, Theorem 3.2 [65]). This implies that we need only modify the set S polynomially many times before $vol(S)$ becomes greater than $vol(OPT) \cdot e^{-O(d \log(d))}$, which gives Theorem 23. Such an initialization can be obtained by finding a basis of $\mathbb{R}^d$ that picks exactly one vector from each part. As discussed above, this problem can be solved by the matroid intersection algorithm over the partition matroid and the linear matroid defined by the vectors.

Algorithm 3 Algorithm to find an approximation to OPT

$S \leftarrow$ set with $|S| = d$, $|S \cap \mathcal{P}_i| = 1$ for all i, and $vol(S) > 0$.

while there exists an f-violating cycle in $G(S)$ **do**

$C =$ minimal f-violating cycle in $G(S)$

$S = S \Delta C$

end while

Return S

Lemma 10 *For any set S with $|S| = d$ and $vol(S) > 0$, if $vol(S) < e^{-12d \log(d)} \cdot vol(OPT)$, then there exists an f-violating cycle in $G(S)$.*

Proof Let $OPT = \{u_1, u_2, \ldots, u_d\}$ and $S = \{v_1, v_2, \ldots, v_d\}$ such that $u_i, v_i \in \mathcal{P}_i$ for all $i \in [d]$. Observe that (v_i, u_i) is an arc in the exchange graph for each i since u_i and v_i belong to the same part, given $u_i \neq v_i$.

Abusing notation slightly, let V_T and V_S be matrices whose columns are the vectors in OPT and S, respectively. Let A be the coefficient matrix of T with respect to S, i.e., $V_T = V_S A^\top$. Then

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

Let $X = OPT \backslash S$, $Y = S \backslash OPT$, and $|X| = |Y| = k$. Without loss of generality, let

$Y = \{v_1, \ldots, v_k\}$ and $X = \{u_1, \ldots, u_k\}$. Then

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

where A_k is the sub-matrix of A corresponding to rows in X and columns in Y. Then $\det(A) = \det(A_k)$.

As per the hypothesis in the lemma, we have $\det(V_S V_S^\top) < \det(V_T V_T^\top) \cdot e^{-24d\log(d)}$. Therefore,

$$|\det(A_k)| > e^{12d\log(d)} \geq e^{12k\log(k)}. \tag{3.1}$$

By the Leibniz formula, we have $\det(A_k) = \sum_{\sigma \in \mathcal{S}_k} \operatorname{sign}(\sigma) \prod_{i=1}^{k} a_{i\sigma(i)}$. Taking absolute values gives $|\det(A_k)| \leq \sum_{\sigma \in \mathcal{S}_k} \prod_{i=1}^{k} |a_{i\sigma(i)}|$. Since $|\mathcal{S}_k| = k! \leq e^{k\log(k)}$, there exists a permutation $\sigma \in \mathcal{S}_k$ such that

$$\prod_{i=1}^{k} |a_{i\sigma(i)}| > |\det(A_k)| \cdot e^{-k\log(k)} \geq e^{11k\log(k)}. \tag{3.2}$$

Let the cycle decomposition of this σ be $\sigma = \{C_1, C_2, \ldots, C_\ell\}$. Then each C_j corresponds to a unique cycle in $G(S)$ with $2|C_j|$ hops by considering the forward arcs $(u_i, v_{\sigma(i)})$ for each i on the cycle and the backward arcs (v_i, u_i) for each i in C_j. We claim that at least one of these cycles is an f-violating cycle. If not, then by the definition of f-violating cycles, we have $\prod_{i \in C_j} |a_{i\sigma(i)}| \leq f(|C_j|/2)$. Multiplying over all cycles in σ gives

$$\prod_{i=1}^{k} |a_{i\sigma(i)}| = \prod_{j=1}^{\ell} \prod_{i \in C_j} |a_{i\sigma(i)}| < \prod_{j=1}^{\ell} f(|C_i|/2) < e^{11k\log(k)},$$

where the last inequality follows from $\sum_{j=1}^{\ell} |C_j| = k$. This contradicts (3.2), so $G(S)$ must contain an f-violating cycle. □

The requirement in Lemma 10 that $vol(S) < vol(OPT) \cdot e^{-12d\log(d)}$ is tight, up to the

coefficient in the exponent. Consider the case where d is a power of two (or more generally, any d for which a Hadamard matrix of order d is known to exist), $S = \{e_1, \ldots, e_d\}$ consists of the standard basis vectors, and $L = H = \{h_1, \ldots, h_d\}$ consists of the columns of the $d \times d$ Hadamard matrix. The entries of H are all ± 1, and $h_i^\top h_j = 0$ for $i \neq j$. Then $vol(S) = 1$, and the optimal solution is $OPT = H$, which has objective value

$$vol(H) = \prod_{i=1}^{d} \|h_i\| = d^{d/2} = e^{\frac{d}{2}\log(d)} \cdot vol(S),$$

since the vectors in H are orthogonal. Meanwhile, the exchange matrix in this case is $A = H^\top$. Since all the entries of A are ± 1, we know that the product of the entries along any cycle will have an absolute value of 1. Thus, we cannot find an f-violating cycle in the same way, despite the fact that $vol(S) \leq vol(OPT) \cdot e^{-\frac{d}{2}\log(d)}$.

3.3.2 Cycle Exchange and Determinant

Now we show that exchanging on a minimal f-violating cycle C increases the objective of the output set by at least a factor of two. The proof relies on two technical lemmas. First, observe that the arc weights given by $w_0(u, v)$ are exactly how much the objective will change if switch from the solution S to $S + u - v$ in the solution. But switching on a cycle will switch multiple elements at the same time. Since our function $vol(\cdot)$ (or more appropriately $\log vol(\cdot)$) is not additive, it is not clear what the total change in the objective will be. The following lemma characterizes exactly how the objective changes when we switch a large set.

Consider our current solution S. Let C be the minimal cycle found, and $\ell = |C|/2$. Let $X = C \cap L$ and $Y = C \cap S$. Thus, the updated set will be $T = (S \cup X) \setminus Y$. We will use the notation V_X, V_Y and V_S to represent the matrices whose columns are the vectors in their respective sets. Note that V_S is $d \times d$ while both V_X and V_Y are $d \times \ell$. Observe that $vol(S)^2 = \det(V_S V_S^\top)$ and $vol(T)^2 = \det(V_T V_T^\top) = \det(V_S V_S^\top + V_X V_X^\top - V_Y V_Y^\top)$.

Crucially, we show that the matrix consisting of coefficients a_{uv} that define the weights on the arcs of the exchange graph for $u \in X$ and $v \in Y$ also defines the change in objective value.

Lemma 11 *Let S be a basis, let X and Y be sets with $|X| = |Y| = \ell$ and $Y \subseteq S$. Let A be the $\ell \times d$ matrix of coefficients so that $V_X = V_S A^\top$, 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 $vol(T)^2 = vol(S)^2 \cdot \det(A_C A_C^\top)$.*

Proof Order the columns of V_S so that V_Y makes up the first ℓ columns of V_S. Let A' be the $\ell \times (d-\ell)$ submatrix of A consisting of the remaining columns not already in A_C. Then

$$V_T = V_S \begin{bmatrix} A_C & A' \\ 0 & I_{d-\ell} \end{bmatrix}^\top,$$

which implies that

$$\det(T) = \det(S) \cdot \det(A_C).$$

□

Without loss of generality, let $C = (v_0 \to u_1 \to v_1 \to u_2 \to v_2 \to \dots u_\ell \to v_0)$ so that $X = \{u_1, \dots u_\ell\}$ and $Y = \{v_1, \dots, v_{\ell-1}, v_0\}$, and order the columns of A_C accordingly so that the ℓ-th column corresponds to v_0. Observe that diagonal entries of the A_C correspond to coefficient of v_i when expressing u_i in basis of S and thus equals a_{ii}. C being f-violating implies that the product of the diagonal entries $\prod_{i=1}^{\ell} |a_{ii}| > f(\ell)$. To show that the volume of T is large, we need to show $|\det(A_C)|$ is large. To this end, we utilize crucially that C is the *minimal* f-violating cycle. Observe that the off-diagonal entries a_{ij} exactly correspond to the weight on chords of the cycle. Since each chord introduces a cycle with smaller number of arcs, by minimality we know that it is not f-violating. This allows us to prove upper bounds on the off-diagonal entries of the matrix A_C. Finally, a careful argument

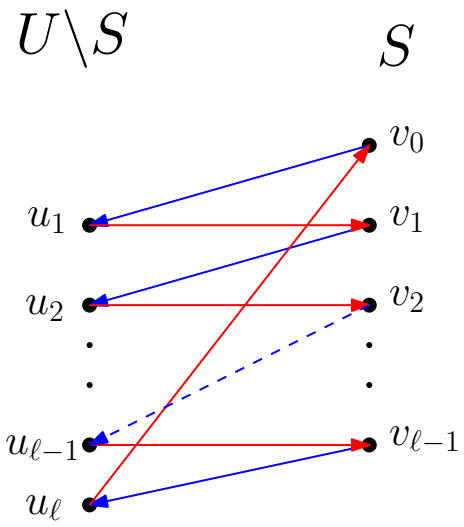


Figure 3.2: The cycle C

allows us to give a lower bound on the determinant of any matrix with such bounds on the off-diagonal entries. We now expand on the above outline below.

Lemma 12 *If C is a minimal f-violating cycle in $G(S)$, then $vol(S\Delta C) \geq 2 \cdot vol(S)$.*

Proof Let $C = (v_0 \to u_1 \to v_1 \to u_2 \to v_2 \to \ldots u_\ell \to v_0)$ where v_i, u_{i+1} belong to the same part and $v_i \in S$ (See Figure 3.2).

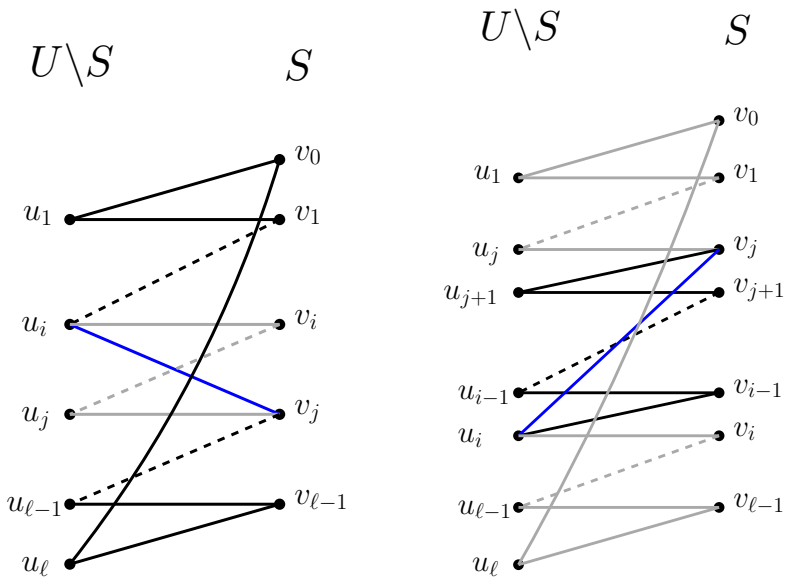
By Lemma 11, we know that

$$vol(S\Delta C) = \det(A_C A_C^\top)^{1/2} \cdot vol(S) = |\det(A_C)| \cdot vol(S).$$

We will index the entries of A_C according to the indices of u_i and v_j where the last column corresponds to v_0. Since C has 2ℓ hops, A_C is an $\ell \times \ell$ matrix.

We now bound each entry of the matrix A_C in terms of its diagonal entries, $a_{i,i}$ for $i = 1, \ldots, n$. We show upper bounds on the absolute value of each entry as a function of the diagonal entries. If we were to add the edge from u_i to v_j this would create additional subcycles, and the fact that C is a minimal f-violating cycle will give an upper bound on the weight of edge (u_i, v_j) in terms of f, and the other existing edges on the cycle.

Consider the i, j-th entry of A_C. For $i > j$, define the cycle $C_{i,j} := (u_i \to v_j \to$



(a) Shortcut when $i < j$ (b) Sub-cycle when $i > j$

Figure 3.3: Structure when the edge $u_i \to v_j$ (in blue) is added

$u_{j+1} \to v_{j+1} \to \ldots v_{i-1} \to u_i$). $C_{i,j}$ is a cycle with $2(i-j)$ hops and $V(C_{i,j}) \subset V(C)$. C being a minimal f-violating cycle implies that $C_{i,j}$ is not an f-violating cycle. Therefore, $e^{-w_0(C_{i,j})} = |a_{i,j}| \cdot \prod_{s=j+1}^{i-1} |a_{s,s}| < f(i-j)$. This implies

$$|a_{i,j}| < \frac{f(i-j)}{\prod_{s=j+1}^{i-1} |a_{s,s}|}. \tag{3.3}$$

Similarly for $j = \ell$, we have $|a_{i,\ell}| < \frac{f(i)}{\prod_{s=1}^{i-1} |a_{s,s}|}$.

For $i < j < \ell$, define $C'_{i,j} := (v_0 \to u_1 \to v_1 \to \ldots u_i \to v_j \to u_{j+1} \to \ldots u_\ell \to v_0)$. Again, $C'_{i,j}$ is a cycle with $2(\ell - j + i)$ hops which is not f-violating. Therefore,

$$|a_{i,j}| \cdot \prod_{s=1}^{i-1} |a_{s,s}| \cdot \prod_{s=j+1}^{\ell} |a_{s,s}| < f(\ell - j + i). \tag{3.4}$$

Since C is an f-violating cycle, we also have

$$\prod_{s=1}^{\ell} |a_{s,s}| > f(\ell). \tag{3.5}$$

Combining (3.4) and (3.5) gives

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

Let B_ℓ be the matrix obtained by applying the following operations to A_C

- Multiply the last column by $a_{1,1}$ and for $j < \ell$, divide the j-th column by $\prod_{s=2}^{j} a_{s,s}$
- Divide the first row by $a_{1,1}$ and for $i > 1$, multiply the i-th row by $\prod_{s=2}^{i-1} a_{s,s}$
- Divide the last column by $f(\ell)$ and, if needed, flip the sign of the last column so that $a_{\ell,\ell} > 0$.

The scaling from the first row and last column cancel each other, and otherwise the the row scaling on row k cancels with the column scaling on column $k-1$. Thus $|\det(A_C)| = f(\ell) \cdot |\det(B_\ell)|$. We can also bound the entries of B_ℓ in terms of only ℓ and the function f.

- $b_{1,\ell} = a_{a,\ell}/f(\ell)$, so $|b_{i,\ell}| \le f(1)/f(\ell)$,
- $b_{i,\ell} = a_{i,\ell} \cdot a_{1,1} \cdot \prod_{s=2}^{i-1} a_{s,s}/f(\ell)$, so $|b_{i,\ell}| \le f(i)/f(\ell)$ for $1 < i < \ell$,
- $b_{1,j} = a_{1,j} \cdot a_{1,1}^{-1} \cdot \left(\prod_{s=2}^{j} a_{s,s}\right)$, so $|b_{1,j}| \le f(\ell - j + 1)/f(\ell)$ for $1 < j < n$, and otherwise
- $b_{i,j} = a_{i,j} \cdot \left(\prod_{s=2}^{i-1} a_{s,s}\right) \cdot \left(\prod_{s=2}^{j} a_{s,s}\right)$, so when $1 < i < j < \ell$ we have

$$|b_{ij}| \le |a_{i,j}| \cdot \left(\prod_{s=i}^{j} |a_{s,s}|\right)^{-1} \le f(\ell - j + i)/f(\ell),$$

  and when $1 < j < i < \ell$ we have

$$|b_{i,j}| \le |a_{i,j}| \cdot \left(\prod_{s=j+1}^{i-1} a_{s,s}\right) \le f(i - j).$$

In summary,

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

Because of the cancellations in the determinant, and the fact that we only have upper bounds on the entries of B_ℓ, it is difficult to directly bound the determinant. Since our goal is to show that the determinant is close to the product of the diagonal entries, we can instead consider the permanent, which will be easier to bound. To see this, first recall that

$$\det(M) = \sum_{\sigma \in \mathcal{S}_n} (-1)^{\operatorname{sgn}(\sigma)} \prod_{i=1}^{n} m_{i,\sigma(i)},$$

so we can get the lower bound

$$|\det(M)| \geq \prod_{i=1}^{n} |m_{ii}| - \sum_{\sigma \neq \mathrm{id}} \prod_{i=1}^{n} |m_{i,\sigma(i)}|.$$

Conversely,

$$\operatorname{perm}(M) = \sum_{\sigma \in \mathcal{S}_n} \prod_{i=1}^{n} m_{i,\sigma(i)},$$

without the signs, so we get the upper bound

$$|\operatorname{perm}(M)| \leq \prod_{i=1}^{n} |m_{ii}| + \sum_{\sigma \neq \mathrm{id}} \prod_{i=1}^{n} |m_{i,\sigma(i)}|.$$

Thus, a tight-enough upper bound on the permanent will also result in a lower bound on the determinant as well.

We have reduced the problem to calculating the permanent of a $\ell \times \ell$ matrix with fixed entries, which can be done more-of-less by direct calculation after decomposing the definition into different classes of permutations. The result of this calculation is summarized in Lemma 14 which states that $\operatorname{perm}(B_\ell) \leq (1 + 0.05/\ell)$; we leave the proof to Section 3.4. Finally, we conclude that $\det(A_C) \geq f(\ell) \cdot (1 - 0.05/\ell) \geq 2$. □

3.3.3 Update Step for general matroids

Consider the case when $\mathcal{M} = ([n], \mathcal{I})$ is a general matroid of rank d. When we exchange on a cycle C and update $S \leftarrow S\Delta C$, the resulting set is guaranteed to be independent in the linear matroid because of the determinant bounds in Lemma 12, but it is not clear that it would be independent in the general constraint matroid, $\mathcal{M}$, when $\mathcal{M}$ is not a partition matroid. However, by exchanging on a minimal f-violating cycle in our algorithm, we can make the same guarantee.

In this section, we prove the existence of an f-violating cycle for any matroid $\mathcal{M}$ with rank d when the current basis S is sufficiently smaller in volume than the optimal solution OPT. We also prove that exchanging on a minimal f-violating cycle preserves independence in $\mathcal{M}$.

Theorem 24 *For any basis S with $|S| = d$ and $vol(S) > 0$, if $vol(S) < vol(OPT) \cdot e^{-12d\log(d)}$, then there exists an f-violating cycle in $G(S)$.*

Proof Since S and OPT are independent and $|S| = |OPT|$, by Theorem 10 (Chapter 39, Corollary 39.12a, [55]) there exists a perfect matching between $OPT \backslash S$ and $S \backslash OPT$ using the backward arcs in $G(S)$. Let $X = OPT \backslash S$, $Y = S \backslash OPT$, and $|X| = |Y| = k$. Without loss of generality, let $Y = \{v_1, \ldots, v_k\}$ and $X = \{u_1, \ldots, u_k\}$ such that $(v_i \rightarrow u_i)$ is an arc in $G(S)$ for all $i \in [k]$.

Let V_T and V_S be matrices whose columns are the vectors in OPT and S, respectively. Let A be the coefficient matrix of T with respect to S, i.e., $V_T = V_S A^\top$. Then

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

where A_k is the sub-matrix of A corresponding to rows in X and columns in Y. Then by

the same proof as in Lemma 10, there exists a permutation $\sigma \in \mathcal{S}_k$ such that

$$\prod_{i=1}^{k} |a_{i\sigma(i)}| > |\det(A)| \cdot e^{-k\log(k)} \geq e^{11k\log(k)}. \tag{3.6}$$

Let the cycle decomposition of σ be $\sigma = \{C_1, C_2, \ldots, C_\ell\}$ where $C_i = (i_1 \to i_2 \to \ldots i_j \to i_1)$. Since there is an edge from $v_{\sigma(j)}$ to $u_{\sigma(j)}$ for all j, every cyclic permutation C_i corresponds to a cycle $(u_{i_1} \to v_{i_2} \to u_{i_2} \to v_{i_3} \ldots \to u_{i_j} \to v_{i_1} \to u_{i_1})$ in $G(S)$. We claim that at least one of these cycles is an f-violating cycle. If not, then by the definition of f-violating cycles, we have $\prod_{i\in C_j} |a_{i\sigma(i)}| \leq f(|C_j|/2)$ for all $j \leq \ell$. Multiplying over all the cycles in σ gives

$$\prod_{i=1}^{k} |a_{i\sigma(i)}| = \prod_{j=1}^{\ell} \prod_{i\in C_j} |a_{i\sigma(i)}| \leq \prod_{j=1}^{\ell} f(|C_j|/2) < e^{11k\log(k)},$$

where the last inequality follows from $\sum_{j=1}^{\ell} |C_j| = k$. This contradicts (3.6), so $G(S)$ must contain an f-violating cycle. □

Now we show that the update step maintains independence in the constraint matroid. The central idea is the same as in the proof of Lemma 1, but adapted to the setting of edge weights instead of vertex weights.

Lemma 13 *If C is a minimal f-violating cycle in $G(S)$, then $S\Delta C$ is independent in $\mathcal{M}$.*

Proof For clarity, let $V(C)$ denote the vertex set of C. Let $T := S\Delta V(C)$ and let $|C| = 2\ell$. Let us consider the graph $G(S)$ with weights w_ℓ, and define $w_\ell(D) := \sum_{e\in D} w_\ell(e)$ for any cycle D. Since C is an f-violating cycle, $w_\ell(C) = w_0(C) + \log(f(\ell)) < 0$.

Let the set of backward arcs in C be N_1, and the set of forward arcs be N_2. For the sake of contradiction, assume that $T \notin \mathcal{I}$. Then, by the converse of Theorem 11 (Chapter 39, Theorem 39.13, [55]), there exists a perfect matching N_1' on $V(C)$ consisting of only backward arcs such that $N_1 \neq N_1'$. Let A be a multiset of arcs consisting two copies of

each arc in N_2 and one copy of each arc in N_1 and N_1' (with arcs in $N_1 \cap N_1'$ appearing twice). Consider the directed graph $D = (V(C), A)$. Each vertex of $V(C)$ has exactly two incoming and two outgoing arcs in D. Moreover, since $N_1' \neq N_1$ there is an edge of N_1 which creates a chord across the copy of C in D, and so there is a directed circuit C_1 with $V(C_1)$ a proper subset of $V(C)$. As D is Eulerian we can extend this to a decomposition of its arcs into directed circuits $C_1, \ldots, C_k$. Since only arcs in N_2 have non-zero weights, we have $\sum_{i=1}^{k} w_\ell(C_i) = 2w_\ell(C)$.

Because $V(C_1)$ is a proper subset of $V(C)$, at most one cycle C_j can have $V(C_j) = V(C)$. If for some j, $V(C) = V(C_j)$, then $w_\ell(C_j) = w_\ell(C)$ as C_j must contain every edge in N_2. So, $\sum_{i \neq j} w_\ell(C_i) = w_\ell(C) < 0$ and there exists a cycle C_i such that $V(C_i)$ is a proper subset of $V(C)$ and $w_\ell(C_i) < 0$. Otherwise $V(C_j)$ is a proper subset of $V(C)$ for all j and $\sum_i w_\ell(C_i) = 2w_\ell(C) < 0$. Again, there exists a cycle C_i such that $V(C_i)$ is a proper subset of $V(C)$ and $w_\ell(C_i) < 0$.

Let C' be the directed cycle such that $V(C')$ is a proper subset of $V(C)$ and $w_\ell(C') \leq w_\ell(C) < 0$. Define $y := |C'|/2$. Thus $w_\ell(C') = y \cdot \log(f(\ell))/\ell + w_0(C') < 0$. Since $y < \ell$, $\log(f(y))/y \leq \log(f(\ell))/\ell$. Therefore $w_0(C') \leq -y \cdot \log(f(\ell))/\ell \leq -\log(f(y))$. So C' is an f-violating cycle with $V(C') \subset V(C)$, which contradicts the fact that C is a minimal f-violating cycle. □

3.4 A Permanent Calculation

The following lemma calculates a bound on the permanent of a particular numeric matrix, and is a key component in proving that the objective increases by a constant factor in every iteration for all the determinant maximization algorithms.

Lemma 14 *Let $B_\ell \in \mathbf{R}^{\ell \times \ell}$ satisfy the following properties*

- $b_{i,i} = 1$ *for all* $i \in [\ell]$,
- $0 \leq b_{i,j} \leq 2 \cdot f(i-j)$ *for all* $j < i \leq \ell$, *and*

- $0 \le b_{i,j} \le 2 \cdot f(\ell - j + i)/f(\ell)$ *for all* $i < j \le \ell$.

Then the permanent of B_ℓ *is bounded close to one:* $\mathrm{perm}(B_\ell) \le (1 + 0.05/\ell)$.

Before jumping into the proof we will begin with an overview of the approach, and review some background on counting permutations.

Our goal is to show that the permanent of B_ℓ is close to the product of its diagonal entries (which in this case is always 1). We can write the permanent as a sum over permutations

$$\mathrm{perm}(B_\ell) = \sum_{\sigma \in \mathcal{S}_\ell} \prod_{i=1}^{\ell} b_{i,\sigma(i)} = 1 + \sum_{\sigma \in \mathcal{S}_\ell \setminus \{\mathrm{id}_\ell\}} \prod_{i=1}^{\ell} b_{i,\sigma(i)},$$

so it suffices to show that $\prod_{i=1}^{\ell} b_{i,\sigma(i)}$ is very small for all non-identity permutations σ. Of course, we cannot address all such permutations individually, so we will divide them into categories, and provide a uniform bound for each subset of permutations. Some of these subsets will have larger upper bounds, but we will also show that there are fewer such permutations.

Concretely, we categorize all permutations in $\mathcal{S}_\ell \setminus \{\mathrm{id}_\ell\}$ based on the number of *fixed points* and *exceedances*. The set of fixed points of a permutation $\sigma \in \mathcal{S}_\ell$ is defined as $\{i \in [\ell] : \sigma(i) = i\}$ and the exceedance of σ is defined as the number of indices i such that $\sigma(i) > i$ (for more details, see Definition 8 and Lemma 15). Let $\mathcal{S}_\ell(n, k)$ denote the subset of $\mathcal{S}_\ell$ with $\ell - n$ fixed points and k exceedances (see Figure 3.4).

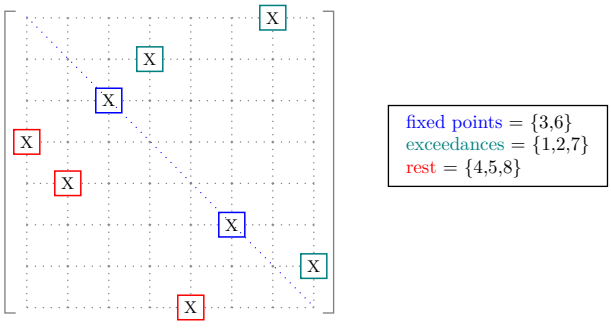


Figure 3.4: A permutation and corresponding index partition in $\mathcal{S}_8(6, 3)$.

By definition, all permutations in $\mathcal{S}_\ell \backslash \{\mathrm{id}_\ell\}$ have at most $\ell - 2$ fixed points and at least 1 exceedance. So we can further expand

$$\mathrm{perm}(B_\ell) = 1 + \sum_{n=2}^{\ell} \sum_{k=1}^{n-1} \sum_{\sigma \in \mathcal{S}_\ell(n,k)} \prod_{i=1}^{\ell} b_{i,\sigma(i)}.$$

Using the upper bounds on the entries of B_ℓ we can see that for all permutations in $\mathcal{S}_\ell(n,k)$ the term $\prod_{i=1}^{\ell} b_{i,\sigma(i)}$ will have a similar form.

Claim 3.4.1 *For every permutation $\sigma \in \mathcal{S}_\ell(n,k)$ there exist integers $x_1, \ldots, x_n$ with $1 \leq x_i \leq \ell - 1$ for all i, and $\sum_{i=1}^{n} x_i = \ell \cdot k$ such that*

$$\prod_{i=1}^{\ell} b_{i,\sigma(i)} \leq 2^n \cdot \frac{\prod_{i=1}^{n} f(x_i)}{f(\ell)^k}.$$

Proof For a permutation $\sigma \in \mathcal{S}_\ell(n,k)$,

$$\begin{aligned}\prod_{i=1}^{\ell} b_{i,\sigma(i)} &= \prod_{i=\sigma(i)} b_{i,i} \cdot \prod_{i>\sigma(i)} b_{i,\sigma(i)} \cdot \prod_{i<\sigma(i)} b_{i,\sigma(i)} = \prod_{i>\sigma(i)} b_{i,\sigma(i)} \cdot \prod_{i<\sigma(i)} b_{i,\sigma(i)} \\ &\leq \prod_{i>\sigma(i)} 2f(i-\sigma(i)) \cdot \prod_{i<\sigma(i)} \frac{2f(\ell - \sigma(i) + i)}{f(\ell)}.\end{aligned}$$

Since $\sum_{i=1}^{\ell} i - \sigma(i) = 0$

$$\sum_{i>\sigma(i)} i - \sigma(i) + \sum_{i<\sigma(i)} \ell - \sigma(i) + i = \ell \cdot |\{i : \sigma(i) > i\}| = \ell \cdot k.$$

Therefore, if $\sigma \in \mathcal{S}_\ell(n,k)$, then there exist integers $x_1, x_2, \ldots, x_n$ with $1 \leq x_i \leq \ell - 1$ for all i, $\sum_{i=1}^{n} x_i = \ell \cdot k$, such that

$$\prod_{i=1}^{\ell} b_{i,\sigma(i)} \leq 2^n \cdot \frac{\prod_{i=1}^{n} f(x_i)}{f(\ell)^k}.$$

□

Using the convexity properties of the function f, we are able to provide more concrete upper bounds depending on the values of k and n.

Claim 3.4.2 *If $\sigma \in \mathcal{S}_\ell(n,k)$ then*

$$\prod_{i=1}^{\ell} b_{i,\sigma(i)} \leq \begin{cases} 2^{2n-k-1} \cdot \frac{((2k-n+1)!)^{11}}{\ell^{11k}} & \text{if } k \geq n/2, \\ 2^{2n-k} \cdot \frac{1}{\ell^{11(k-1)} \cdot (\ell \cdot (\ell-1) \ldots (\ell-n+2k))^{11}} & \text{if } k < n/2. \end{cases}$$

Proof By Claim 3.4.1 there are integers $x_1, \ldots, x_n$ with $1 \leq x_i \leq \ell - 1$ for all i, and $\sum_{i=1}^{n} x_i = \ell \cdot k$ such that

$$\prod_{i=1}^{\ell} b_{i,\sigma(i)} \leq 2^n \cdot \frac{\prod_{i=1}^{n} f(x_i)}{f(\ell)^k}\,.$$

In order to maximize the upper bound, we only need to maximize the product $\prod_{i=1}^{n} f(x_i)$ in the numerator. The function f has the property that if $a \geq b$ then $f(a)f(b) \leq f(a+1)f(b-1)$, so be repeatedly applying this inequality we can see that the maximum is attained when as many of the x_i as possible are at the upper bound $\ell - 1$.

If $k \geq n/2$, since $x_1, \ldots, x_n$ satisfy $\sum_{i=1}^{n} x_i = \ell \cdot k$, and $1 \leq x_i \leq \ell - 1$, then $\prod_{i=1}^{n} f(x_i)$ is maximized when $x_1 = \ldots = x_k = \ell - 1$, $x_{k+1} = 2k - n + 1$, and $x_{k+2} = \ldots = x_n = 1$. Similarly, when $k < n/2$, $\prod_{i=1}^{n} f(x_i)$ is maximized when $x_1 = \ldots = x_{k-1} = \ell - 1$, $x_k = \ell - n + 2k - 1$, and $x_{k+1} = \ldots = x_n = 1$. So,

$$\prod_{i=1}^{n} f(x_i) \leq \begin{cases} f(\ell-1)^k \cdot f(2k-n+1) \cdot f(1)^{n-k-1} & \text{if } k \geq n/2 \\ f(\ell-1)^{k-1} \cdot f(\ell-n+2k-1) \cdot f(1)^{n-k} & \text{otherwise.} \end{cases}$$

Plugging these bounds in, we get

$$\prod_{i=1}^{\ell} b_{i,\sigma(i)} \leq \begin{cases} 2^n \cdot \frac{f(\ell-1)^k \cdot f(2k-n+1) \cdot f(1)^{n-k-1}}{f(\ell)^k} & \text{if } k \geq n/2 \\ 2^n \cdot \frac{f(\ell-1)^{k-1} \cdot f(\ell-n+2k-1) \cdot f(1)^{n-k}}{f(\ell)^k} & \text{if } k < n/2\,. \end{cases}$$

Using the definition of f, for any $k \geq n/2$,

$$\prod_{i=1}^{\ell} b_{i,\sigma(i)} \leq 2^{2n-k-1} \cdot \frac{((2k-n+1)!)^{11}}{\ell^{11k}},$$

and for $k < n/2$,

$$\prod_{i=1}^{\ell} b_{i,\sigma(i)} \leq 2^{2n-k} \cdot \frac{1}{\ell^{11(k-1)} \cdot (\ell \cdot (\ell-1) \ldots (\ell-n+2k))^{11}}.$$

□

To fully use these upper bounds, we also need to understand how many permutations are in $\mathcal{S}_\ell(n,k)$ for each n and k.

Definition 8 (Exceedances, Eulerian numbers, and Derangement numbers)

- *The exceedance of a permutation $\sigma \in \mathcal{S}_n$ is defined as $|\{i \in [n-1] : \sigma(i) > i\}|$.*
- *The Eulerian number $E(n,k)$ is defined to be the number of permutations in $\mathcal{S}_n$ with $k-1$ exceedances.*
- *The derangement number $T(n,k)$ is defined to be the number of derangements in $\mathcal{S}_n$ with k exceedances.*

The explicit formula for $E(n,k)$ is $E(n,k) = \sum_{j=0}^{k+1}(-1)^j \binom{n+1}{j} \cdot (k+1-j)^n$ (page 273, [66]). The exponential generating function of $E(n,k)$ is given by (page 273, [66])

$$\sum_{n=0}^{\infty} \sum_{k=0}^{n} E(n,k)\ t^k\ \frac{x^n}{n!} = \frac{t-1}{t-e^{(t-1)x}}\,. \tag{3.7}$$

The exponential generating function of $T(n,k)$ is given by (Proposition 5, [67])

$$\sum_{n=0}^{\infty} \sum_{k=1}^{n-1} T(n,k)\ t^k\ \frac{x^n}{n!} = e^{-t} \cdot \frac{t-1}{t-e^{(t-1)x}}\,. \tag{3.8}$$

Comparing (3.7) *and* (3.8)*, we infer*

$$T(n,k) = \sum_{j=0}^{n} (-1)^{(n-j)} \cdot \binom{n}{j} \cdot E(j,k)\,. \tag{3.9}$$

Now we can combine our upper bounds on the product of the matrix entries along the permutations, with the counts of the number of permutations with a given number of fixed points and exceedances to get a complete upper bound on the determinant.

Lemma 15 *$\mathcal{S}_\ell(n,k)$ denote the set of permutations on $[\ell]$ with $\ell - n$ fixed points and k exceedances. Then*

$$|\mathcal{S}_\ell(n,k)| \leq \binom{\ell}{n} \cdot \theta(n,k) \quad \textit{where,}$$

$$\theta(n,k) := \begin{cases} 1 & \textit{if } k = 1 \textit{ or } k = n-1 \\ (2k+3)^n & \textit{if } 2 \leq k < n/2 \\ (2n-2k+5)^n & \textit{if } n/2 \leq k \leq n-2\,. \end{cases}$$

Proof If we fix the $\ell - n$ fixed points for a permutation in $\mathcal{S}_\ell(n,k)$, then we get a derangement on the remaining n points with k exceedances. Therefore $|\mathcal{S}_\ell(n,k)| = \binom{\ell}{n} \cdot T(n,k)$. So, it suffices to prove that

- $T(n,1) = T(n,n-1) = 1$,
- $T(n,k) = T(n,n+1-k) < (2k+3)^n$ for any $k \in \{2,\ldots,n-2\}$.

From definition 8, $E(n,m) = \sum_{k=0}^{m+1} (-1)^k \binom{n+1}{k} \cdot (m+1-k)^n$. So removing terms with

odd k from $E(n,m)$ does not decrease it value. Therefore,

$$\begin{aligned} E(n,m) &< \binom{n+1}{0} \cdot (m+1)^n + \binom{n+1}{2} \cdot (m+1-2)^n + \dots \\ &= \sum_{i=0}^{m+1} \binom{n+1}{2i} \cdot (m+1-2i)^n \\ &< (m+1)^n \cdot \left(\sum_{i=0}^{n} \binom{n+1}{2i} \right) \leq (m+1)^n \cdot 2^n. \end{aligned}$$

Using equation 3.9 and taking absolute values, we get

$$T(n,k) < \sum_{j=0}^{n} \binom{n}{j} \cdot E(j,k) < \sum_{j=0}^{n} \binom{n}{j} \cdot 2^j \cdot (k+1)^j = (1+2(k+1))^n.$$

□

Proof of Lemma 14 For $\ell = 2$, $\mathrm{perm}(B_2) = 1 + b_{1,2} \cdot b_{2,1} \leq 1 + \frac{4f(1)^2}{f(2)} < 1.006$.

So we assume $\ell \geq 3$. We can expand the permanent in terms of the subsets $\mathcal{S}_\ell(n,k)$:

$$\mathrm{perm}(B_\ell) = 1 + \sum_{n=2}^{\ell} \sum_{k=1}^{n-1} \sum_{\sigma \in \mathcal{S}_\ell(n,k)} \prod_{i=1}^{\ell} b_{i,\sigma(i)}\,. \tag{3.10}$$

We will now bound the sum of $\prod_{i=1}^{\ell} b_{i,\sigma(i)}$ over all permutations σ in $\mathcal{S}_\ell(n,k)$ based on the relative values of n and k.

For fixed k and n with $k < n/2$, summing over all permutations in $\mathcal{S}_\ell(n,k)$ gives

$$\sum_{\sigma \in \mathcal{S}_\ell(n,k)} \prod_{i=1}^{\ell} b_{i,\sigma(i)} \leq |\mathcal{S}_\ell(n,k)| \cdot \max_{\sigma \in \mathcal{S}_\ell(n,k)} \prod_{i=1}^{\ell} b_{i,\sigma(i)} = \binom{\ell}{n} \cdot \theta(n,k) \cdot \max_{\sigma \in \mathcal{S}_\ell(n,k)} \prod_{i=1}^{\ell} b_{i,\sigma(i)} \tag{3.11}$$

$$\leq \binom{\ell}{n} \cdot \theta(n,k) \cdot 2^{2n-k} \cdot \frac{1}{\ell^{11(k-1)} \cdot (\ell \cdot (\ell-1) \dots (\ell-n+2k))^{11}}\,, \tag{3.12}$$

where the last inequality follows from Claim 3.4.2 and we use Lemma 15 to count de-

rangements. We will bound the three terms of equation (3.12), $\binom{\ell}{n}$, $\theta(n,k)$, and 2^{2n-k}, separately.

Expanding the first term, we get

$$\binom{\ell}{n} \cdot \frac{1}{\ell^{(k-1)} \cdot (\ell \cdot (\ell-1) \dots (\ell-n+2k))} = \frac{\ell \cdot (\ell-1) \dots (\ell-n+1)}{n! \cdot \ell^{(k-1)} \cdot (\ell \cdot (\ell-1) \dots (\ell-n+2k))} < \frac{1}{n! \cdot \ell^{k-1}} . \tag{3.13}$$

For the third term, note that $k < n/2$ implies that $2^{2n-k} < 2^{3n-3k}$, and since $k \geq 1$, $\ell - n + 2k > 2$. Using these two facts, we get

$$2^{2n-k} \cdot \frac{1}{\ell^{3(k-1)} \cdot (\ell \cdot (\ell-1) \dots (\ell-n+2k))^3} < \frac{2^{3(n-k)}}{(\ell-n+2k)^{3(n-k)}} < 1 . \tag{3.14}$$

Plugging (3.13) and (3.14) in (3.12), we get

$$\sum_{\sigma \in \mathcal{S}_\ell(n,k)} \prod_{i=1}^{\ell} b_{i,\sigma(i)} \leq \frac{1}{n! \cdot \ell^{k-1}} \cdot \theta(n,k) \cdot \frac{1}{\ell^{7(k-1)} \cdot (\ell \cdot (\ell-1) \dots (\ell-n+2k))^7} . \tag{3.15}$$

For $k = 1$, $\theta(n,k) = 1$ by Lemma 15, and therefore

$$\sum_{\sigma \in \mathcal{S}_\ell(n,1)} \prod_{i=1}^{\ell} b_{i,\sigma(i)} \leq \frac{1}{n!} \cdot \frac{1}{(\ell \cdot (\ell-1) \dots (\ell-n+2))^7} . \tag{3.16}$$

Summing over all n gives

$$\sum_{n=2}^{\ell} \sum_{\sigma \in \mathcal{S}_\ell(n,1)} \prod_{i=1}^{\ell} b_{i,\sigma(i)} \leq \sum_{n=2}^{\ell} \frac{1}{n!} \cdot \frac{1}{(\ell \cdot (\ell-1) \dots (\ell-n+2))^7} \leq \frac{1}{2\ell^6} . \tag{3.17}$$

For any $2 \leq k < n/2$, using Lemma 15, we have $\theta(n,k) = (2k+3)^n$. Since $k \geq 2$,

$2k+3 \le 2\cdot 2k$, and therefore $\theta(n,k) \le (2\cdot 2k)^{n+2}$. Plugging this is (3.15), we get

$$\sum_{\sigma\in\mathcal{S}_\ell(n,k)} \prod_{i=1}^{\ell} b_{i,\sigma(i)} \le \frac{1}{n!\cdot \ell^{k-1}} \cdot (2k)^n \cdot 2^n \cdot \frac{1}{\ell^{7(k-1)}\cdot(\ell\cdot(\ell-1)\dots(\ell-n+2k))^7}\,. \tag{3.18}$$

Moreover $k < n/2$ implies that $n-k > n/2$, and as a result $n \le 2(n-k)$ and $2^n\cdot(2k)^n \le 2^{2(n-k)}\cdot(2k)^{2(n-k)}$. Therefore,

$$\begin{aligned} 2^n\cdot(2k)^n \cdot &\frac{1}{\ell^{4(k-1)}\cdot(\ell\cdot(\ell-1)\dots(\ell-n+2k))^4} \\ &< 2^{2(n-k)}\cdot(2k)^{2(n-k)}\cdot\frac{1}{\ell^{4(k-1)}\cdot(\ell\cdot(\ell-1)\dots(\ell-n+2k))^4} \\ &< \left(\frac{2}{\ell-n+2k}\right)^{2(n-k)}\cdot\left(\frac{2k}{\ell-n+2k}\right)^{2(n-k)} < 1, \end{aligned} \tag{3.19}$$

where the last inequality follows from $2k \le \ell-n+2k$ and $2 \le \ell-n+2k$.

Combining (3.18) and (3.19), we have for any $2 \le k < n/2$,

$$\sum_{\sigma\in\mathcal{S}_\ell(n,k)} \prod_{i=1}^{\ell} b_{i,\sigma(i)} \le \frac{1}{n!\cdot\ell^{4(k-1)}\cdot(\ell\cdot(\ell-1)\dots(\ell-n+2k))^3}\,. \tag{3.20}$$

Summing over all k and n with $2 \le k < n/2$ gives

$$\begin{aligned} \sum_{n=2}^{\ell}\sum_{k=2}^{\lceil n/2\rceil-1}\sum_{\sigma\in\mathcal{S}_\ell(n,k)}\prod_{i=1}^{\ell} b_{i,\sigma(i)} &= \sum_{n=5}^{\ell}\sum_{k=2}^{\lceil n/2\rceil-1}\sum_{\sigma\in\mathcal{S}_\ell(n,k)}\prod_{i=1}^{\ell} b_{i,\sigma(i)} && \text{(since } n>2k>4\text{)} \\ &\le \sum_{n=5}^{\ell}\sum_{k=2}^{\lceil n/2\rceil-1}\frac{1}{n!\cdot\ell^{4(k-1)}\cdot(\ell\cdot(\ell-1)\dots(\ell-n+2k))^3} \\ &&& \text{(from (3.20))} \\ &\le \frac{1}{5!}\sum_{n=5}^{\ell}\sum_{k=2}^{\lceil n/2\rceil-1}\frac{1}{\ell^{4(k-1)}} \le \frac{1}{5!}\sum_{n=5}^{\ell}\sum_{k=2}^{\lceil n/2\rceil-1}\frac{1}{\ell^4} && \text{(since } k>2\text{)} \\ &\le \frac{1}{5!}\cdot\ell\cdot\lfloor\ell/2\rfloor\cdot\frac{1}{\ell^4} \le \frac{1}{2\cdot 5!\cdot\ell^2}\,. \end{aligned} \tag{3.21}$$

Similarly, for a fixed k and n with $k \geq n/2$, summing over all permutations in $\mathcal{S}_\ell(n,k)$,

$$\sum_{\sigma \in \mathcal{S}_\ell(n,k)} \prod_{i=1}^{\ell} b_{i,\sigma(i)} \leq \binom{\ell}{n} \cdot \theta(n,k) \cdot 2^{2n-k-1} \cdot \frac{((2k-n+1)!)^{11}}{\ell^{11k}}. \tag{3.22}$$

We again bound the three terms, namely $\binom{\ell}{n}$, $\theta(n,k)$, and 2^{2n-k-1} separately.

For the first term, since $n/2 \leq k$, $2n-k-1 \leq 3k-1$, and therefore

$$2^{2n-k-1} \cdot \frac{((2k-n+1)!)^3}{\ell^{3k}} \leq 2^{3k-1} \cdot \frac{((2k-n+1)!)^3}{\ell^{3k}}.$$

Since $2k-n+1 \leq k$, and $k+1 \leq \ell$, $2^{3k-1} \cdot \frac{((2k-n+1)!)^3}{\ell^{3k}} \leq \frac{1}{2} \cdot \frac{(2^k\, k!)^3}{(k+1)^{3k}}$.

For $k = 1,2,3,4,5$, $\frac{(2^k k!)^3}{(k+1)^{3k}} \leq 1$. For $k \geq 6$, $2^k k! \leq k^k$. Therefore,

$$2^{2n-k-1} \cdot \frac{((2k-n+1)!)^3}{\ell^{3k}} \leq \frac{1}{2}. \tag{3.23}$$

Expanding the third term,

$$\begin{aligned}
\binom{\ell}{n} \cdot \frac{((2k-n+1)!)^2}{\ell^{2k}} &= \frac{\ell \cdot (\ell-1) \ldots (\ell-n+1)}{n!} \cdot \frac{((2k-n+1)!)^2}{\ell^{2k}} \\
&\leq \frac{1}{n!} \cdot \frac{((2k-n+1)!)^2}{\ell^{2k-n}} \\
&= \frac{1}{\ell(n-1)!} \cdot \frac{((2k-n+1)!)^2}{n\ell^{2k-n-1}} \leq \frac{(2k-n+1)!}{\ell(n-1)!}.
\end{aligned}$$

Since $k+1 \leq n$, we have $2k-n+1 \leq n-1$, and therefore

$$\binom{\ell}{n} \cdot \frac{((2k-n+1)!)^2}{\ell^{2k}} \leq \frac{(2k-n+1)!}{l(n-1)!} \leq \frac{1}{\ell}. \tag{3.24}$$

Plugging in (3.23) and (3.24) in (3.22),

$$\sum_{\sigma \in \mathcal{S}_\ell(n,k)} \prod_{i=1}^{\ell} b_{i,\sigma(i)} \leq \frac{1}{2\ell} \cdot \theta(n,k) \cdot \frac{((2k-n+1)!)^6}{\ell^{6k}}. \tag{3.25}$$

Since $\theta(n, n-1) = 1$, for $k = n-1$, we have

$$\sum_{\sigma \in \mathcal{S}_\ell(n,n-1)} \prod_{i=1}^{\ell} b_{i,\sigma(i)} \leq \frac{1}{2\ell} \cdot \frac{((n-1)!)^6}{\ell^{6(n-1)}}. \tag{3.26}$$

Summing over all n gives

$$\begin{aligned} \sum_{n=2}^{\ell} \sum_{\sigma \in \mathcal{S}_\ell(n,n-1)} \prod_{i=1}^{\ell} b_{i,\sigma(i)} &\leq \sum_{n=2}^{\ell} \frac{1}{2\ell} \cdot \frac{((n-1)!)^6}{\ell^{6(n-1)}} \leq \sum_{n=2}^{\ell} \frac{1}{2\ell} \cdot \frac{1}{\ell^6} \qquad \text{(since } n-1 < \ell\text{)} \\ &\leq \frac{1}{2\ell^6} \end{aligned} \tag{3.27}$$

By Lemma 15, for $k < n-1$, $\theta(n,k) \leq (2n-2k+5)^n$ for $k \geq n/2$, and

$$\sum_{\sigma \in \mathcal{S}_\ell(n,k)} \prod_{i=1}^{\ell} b_{i,\sigma(i)} \leq \frac{1}{2\ell} \cdot (2n-2k+5)^n \cdot \frac{((2k-n+1)!)^6}{\ell^{6k}}.$$

Let $z = 2k - n$, then

$$\begin{aligned} (2n-2k+5)^n \cdot \frac{((2k-n+1)!)^6}{\ell^{6k}} &= (2k-2z+5)^{2k-z} \cdot \frac{((z+1)!)^6}{\ell^{6k}} \\ &\leq (2k-z+5)^{2k-z} \cdot \frac{((z+1)!)^6}{\ell^{6k}}. \end{aligned}$$

Taking the partial derivative of $\frac{(2k-z+5)^{2k-z}}{\ell^{6k}}$ with respect to k,

$$\frac{\partial \frac{(2k-z+5)^{2k-z}}{\ell^{6k}}}{\partial k} = \frac{(2k-z+5)^{2k-z}}{\ell^{6k}} \left(2 \cdot \log(2k-z+5) + 2 \cdot \frac{2k-z}{2k-z+5} - 6\log(\ell) \right),$$

where the last inequality follows since $2k - z \leq \ell$ and $\ell \geq 3$.

Therefore $\frac{(2k-z+5)^{2k-z+2}}{\ell^{6k}}$ is a non-increasing function of k. Since $n = 2k - z \geq 1$, k

satisfies $2k \geq z+1$. So $\frac{(2k-z+5)^{2k-z}}{\ell^{6k}}$ is maximized when $2k = z+1$. Therefore,

$$\begin{aligned}(2k-z+5)^{2k-z} \cdot \frac{((z+1)!)^6}{\ell^{6k}} &\leq 6 \cdot \frac{((z+1)!)^6}{\ell^{6k}} = 6 \cdot \frac{((2k-n+1)!)^6}{\ell^{6k}} \\ &\leq 6 \cdot \frac{\ell^{6(2k-n+1)}}{\ell^{6k}} \leq \frac{6}{\ell^{6(n-k-1)}}\,.\end{aligned}$$

Plugging this bound in (3.25) gives

$$\sum_{\sigma \in \mathcal{S}_\ell(n,k)} \prod_{i=1}^{\ell} b_{i,\sigma(i)} \leq \frac{1}{2\ell} \cdot \frac{6}{\ell^{6(n-k-1)}}\,. \tag{3.28}$$

Summing over all n and k gives

$$\begin{aligned}\sum_{n=2}^{\ell} \sum_{k=\lceil n/2 \rceil}^{n-2} \sum_{\sigma \in \mathcal{S}_\ell(n,k)} \prod_{i=1}^{\ell} b_{i,\sigma(i)} &\leq \sum_{n=2}^{\ell} \sum_{k=\lceil n/2 \rceil}^{n-2} \frac{1}{2\ell} \cdot \frac{6}{\ell^{6(n-k-1)}} \leq \sum_{n=2}^{\ell} \sum_{k=\lceil n/2 \rceil}^{n-2} \frac{1}{\ell} \cdot \frac{3}{\ell^6} \\ &\qquad \text{(since } n-k-1 \geq 1\text{)} \\ &\leq \ell \cdot \ell \cdot \frac{3}{\ell^7} \leq \frac{3}{\ell^5}\,. \end{aligned} \tag{3.29}$$

Plugging in (3.17), (3.21), (3.27), and (3.29) into (3.10),

$$\begin{aligned}\operatorname{perm}(B_\ell) &= 1 + \sum_{n=2}^{\ell} \sum_{k=1}^{n-1} \sum_{\sigma \in \mathcal{S}_\ell(n,k)} \prod_{i=1}^{\ell} b_{i,\sigma(i)} \\ &= 1 + \sum_{n=2}^{\ell} \sum_{\sigma \in \mathcal{S}_\ell(n,1)} \prod_{i=1}^{\ell} b_{i,\sigma(i)} + \sum_{n=5}^{\ell} \sum_{k=2}^{\lceil n/2 \rceil - 1} \sum_{\sigma \in \mathcal{S}_\ell(n,k)} \prod_{i=1}^{\ell} b_{i,\sigma(i)} \\ &\quad + \sum_{n=2}^{\ell} \sum_{\sigma \in \mathcal{S}_\ell(n,n-1)} \prod_{i=1}^{\ell} b_{i,\sigma(i)} + \sum_{n=2}^{\ell} \sum_{k=\lceil n/2 \rceil}^{n-2} \sum_{\sigma \in \mathcal{S}_\ell(n,k)} \prod_{i=1}^{\ell} b_{i,\sigma(i)} \\ &\leq 1 + \frac{1}{2\ell^5} + \frac{1}{240\ell^2} + \frac{1}{2\ell^6} + \frac{3}{\ell^5} \leq 1 + \frac{0.05}{\ell}\,.\end{aligned}$$

□

3.5 Conclusion and Open Directions

In the case of cardinality constraints, the original $O(d^d)$-approximation of [59] was improved to an $O((\log d)^d)$-approximation in [63] by first finding a good scaling of the input vectors, and then using the same algorithm. We do not expect a similar approach can lead to an improvement for this matroid-intersection-based algorithm. Even when all the input vectors are unit vectors, there exists an instance where the determinant is $O(d^{-d})$ and every single element swap leads to a decrease in volume, despite the existence of a solution with determinant one. Thus, it remains open to obtain and $O(e^d)$-approximation for the determinant maximization problem with matroid (or even just partition) constraints.

Another interesting direction is to consider wether similar spectral objectives are also amendable to matroid-intersection algorithms. In [2] they provide a detailed analysis of local search for various experimental design problems. Since the matroid-intersection approach relies on first understanding how the spectral objective behaves under individual swaps, it is possible that these ideas or techniques could be extended to the general matroid constraint setting.

CHAPTER 4

ASYMMETRIC NASH SOCIAL WELFARE

This chapter is based on joint work with Aditi Laddha, Madhusudhan Pittu, and Mohit Singh. The original version [68] appeared in the proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2024).

Fair and efficient division of resources among agents is a fundamental problem arising in various fields [7, 8, 9, 10, 11, 12]. 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 $\mathbf{v}_i : 2^{\mathcal{G}} \to \mathbf{R}_{\geq 0}$ 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 [29]. 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 with divisible goods. Informally, if a game satisfies the following axioms: (1) each individual wishes to maximize the utility to themselves, (2) if we restrict the set of allowable divisions without removing the optimal division, then the optimal division does not change, and (3) the individuals have equal bargaining skill, then the optimal assignment will maximize the NSW objective. This idea was later generalized to multiple players [26, 27]. 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 [28] 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 [29] 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 [29, 30].

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:

$$\prod_{i \in \mathcal{A}} \left(\mathbf{v}_i(\sigma^{-1}(i)) \right)^{w_i}.$$

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

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

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 [31, 32] in the context of two-person bargaining games, and can also be obtained through an axiomatic approach. Starting from Nash's three original axioms, if we throw out the third (which states that

players have equal bargaining skill) then [32] showed that there must be weights such that the optimal bargain maximizes the weighted NSW objective. Due to the added flexibility of assigning agents different weights, problems in many diverse domains can be modeled using the weighted objective, including bargaining theory [28, 33], water resource allocation [34, 35], and climate agreements [36]. From a context of indivisible goods, the study of this problem has been much more recent [37, 38, 39]. There have also been attempts to extend the ideas of envy-freeness to the weighted setting [40, 41], 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 chapter, we aim to shed light on the weighted Nash Social Welfare problem, mainly focusing on mathematical programming relaxations for the problem, and approximation algorithms that can be obtained by rounding solutions to the relaxations.

4.1 Related Work

The problem of finding the allocation that maximizes the Nash Social Welfare objective is an NP-hard problem, as was proven by [69]. Additionally, [70] 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 [71] using analogies from market equilibrium. [42] provided an improved analysis of the algorithm from [71] and introduced a convex programming relaxation. Using an entirely different approach, [43] 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 [30], 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, [38] 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. [72] provided a constant factor approximation algorithm for the unweighted case with submodular valuations. More recently, [39] 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 4 is always upper bounded by $\log(nw_{\max})$, which is exactly the guarantee of previous work [39]. 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 [73] and more general submodular valuations [74]. 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.

4.2 Formulations for the Nash Social Welfare Problem

In this section we will go through the history of convex relaxations for the Nash social welfare problem (without weights) . We start with the simplest (and weakest formulation), and then show how it was strengthened in two distinct ways by [60] and [42]. As part of the work on the weighted Nash social welfare problem in this chapter we will generalize both approaches to the weighted setting, so it is important to have a firm understanding these important formulations.

In the Nash social welfare problem we are asked to find an assignment which maximizes the product (or equivalently geometric mean) of the agents' valuations:

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

This objective originally arose through Nash's study of two player bargaining games with divisible goods. Informally, if a game satisfies the following axioms:

1. each individual wishes to maximize the utility to themselves,
2. if we restrict the set of allowable divisions without removing the optimal division, then the optimal division does not change, and
3. the individuals have equal bargaining skill,

then the optimal assignment will maximize the Nash social welfare objective.

We will more often use the logarithm of the Nash social welfare objective, and so define

$$\mathrm{NSW}(\sigma) := \sum_{i \in A} \log \left(\sum_{j \in \sigma^{-1}(i)} v_{ij} \right).$$

With divisible goods, it is easy to model this problem as a convex program. For each $i \in \mathcal{A}$ and $j \in \mathcal{G}$ we create a variable x_{ij} which tracks the portion of item j which is assigned to

agent i. This leads to the Eisenberg-Gale program [75] for Nash social welfare:

$$\begin{aligned} \max \quad & \sum_{i \in \mathcal{A}} \log \left(\sum_{j \in \mathcal{G}} v_{ij} \cdot x_{ij} \right) & \\ \text{s.t} \quad & \sum_{i \in \mathcal{A}} x_{ij} = 1 & \forall j \in \mathcal{G} \\ & x_{ij} \geq 0 & \forall i \in \mathcal{A}, j \in \mathcal{G}. \end{aligned}$$

This is an exact formulation for the case of divisible goods, and the objective is convex so it can be solved efficiently.

As an alternative, there is the Shmyrev relaxation [76], which we show to be equivalent to the Eisenberg-Gale program in Lemma 16.

$$\begin{aligned} \max_{b,q} \quad & \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{G}} b_{ij} \log \left(v_{ij} \right) - \sum_{j \in \mathcal{G}} \left(\sum_{i \in \mathcal{A}} b_{ij} \right) \log \left(\sum_{i \in \mathcal{G}} b_{ij} \right) \\ \text{s.t.} \quad & \sum_{j} b_{ij} = 1 \quad \forall i \in \mathcal{A} \\ & b_{ij} \geq 0 \quad \forall (i,j) \in \mathcal{A} \times \mathcal{G}. \end{aligned}$$

We can show that these two programs are equivalent giving a way to convert an optimal solution to one into a solution to the other with the same objective. The proof crucially uses the fact that we are starting from an optimal solution, so the same transformation does not necessarily preserve the objective of other solutions.

It is also possible to show that the programs are equivalent by comparing their dual programs. In [42] they showed that their duals are equivalent via a change of variables.

Lemma 16 *The optimal values of the Eisenberg-Gale and Shmyrev programs are equal.*

Proof First, suppose we have an optimal solution x to the Eisenberg-Gale program. We will set

$$b_{ij} = \frac{v_{ij} x_{ij}}{\sum_{j' \in \mathcal{G}} v_{ij'} x_{ij'}}.$$

The sum of the numerator over all $j \in \mathcal{G}$ is exactly the denominator, so we can see that the solution b we constructed is feasible for the Shmyrev program. Before trying to compute the objective value, we need to understand the optimality conditions for the solution x.

We can start by writing down the Lagrangian relaxation, with variables λ_j corresponding to the constraints $\sum_{i \in \mathcal{A}} x_{ij} = 1$.

$$L(x; \lambda) = \sum_{i \in \mathcal{A}} \log \left(\sum_{j \in \mathcal{G}} v_{ij} x_{ij} \right) + \sum_{j \in \mathcal{G}} \lambda_j \left(1 - \sum_{i \in \mathcal{A}} x_{ij} \right).$$

We get the first-order optimality conditions by taking the derivative with respect to x;

$$\frac{\partial}{\partial x_{kl}} L = \frac{v_{kl}}{\sum_{j \in \mathcal{G}} v_{kj} x_{kj}} - \lambda_l,$$

so since x is optimal we know $\frac{v_{kl}}{\sum_{j \in \mathcal{G}} v_{kj} x_{kj}} = \lambda_l$ for all $k \in \mathcal{A}$ and $l \in \mathcal{G}$. In particular, we can write $b_{ij} = x_{ij} \cdot \lambda_j$. Now, plugging this into the objective we see that

$$\begin{aligned}
\sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{G}} b_{ij} \log (v_{ij}) - \sum_{j \in \mathcal{G}} \left(\sum_{i \in \mathcal{A}} b_{ij} \right) \log \left(\sum_{i \in \mathcal{G}} b_{ij} \right) & \\
&= \sum_{j \in \mathcal{G}} \lambda_j \sum_{i \in \mathcal{A}} x_{ij} \log(v_{ij}) - \sum_{j \in \mathcal{G}} \lambda_j \left(\sum_{i \in A} x_{ij} \right) \log \left(\lambda_j \sum_{i \in A} x_{ij} \right) \\
&= \sum_{j \in \mathcal{G}} \lambda_j \sum_{i \in \mathcal{A}} x_{ij} \cdot \log \left(\frac{v_{ij}}{\lambda_j} \right) \\
&= \sum_{j \in \mathcal{G}} \lambda_j \sum_{i \in \mathcal{A}} x_{ij} \cdot \log \left(\sum_{j' \in \mathcal{G}} v_{ij'} x_{ij'} \right) \\
&= \sum_{i \in A} \log \left(\sum_{j \in \mathcal{G}} v_{ij} x_{ij} \right) \cdot \left(\sum_{j \in \mathcal{G}} \lambda_j x_{ij} \right) \\
&= \sum_{i \in A} \log \left(\sum_{j \in \mathcal{G}} v_{ij} x_{ij} \right),
\end{aligned}$$

where we have used the fact that $\sum_{i \in A} x_{ij} = 1$, and the optimality conditions.

For the reverse direction suppose that we have an optimal solution b_{ij} to the Shmyrev

program. We will set

$$x_{ij} = \frac{b_{ij}}{\sum_{i'} b_{i'j}}.$$

Once again we have chosen the denominator so that feasibility is obvious. And, once again, we first need to write down the optimality conditions for the solution b.

In this case, the Lagrangian relaxation has variables γ_i corresponding to the constraint $\sum_{j \in \mathcal{G}} b_{ij} = 1$ for all $i \in \mathcal{A}$.

$$L(b; \gamma) = \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{G}} b_{ij} \log(v_{ij}) - \sum_{j \in \mathcal{G}} \left(\sum_{i \in \mathcal{A}} b_{ij} \right) \log \left(\sum_{i \in \mathcal{G}} b_{ij} \right) + \sum_{i \in \mathcal{A}} \gamma_i \left(1 - \sum_{j \in \mathcal{G}} b_{ij} \right).$$

We get the first-order optimality conditions by taking the derivative with respect to b;

$$\frac{\partial}{\partial b_{kl}} L = \log(v_{kl}) - 1 - \log \left(\sum_{i \in \mathcal{A}} b_{il} \right) - \gamma_k,$$

so since b is optimal we know that $\frac{v_{kl}}{\sum_{i \in \mathcal{A}} b_{il}}$ is independent of l for all $k \in \mathcal{A}$. Now, we can plug this into the objective

$$\begin{aligned}
\sum_{i \in A} \log \left(\sum_{j \in \mathcal{G}} v_{ij} x_{ij} \right) &= \sum_{i \in \mathcal{A}} \log \left(\sum_{j \in \mathcal{G}} \frac{v_{ij} b_{ij}}{\sum_{i' \in \mathcal{A}} b_{i'j}} \right) \\
&= \sum_{i \in \mathcal{A}} \log \left(\sum_{j \in \mathcal{G}} e^{\gamma_i + 1} \cdot b_{ij} \right) \\
&= \sum_{i \in \mathcal{A}} (\gamma_i + 1) \\
&= \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{G}} b_{ij} (\gamma_i + 1) \\
&= \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{G}} b_{ij} \log \left(\frac{v_{ij}}{\sum_{i' \in \mathcal{A}} b_{i'j}} \right),
\end{aligned}$$

and after pulling out the denominator from the logarithm we get exactly the objective of b.

□

While both of the above programs are exact for the fractional allocation problem, we are more interested in the case of indivisible goods. Both programs are still relaxations for the integral problem [42], which is straightforward for the Eisenberg-Gale relaxation, but requires we properly interpret the variables of the Shmyrev relaxation.

Lemma 17 *Both the Eisenberg-Gale program and the Shmyrev program are relaxations for the indivisible Nash social welfare problem.*

Proof Given an assignment $\sigma : \mathcal{G} \to \mathcal{A}$ we want to show that there are corresponding solutions to the Eisenberg-Gale and Shmyrev programs which have the same objective value.

For the Eisenberg-Gale program we can simply set

$$x_{ij} = \begin{cases} 1 & \text{if } \sigma(j) = i, \\ 0 & \text{otherwise.} \end{cases}$$

This will satisfy the constraint, since for each $j \in G$ there is exactly one i such that $x_{ij} = 1$ and the rest are zero. Then we can calculate that

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

For the Shmyrev program we must set

$$b_{ij} = \begin{cases} \frac{v_{ij}}{\sum_{k \in \sigma^{-1}(i)} v_{ik}} & \text{if } \sigma(j) = i \\ 0 & \text{otherwise.} \end{cases}$$

The way we interpret the variables of the Shmyrev program is that each agent has a budget of 1, which they must divide among the items. The definition above states that the proportion of agent i's budget which they spend on item j should be 0 if they were not assigned, and otherwise be equal to the proportion of i's utility which came from item j. By summing

the b_{ij} variables over items $j \in G$ we only retain the terms which were assigned to agent i, which cancels with the denominator to show we satisfy the constraint. Now the objective value is

$$\begin{aligned}
&\sum_{i\in A}\sum_{j\in G} b_{ij}\log(v_{ij}) - \sum_{j\in\mathcal{G}}\left(\sum_{i\in\mathcal{A}} b_{ij}\right)\log\left(\sum_{i\in\mathcal{A}} b_{ij}\right) \\
&\qquad = \sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{G}} b_{ij}\log(v_{ij}) - \sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{G}} b_{ij}\log(b_{\sigma(j),j}) \\
&\qquad = \sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{G}} b_{ij}\log(v_{ij}) - \sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{G}} b_{ij}\log\left(\frac{v_{ij}}{\sum_{k\in\sigma^{-1}(i)} v_{ik}}\right) \\
&\qquad = \sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{G}} b_{ij}\log\left(\sum_{k\in\sigma^{-1}(i)} v_{ik}\right) \\
&\qquad = \sum_{i\in\mathcal{A}}\log\left(\sum_{k\in\sigma^{-1}(i)} v_{ik}\right) = \mathrm{NSW}(\sigma).
\end{aligned}$$

□

Unfortunately, both programs are very bad relaxations for the integral version of the problem; they cannot even distinguish between zero and non-zero objective when the the number of items is equal to the number of agents [71].

Consider an instance with n items and n agents, where there is exactly one valuable item. Let us say that $v_{i1} = 1$ for all $i \in \mathcal{A}$, and $v_{ij} = 0$ for all $j \neq 1$. Then every assignment must have only one agent with non-zero utility, so for any σ we have $\mathrm{NSW}(\sigma) = -\infty$. On the other hand, in the Eisenberg-Gale program we can set $x_{ij} = 1/n$ for all $i \in \mathcal{A}$ and $j \in \mathcal{G}$. The objective value of this solution is

$$\sum_{i\in\mathcal{A}} \log(1/n) = -n\log(n).$$

Since the Shmyrev program is equivalent, it also has unbounded gap.

There have been a few approaches to repair this problem by adding additional con-

straints to create stronger relaxations. The first, due to [42], adds an additional constraint to the Shmyrev program.

$$\begin{aligned}
\max_{b} \quad & \sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{G}} b_{ij}\log\left(v_{ij}\right) - \sum_{j\in\mathcal{G}}\left(\sum_{i\in\mathcal{A}} b_{ij}\right)\log\left(\sum_{i\in\mathcal{A}} b_{ij}\right) && \text{(SpendingRestricted)}\\
\text{s.t.} \quad & \sum_{j} b_{ij} = 1 \quad \forall i \in \mathcal{A}\\
& \sum_{i} b_{ij} \leq 1 \quad \forall j \in \mathcal{G}\\
& b_{ij} \geq 0 \quad \forall i \in \mathcal{A}, j \in \mathcal{G}.
\end{aligned}$$

As we mentioned earlier, the b_{ij} variables can be interpreted as the proportion of agent i's budget which they spend on item j. The new constraint says that the total spending on each item is bounded by 1. This prevents the case that there is one very valuable item on which all the agents would like to spend all their budget. It is still a relaxation, since if we set

$$b_{ij} = \begin{cases} \frac{v_{ij}}{\sum_{k\in\sigma^{-1}(i)} v_{ik}} & \text{if } \sigma(j) = i \\ 0 & \text{otherwise,} \end{cases}$$

we already have the inequality

$$\sum_{i\in\mathcal{A}} b_{ij} = \frac{v_{\sigma(j),j}}{\sum_{k:\sigma(i)=\sigma(j)} v_{\sigma(i),k}} \leq 1,$$

since the sum only picks up a single term.

An alternative approach was used in [60]. They added auxiliary variables to the Eisenberg -Gale program which act as a scaling on the valuations for each item separately. Intuitively this allows the program to automatically scale down the valuation of an item if it is valued extremely highly by all agents. This program is concave in x and convex in $\log y$.

$$\begin{aligned} \max_{\mathbf{x}\geq 0}\ \inf_{\mathbf{y}>0} \quad & \sum_{i\in\mathcal{A}} \log\left(\sum_{j\in\mathcal{G}} x_{ij}\ v_{ij}\ y_j\right) \\ \text{s.t.} \quad & \sum_{i\in\mathcal{A}} x_{ij} = 1 \quad \forall j \in \mathcal{G} \\ & \prod_{j\in S} y_j \geq 1 \quad \forall S \in \binom{\mathcal{G}}{n}. \end{aligned} \tag{LogConcave}$$

In this case it is less clear that the formulation is still a relaxation. We can still set

$$x_{ij} = \begin{cases} 1 & \text{if } \sigma(j) = i, \\ 0 & \text{otherwise.} \end{cases},$$

but now we also need to worry about the extra y variables to show that the objective is correct. We will phrase this as a separate lemma, since we will use the inequality again when we generalize these programs in Chapter 4.

Lemma 18 *Let $\sigma : \mathcal{G} \to \mathcal{A}$ be an assignment, and $y \in \mathbf{R}^{\mathcal{G}}$ 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

$$\sum_{i\in\mathcal{A}} \log\left(\sum_{j\in S_i} v_{ij}\ y_j\right) = \log\left(\prod_{i\in\mathcal{A}}\left(\sum_{j\in S_i} v_{ij}\ y_j\right)\right)$$

$$
\begin{aligned}
&= \log\left(\sum_{S\in\mathcal{S}}\left(\prod_{j\in S} y_j\right)\prod_{j\in S} v_{\sigma(j)j}\right)\\
&\geq \log\left(\sum_{S\in\mathcal{S}}\prod_{j\in S} v_{\sigma(j)j}\right)\\
&= \log\left(\prod_{i\in\mathcal{A}}\sum_{j\in S_i} v_{ij}\right) = \sum_{i\in\mathcal{A}}\log\left(\sum_{j\in S_i} v_{ij}\right).
\end{aligned}
$$

The only inequality uses the fact that $\prod_{j\in S} y_j \geq 1$ term-by-term in the sum. □

The program (SpendingRestricted) was used in [42] to develop a combinatorial rounding algorithm, which gives a 2-approximation for the Nash social welfare problem.

The algorithm starts by solving (SpendingRestricted) to get an optimal solution b, and computing auxiliary variables $q_j = \sum_{i\in\mathcal{A}} b_{ij}$. If we fix the variables q_j and allow the b variables to vary, then the objective becomes linear, and the constraints are form an assignment polytope, so can assume that the support graph of b, denoted $G(b)$, is acyclic. For each tree in the support graph they select a root agent arbitrarily, and then integrally assign all leaf items, or items with $q_j \leq 1/2$ to their parent agent. Finally, the remaining items are matched by computing the optimal matching given their current assignments and using the weights $\log(v_{ij})$ to convert the problem into a standard max-weight matching problem.

Algorithm 4 Spending-restricted rounding [71, 42]

Compute an optimal solution (b, q) to (SpendingRestricted) with acyclic support.

Choose a root agent for each tree in the support graph $G(b)$.

Assign any leaf items in the trees to their parent agents.

Assign any item with $q_j \leq 1/2$ to its parent agent.

Compute the optimal matching of remaining items to their adjacent agents.

Since this algorithm rounds a solution to (SpendingRestricted) it actually implies that the integrality gap is at most 2.

The alternative approach using (LogConcave) was also used to develop a simple round-

ing algorithm.

Algorithm 5 Log-concave rounding [60]

Compute an optimal solution x to (LogConcave).

Independently for each item $j \in \mathcal{G}$ assign item j to one agent, with probability x_{ij}.

They take advantage of the insight that the objective of (LogConcave) is a stable polynomial to analyze this simple rounding algorithm. The result is an e-approximation, which shows that the integrality gap of (LogConcave) is at most e.

Since (LogConcave) and (SpendingRestricted) have the same optimal value for all instances, the programs will also have the same integrality gap. Thus, the rounding algorithm for (SpendingRestricted) also implies that the integrality gap of (LogConcave) is at most 2. There is no known rounding algorithm for this program which attains this bound.

4.3 Notation

In order to state the key results of this chapter, 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 Gibb'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 9 (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.

We will often refer to the structural properties of given solutions in terms of their support.

Definition 10 (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 11 (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.*

4.4 Summary of Results

Our main contributions are (1) to show equivalence between different mathematical relaxations for the unweighted Nash Social Welfare problem, (2) generalize these formulations to give new mathematical formulations for the weighted NSW and finally (3) use the equivalence and generalize the algorithms for unweighted case to obtain improved approximation algorithms for weighted NSW.

4.4.1 Equivalence of Convex Programs

Our first result relates two previous convex programming relaxations for the unweighted Nash Social Welfare problem presented in [42] and [43].

Building on the algorithm of [71], [42] introduced the following relaxation for the unweighted Nash Social Welfare problem.

$$\begin{aligned}
\max_{b} \quad & \frac{1}{n}\sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{G}} b_{ij}\log\left(v_{ij}\right) - \frac{1}{n}\sum_{j\in\mathcal{G}}\left(\sum_{i\in\mathcal{A}} b_{ij}\right)\log\left(\sum_{i\in\mathcal{A}} b_{ij}\right) && \text{(CVX-Unweighted)}\\
\text{s.t.} \quad & \sum_{j} b_{ij} = 1 \quad \forall i \in \mathcal{A}\\
& \sum_{i} b_{ij} \leq 1 \quad \forall j \in \mathcal{G}\\
& b_{ij} \geq 0 \quad \forall (i,j) \in \mathcal{A}\times\mathcal{G}.
\end{aligned}$$

They showed that (CVX-Unweighted) is a convex relaxation, and the prices used by the algorithm presented in [71] 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:

$$b_{ij} = \begin{cases} \frac{v_{ij}}{\sum_{k\in\sigma^{-1}(i)} v_{ik}} & \text{if } \sigma(j) = i\\ 0 & \text{otherwise.}\end{cases} \tag{4.1}$$

In Section 4.2 we verified 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), was presented by Anari et al. [43]. They showed that the objective 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.

$$\max_{\mathbf{x}\geq 0}\ \inf_{\mathbf{y}>0} \quad \frac{1}{n}\sum_{i\in\mathcal{A}} \log\left(\sum_{j\in\mathcal{G}} x_{ij}\ v_{ij}\ y_j\right) \qquad \text{(LogConcave-Unweighted)}$$
$$\text{s.t.} \quad \sum_{i\in\mathcal{A}} x_{ij} = 1 \quad \forall j \in \mathcal{G}$$
$$\prod_{j\in S} y_j \geq 1 \quad \forall S \in \binom{\mathcal{G}}{n}.$$

Here, $\binom{\mathcal{G}}{n}$ denotes the collection of subsets of $\mathcal{G}$ of size n, where $n = |\mathcal{A}|$.

We show that both programs are relaxations for the Nash social welfare problem in Lemma 25.

On the surface, (LogConcave-Unweighted) and (CVX-Unweighted), and their corresponding rounding algorithms are quite different: [42] uses intuition from economics and market equilibrium to both arrive at (CVX-Unweighted) and also to round it, while [43] uses the properties of log-concave polynomials to round (CVX-Unweighted). However, our next result shows that these two convex programs indeed optimize the same objective.

Theorem 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.*

In Section 4.5 we show the programs are equivalent by transforming an optimal solution to one into an optimal solution to the other, and in Section 4.5.1, we 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. This later result is presented in more generality with non-uniform weights, since it is essential to our derivation of (CVX-Weighted).

4.4.2 New Relaxations in the Weighted Case

Besides providing a novel connection between two very different approaches to the unweighted problem, Theorem 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: [42, 71] 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 a stable polynomial, and therefore the machinery introduced in [43] 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, (NCVX-Weighted).

$$\begin{aligned}
\max_{\mathbf{b}} \quad & \sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{G}} w_i\, b_{ij} \log v_{ij} - \sum_{j\in\mathcal{G}}\sum_{i\in\mathcal{A}} w_i\, b_{ij} \log\left(\sum_{i'\in\mathcal{A}} b_{i'j}\right) && \text{(NCVX-Weighted)}\\
\text{s.t.} \quad & \sum_{j\in\mathcal{G}} b_{ij} = 1 \quad \forall i \in \mathcal{A}\\
& \sum_{i\in\mathcal{A}} b_{ij} \le 1 \quad \forall j \in \mathcal{G}\\
& b_{ij} \ge 0 \quad \forall (i,j) \in \mathcal{A}\times\mathcal{G}
\end{aligned}$$

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 2 we can start from (LogConcave-Weighted) and get an equivalent

convex program using the b variables.

$$\begin{aligned}
\max_{\mathbf{b}} \quad & \sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{G}} w_i\, b_{ij} \log v_{ij} - \sum_{j\in\mathcal{G}}\sum_{i\in\mathcal{A}} w_i\, b_{ij} \log\left(\sum_{i'\in\mathcal{A}} w_{i'}\, b_{i'j}\right) + \sum_{i\in\mathcal{A}} w_i \log w_i \\
\text{s.t.} \quad & \sum_{j\in\mathcal{G}} b_{ij} = 1 \quad \forall i \in \mathcal{A} \\
& \sum_{i\in\mathcal{A}} b_{ij} \le 1 \quad \forall j \in \mathcal{G} \\
& b_{ij} \ge 0 \quad \forall (i,j) \in \mathcal{A}\times\mathcal{G} \qquad \text{(CVX-Weighted)}
\end{aligned}$$

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). By setting $\mathbf{b}$ to be the same value as (4.1), it is natural to see that both programs are indeed relaxations.

Theorem 3 (NCVX-Weighted)*,* (LogConcave-Weighted)*, and* (CVX-Weighted) *are relaxations of the weighted Nash Social Welfare problem. Moreover, the convex programs* (LogConcave-Weighted)*, and* (CVX-Weighted) *are equivalent.*

We formally prove Theorem 3 in Section 4.6, and we show that (CVX-Weighted) is equivalent to (LogConcave-Weighted) in Section 4.5.1. 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

$$\begin{aligned}
f_{\text{ncvx}}(\mathbf{b}) &:= \sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{G}} w_i\, b_{ij} \log v_{ij} - \sum_{j\in\mathcal{G}}\sum_{i\in\mathcal{A}} w_i\, b_{ij} \log\left(\sum_{i'\in\mathcal{A}} b_{i'j}\right), \quad \text{and} \\
f_{\text{cvx}}(\mathbf{b}) &:= \sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{G}} w_i\, b_{ij} \log v_{ij} - \sum_{j\in\mathcal{G}}\sum_{i\in\mathcal{A}} w_i\, b_{ij} \log\left(\sum_{i'\in\mathcal{A}} w_{i'}\, b_{i'j}\right) + \sum_{i\in\mathcal{A}} w_i \log w_i.
\end{aligned}$$

With these definitions we can compactly write the two different relaxations.

We can also relate the objective value of the three programs.

$\max_{\mathbf{b}} \quad f_{\text{cvx}}(\mathbf{b})$ s.t. $\quad \mathbf{b} \in \mathcal{P}(\mathcal{A}, \mathcal{G})$	$\max_{\mathbf{b}} \quad f_{\text{ncvx}}(\mathbf{b})$ s.t. $\quad \mathbf{b} \in \mathcal{P}(\mathcal{A}, \mathcal{G})$
(CVX-Weighted)	(NCVX-Weighted)

Lemma 19 *For any any feasible* $\mathbf{b} \in \mathcal{P}(\mathcal{A}, \mathcal{G})$ *and positive 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.*

This result is tight; the integrality gap of (CVX-Weighted) can be as large as $D_{\text{KL}}(\mathbf{w} \,||\, \mathbf{u})$ 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 $f_{\text{cvx}}(b) = D_{\text{KL}}(\mathbf{w} \,||\, \mathbf{u})$, while the $f_{\text{ncvx}}(b)$ and the integral optimal value are zero since for any perfect matching the product of the agents' utilities is exactly 1, and any other assignment leaves at least one agent unmatched.

4.4.3 Rounding Algorithm.

Since (CVX-Weighted) is structurally similar to (CVX-Unweighted) we may hope to apply the rounding algorithm from [42], but unfortunately (CVX-Weighted) lacks a crucial property: optimal solutions of (CVX-Weighted) need not be acyclic. Furthermore, as we saw at the end of the previous section, the integrality gap of (CVX-Weighted) is unbounded. 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.

Lemma 20 *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 $|\mathcal{G}|$.

This lemma takes advantage of the local-optimality conditions for the non-convex relaxation, and its related stability properties. Next, we establish that one can efficiently round any feasible point whose support graph is a forest to an integral assignment.

Theorem 25 *For a Nash Social Welfare instance $(\mathcal{A}, \mathcal{G}, \mathbf{v}, \mathbf{w})$, given a vector $\mathbf{b}^{\text{forest}} \in \mathcal{P}(\mathcal{A}, \mathcal{G})$ such that the support of $\mathbf{b}^{\text{forest}}$ is a forest, there exists a deterministic polynomial time algorithm (Algorithm 7) which returns an assignment $\sigma : \mathcal{G} \to \mathcal{A}$ such that*

$$\text{NSW}(\sigma) \geq f_{\text{cvx}}(\mathbf{b}^{\text{forest}}) - D_{\text{KL}}(\mathbf{w} \,||\, \mathbf{u}) - 2\log 2 - \frac{1}{2e}.$$

By combining Lemma 20, and Theorem 25, we obtain an approximation algorithm with approximation ratio

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

for the weighted Nash Social Welfare problem with additive valuations. When all the weights are the same, this gives a constant factor approximation.

Theorem 4 *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 6) that, given $(\mathcal{A}, \mathcal{G}, \mathbf{v}, \mathbf{w})$, returns an assignment $\sigma : \mathcal{G} \to \mathcal{A}$ such that*

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

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

Algorithm 6 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 [77].

We remark that our algorithm for rounding (NCVX-Weighted) (Algorithm 7) is the same as that in [71]. 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 lose an extra factor of $D_{\mathrm{KL}}(\mathbf{w} \,||\, \mathbf{u})$ because we have no way to find a locally-optimal solution to (NCVX-Weighted) is polynomial time, and so need to return to the convex program for a second time. It is possible to find a locally optimal solution to (NCVX-Weighted) if we allow the running time to depend on the unary representation of the valuations. We discuss this approach, and use it to provide a pseudopolynomial-time algorithm with slightly improved approximation in Appendix A.3.

4.5 Relationships Between the Mathematical Programs

In this section we prove Theorem 2 to show that two natural convex programming formulations for the unweighted Nash Social Welfare problem, (LogConcave-Unweighted) and (CVX-Unweighted), are equivalent. To do this, we will present a direct substitution that takes an optimal solution to (LogConcave-Unweighted) and produces an optimal solution to (CVX-Unweighted) with the same objective. In Section 4.5.1 we present an alternative proof which shows that the programs are equivalent through a chain of duals and changes of variables.

Theorem 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.*

Proof We will start with an optimal solution to (LogConcave-Unweighted) and use it to construct a solution to (CVX-Unweighted). Using the optimality conditions, we will be able to show that the new solution is feasible, has the same objective value, and, moreover, satisfies the optimality conditions for (CVX-Unweighted).

Let x, y be an optimal solution to (LogConcave-Unweighted), and set

$$b_{ij} = \frac{x_{ij} v_{ij} y_j}{\sum_{j' \in \mathcal{G}} x_{ij'} v_{ij'} y_{j'}}.$$

By the choice of denominator it is clear that $\sum_{j \in G} b_{ij} = 1$ for all $i \in \mathcal{A}$. We will need the optimality conditions to show that $\sum_{i \in \mathcal{A}} b_{ij} \leq 1$ for all $j \in \mathcal{G}$.

The program is not actually convex in the variables y, so we write $y_j = e^{\gamma_j}$ before computing the optimality conditions. Note that the constraints $\prod_{j \in S} y_j \geq 1$ becomes $\sum_{j \in S} \gamma_j \geq 0$. First, we will only consider optimality with respect to the inner infimum, for fixed x. When x is fixed, we have non-negative variables λ_S for each subset S of size n, and the Lagrangian relaxation is

$$L(\gamma; \lambda) = \sum_{i \in \mathcal{A}} \log \left(\sum_{j \in \mathcal{G}} x_{ij} v_{ij} e^{\gamma_j} \right) - \sum_{S \in \binom{\mathcal{G}}{n}} \lambda_S \sum_{j \in S} \gamma_j.$$

When we take the derivative we see that

$$\frac{\partial}{\partial \gamma_j} L = \sum_{i \in \mathcal{A}} \frac{x_{ij} v_{ij} y_j}{\sum_{j' \in \mathcal{G}} x_{ij'} v_{ij'} y_{j'}} - \sum_{S : j \in S} \lambda_S,$$

so the optimality conditions imply that

$$\sum_{i \in \mathcal{A}} b_{ij} = \sum_{S : j \in S} \lambda_S.$$

Now

$$n \cdot \sum_{S \in \binom{\mathcal{G}}{n}} \lambda_S = \sum_S \sum_{j \in S} \lambda_S = \sum_{j \in \mathcal{G}} \sum_{S: j \in S} \lambda_S = \sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} b_{ij} = n,$$

which implies that $\sum_{S \in \binom{\mathcal{G}}{n}} \lambda_S = 1$. Thus, we can conclude that

$$\sum_{i \in \mathcal{A}} b_{ij} = \sum_{S: j \in S} \lambda_S \leq \sum_{S \in \binom{\mathcal{G}}{n}} \lambda_S = 1,$$

and that the solution b is indeed feasible.

Next, to show that the objective value is the same, we will need the optimality conditions for x. We will have a non-negative variable α_j for all $j \in \mathcal{G}$, and variables ξ_{ij} for all $i \in \mathcal{A}$, $j \in \mathcal{G}$. The Lagrangian relaxation is

$$L(x; \alpha, \xi) = \sum_{i \in \mathcal{A}} \log \left(\sum_{j \in \mathcal{G}} x_{ij} v_{ij} y_j \right) - \sum_{j \in \mathcal{G}} \alpha_j \left(1 - \sum_{i \in \mathcal{A}} x_{ij} \right) + \sum_{i,j} \xi_{ij} x_{ij}.$$

We can take the derivative with respect to x_{ij} to get

$$\frac{\partial}{\partial x_{ij}} L = \frac{v_{ij} y_j}{\sum_{j' \in \mathcal{G}} x_{ij'} v_{ij'} y_{j'}} - \alpha_j + \xi_{ij}.$$

Thus, combining this with the complementary slackness conditions for the non-negativity constraints for x_{ij} we see that if $x_{ij} > 0$ then

$$\frac{v_{ij} y_j}{\sum_{j' \in \mathcal{G}} x_{ij'} v_{ij'} y_{j'}} = \alpha_j.$$

Multiplying both sides by x_{ij}, we see that we always get the equality

$$b_{ij} = x_{ij} \cdot \alpha_j.$$

With this, we can substitute everything into the new objective.

$$\begin{aligned}
\sum_{i,j} b_{ij} \log(v_{ij}) - \sum_{i,j} b_{ij} \log\left(\sum_{i' \in \mathcal{A}} b_{i'j}\right) &= \sum_{i,j} b_{ij} \log(v_{ij}) - \sum_{i,j} b_{ij} \log\left(\alpha_j \cdot \sum_{i' \in \mathcal{A}} x_{i'j}\right) \\
&= \sum_{i,j} b_{ij} \log\left(\frac{v_{ij}}{\alpha_j}\right) \\
&= \sum_{i,j} b_{ij} \log\left(\frac{\sum_{j' \in \mathcal{G}} x_{ij'} v_{ij'} y_{j'}}{y_j}\right) \\
&= \sum_{i,j} b_{ij} \log\left(\sum_{j' \in \mathcal{G}} x_{ij'} v_{ij'} y_{j'}\right) - \sum_{i,j} b_{ij} \gamma_j \\
&= \sum_{i \in \mathcal{A}} \log\left(\sum_{j \in \mathcal{G}} x_{ij} v_{ij} y_j\right),
\end{aligned}$$

where the last line follows from the complementary slackness condition on the γ_j variables:

$$\sum_{i,j} b_{ij} \gamma_j = \sum_{j \in \mathcal{G}} \sum_{s : j \in S} \lambda_S \gamma_j = \sum_{S \in \binom{\mathcal{G}}{n}} \lambda_S \sum_{j \in S} \gamma_j = 0.$$

Finally, we must show that the new solution we have constructed is optimal, by showing that it satisfies the optimality conditions for (CVX-Unweighted). To determine the optimality conditions we will have dual variables μ_i for $i \in \mathcal{A}$ and $\nu_j \geq 0$ for $j \in \mathcal{G}$. Then the Lagrangian relaxation for (CVX-Unweighted) is

$$\begin{aligned}
L(b; \mu, \nu) = &\sum_{i,j} b_{ij} \log(v_{ij}) - \sum_{i,j} b_{ij} \log\left(\sum_{i' \in \mathcal{A}} b_{i'j}\right) \\
&+ \sum_{i \in \mathcal{A}} \mu_i \left(1 - \sum_{j \in \mathcal{G}} b_{ij}\right) + \sum_{j \in \mathcal{G}} \left(1 - \sum_{i \in \mathcal{A}} b_{ij}\right).
\end{aligned}$$

The partial derivative with respect to b_{kl} is

$$\frac{\partial}{\partial b_{kl}} L = \log(v_{kl}) - \log\left(\sum_{i \in \mathcal{A}} b_{il}\right) - 1 - \mu_k - \nu_l,$$

so the optimality conditions are

$$\frac{v_{ij}}{\sum_{i' \in \mathcal{A}} b_{i'j}} = e \cdot e^{\mu_i} \cdot e^{\nu_j},$$

for all $i \in \mathcal{A}$ and $j \in \mathcal{G}$. As we have already seen, the optimality conditions for the solution x, y imply that

$$\frac{v_{ij}}{\sum_{i' \in \mathcal{A}} b_{i'j}} = \frac{\sum_{j' \in \mathcal{G}} x_{ij'} v_{ij'} y_{j'}}{y_j},$$

where the numerator depends only on i and the denominator depends only on j. By choosing the appropriate scaling, we can ensure that $\nu_j \geq 0$ for all j, while ensuring that the new solution does satisfy all the optimality conditions. □

4.5.1 Generalization to Weighted Nash Social Welfare

In this section we provide an alternative proof of Theorem 2 by showing a chain of dual programs and changes of variables that can transform one program into the other. The same chain of duals is also how we derive the convex relaxation (CVX-Weighted). In order to avoid repeating the entire argument twice, we present it for the first time in the weighted setting. The weights do not add additional complications, and the original proof can be obtained by replacing all weights w_i with the uniform $1/n$ throughout.

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 property.

Assumption Given an instance of Nash Social Welfare $(\mathcal{A}, \mathcal{G}, \mathbf{v}, \mathbf{w})$, the support graph is the bipartite graph between agents and items with an edge between agent i and item j if and only if $v_{ij} > 0$. We assume that there exists a matching of size $|\mathcal{A}|$ in the support graph. In other words, the objective of the Nash Social Welfare problem is non-zero. It is straightforward to verify this assumption given an instance of Nash Social Welfare.

This proof of Theorem 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 [78].

Theorem 26 (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 [43].

Lemma 21 (Lemma 4.3 in [43]) *Let $p : \mathbf{R}^m_{\geq 0} \to \mathbf{R}_{\geq 0}$ 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* $t \in \mathbf{R}$,
- $\log p(\mathbf{y})$ *is convex in* $\log \mathbf{y}$.

Then the following inequality holds

$$\inf_{\mathbf{y}>0:\prod_{j\in S} y_h\geq 1, \forall S\in\binom{[m]}{n}} \log p(\mathbf{y}) = \sup_{\alpha\in[0,1]^m, \sum_j \alpha_j = n} \inf_{\mathbf{y}>0} \quad \log p(\mathbf{y}) - \sum_{j=1}^{m} \alpha_j \log(y_j).$$

While the original result in [43] assumed p to be a homogeneous polynomial with positive coefficients, their proof only relies on the two properties presented in Lemma 21.

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 $\mathbf{w} \geq \mathbf{0}$, 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 27 *The optimal values of* (LogConcave-Weighted) *and* (CVX-Weighted) *are the same.*

By replacing the weights w_i with $1/n$, we can see that this is a generalization of Theorem 2.

Starting with (LogConcave-Weighted) we derive (CVX-Weighted) via a sequence of duals presented in Lemmas 22, 23, and 24.

Lemma 22 *The optimal value of* (LogConcave-Weighted) *is the same as*

$$\inf_{\delta} \max_{\mathbf{x}\in\mathcal{P}} \sum_{i\in\mathcal{A}} w_i \log\left(\sum_{j\in\mathcal{G}} x_{ij} v_{ij} e^{-\delta_j/w_i}\right) + \sum_{j\in\mathcal{G}} \max(0,\delta_j). \qquad \text{(Weighted-Primal)}$$

The following fact is crucial to the proof of this lemma.

Fact 1 *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 21. The first property is easy to verify and the second property follows from Fact 1. Therefore, by Lemma 21, we get

$$\begin{aligned}\inf_{\mathbf{y}\in\mathcal{Q}} \log p_{\mathbf{x}}(\mathbf{y}) &= \inf_{\mathbf{y}\in\mathcal{Q}} \sum_{i\in\mathcal{A}} w_i \log\left(\sum_{j\in\mathcal{G}} x_{ij}\, v_{ij}\, y_j^{1/w_i}\right) \\ &= \sup_{\alpha\in[0,1]^{\mathcal{G}}, \sum_j \alpha_j = n} \inf_{\mathbf{y}>0} \sum_{i\in\mathcal{A}} w_i \log\left(\sum_{j\in\mathcal{G}} x_{ij}\, v_{ij}\, y_j^{1/w_i}\right) - \sum_{j\in\mathcal{G}} \alpha_j \log(y_j).\end{aligned}$$

Substituting $\delta_j = -\log(y_j)$, and taking the supremum over $\mathbf{x}$, we get

$$\begin{aligned}
&\max_{\mathbf{x}\in\mathcal{P}} \inf_{\mathbf{y}\in\mathcal{Q}} \log p_{\mathbf{x}}(\mathbf{y}) \\
&\qquad = \sup_{\mathbf{x}\in\mathcal{P},\alpha\in[0,1]^{\mathcal{G}},\sum_j \alpha_j = n} \inf_{\delta} \sum_{i\in\mathcal{A}} w_i \log\left(\sum_{j\in\mathcal{G}} x_{ij}\, v_{ij}\, e^{-\delta_j/w_i}\right) + \sum_{j\in\mathcal{G}} \alpha_j \delta_j.
\end{aligned}$$

As the domains of both $\mathbf{x}$ and α are compact, using Theorem 26, we get

$$\begin{aligned}
&\max_{\mathbf{x}\in\mathcal{P}} \inf_{\mathbf{y}\in\mathcal{Q}} \sum_{i\in\mathcal{A}} w_i \log\left(\sum_{j\in\mathcal{G}} x_{ij}\, v_{ij}\, y_j^{1/w_i}\right) \\
&\qquad = \inf_{\delta} \max_{\mathbf{x}\in\mathcal{P}} \max_{\alpha\in[0,1]^{\mathcal{G}},\sum_j \alpha_j = n} \sum_{i\in\mathcal{A}} w_i \log\left(\sum_{j\in\mathcal{G}} x_{ij}\, v_{ij}\, e^{-\delta_j/w_i}\right) + \sum_{j\in\mathcal{G}} \alpha_j \delta_j.
\end{aligned}$$

Finally, we claim that

$$\begin{aligned}
&\inf_{\delta} \max_{\mathbf{x}\in\mathcal{P}} \max_{\alpha\in[0,1]^{|\mathcal{G}|},\sum_j \alpha_j = n} \sum_{i\in\mathcal{A}} w_i \log\left(\sum_{j\in\mathcal{G}} x_{ij}\, v_{ij}\, e^{-\delta_j/w_i}\right) + \sum_{j\in\mathcal{G}} \alpha_j \delta_j \\
&\qquad = \inf_{\delta} \max_{\mathbf{x}\in\mathcal{P}} \sum_{i\in\mathcal{A}} w_i \log\left(\sum_{j\in\mathcal{G}} x_{ij}\, v_{ij}\, e^{-\delta_j/w_i}\right) + \sum_{j\in\mathcal{G}} \max(0, \delta_j).
\end{aligned}$$

The key observation is again that if we shift δ by $t \cdot \mathbb{1}$, the change in the two terms cancels out exactly, and the objective does not change.

Let $f(\delta, \mathbf{x}, \alpha)$ denote the inner function

$$f(\delta, \mathbf{x}, \alpha) = \sum_{i\in\mathcal{A}} w_i \log\left(\sum_{j\in\mathcal{G}} x_{ij}\, v_{ij}\, e^{-\delta_j/w_i}\right) + \sum_{j\in\mathcal{G}} \alpha_j \delta_j,$$

and note that $f(\delta, \mathbf{x}, \alpha) = f(\delta + t \cdot \mathbb{1}, \mathbf{x}, \alpha)$ for all $t \in \mathbf{R}$; the change in the two sums exactly cancel out since $\sum_{j\in\mathcal{G}} \alpha_j = |\mathcal{A}|$.

Thus, we can assume that δ has exactly n non-negative entries, and the remaining $m-n$ entries are negative. With this assumptions, it is clear that the optimal choice of α is to set

$\alpha_j = 1$ if $\delta_j \geq 0$ and $\alpha_j = 0$ otherwise. In this case,

$$\sum_{j \in \mathcal{G}} \alpha_j \delta_j = \sum_{j \in \mathcal{G}} \max(0, \delta_j).$$

In the final formulation

$$\inf_{\delta} \max_{\mathbf{x} \in \mathcal{P}} \sum_{i \in \mathcal{A}} w_i \log \left(\sum_{j \in \mathcal{G}} x_{ij}\, v_{ij}\, e^{-\delta_j / w_i} \right) + \sum_{j \in \mathcal{G}} \max(0,\ \delta_j),$$

the best choice of δ will also have exactly n non-negative entries, as otherwise shifting uniformly results in a decrease in the objective. Thus, the two programs are equivalent. $\square$

Lemma 23 *The optimal value of* (Weighted-Primal) *is the same as that of the following program.*

$$\begin{aligned} \inf_{\delta \geq 0, \mathbf{r}, \gamma} \quad & \sum_{j \in \mathcal{G}} e^{r_j} + \sum_{i \in \mathcal{A}} w_i \gamma_i + \sum_{j \in \mathcal{G}} \delta_j + \sum_{i \in \mathcal{A}} (w_i \log w_i - w_i) \qquad \text{(Weighted-Dual)} \\ & r_j + \gamma_i + \frac{\delta_j}{w_i} \geq \log v_{ij} \quad \forall (i,j) \in \mathcal{A} \times \mathcal{G}. \end{aligned}$$

Proof For a fixed δ, let us first re-write the internal maximum of (Weighted-Primal) as

$$\begin{aligned} \max_{\mathbf{x}, \mathbf{u}} \quad & \sum_{i \in \mathcal{A}} w_i \log u_i + f(\delta) \qquad (4.2) \\ & u_i \leq \sum_{j \in \mathcal{G}} x_{ij}\, v_{ij}\, e^{-\delta_j / w_i} \quad \forall i \in \mathcal{A} \\ & \sum_{i \in \mathcal{A}} x_{ij} \leq 1 \quad \forall j \in \mathcal{G} \\ & \mathbf{x} \geq 0\,, \end{aligned}$$

where $f(\delta) = \sum_{j \in \mathcal{G}} \max(0, \delta_j)$.

Let β_i, p_j, and θ_{ij} be the Lagrange dual variables associated with the constraints corre-

sponding to agent i, item j, and agent-item pair (i, j), respectively. The Lagrangian of the above convex program is defined as follows

$$\begin{aligned}
&L(\mathbf{x}, \mathbf{u}, \beta,\theta, \mathbf{p}) \\
&= f(\delta)+ \sum_{i\in\mathcal{A}} w_i \log u_i + \sum_{i\in\mathcal{A}} \beta_i \left(\sum_{j\in\mathcal{G}} x_{ij}\, v_{ij}\, e^{-\delta_j/w_i} - u_i \right) \\
&\qquad + \sum_{j\in\mathcal{G}} p_j \left(1 - \sum_{i\in\mathcal{A}} x_{ij} \right) + \sum_{i,j} \theta_{ij} x_{ij} \\
&= f(\delta)+ \left[\sum_{i\in\mathcal{A}} (w_i \log u_i - \beta_i u_i) + \sum_{i,j} x_{ij} \left(\beta_i\, v_{ij}\, e^{-\delta_j/w_i} + \theta_{ij} - p_j \right) + \sum_{j\in\mathcal{G}} p_j \right].
\end{aligned}$$

The Lagrange dual of (4.2) is given by

$$g(\beta, \theta, \mathbf{p}) = \max_{\mathbf{x}\in\mathcal{P}, \mathbf{u}\geq 0} L(\mathbf{x}, \mathbf{u}, \beta, \theta, \mathbf{p}).$$

Once again, it is easy to verify that Slater's condition is satisfied. Thus the optimum value of the infimum of Lagrange dual over $\beta, \theta, \mathbf{p} \geq 0$ is exactly equal to the optimum of (4.2).

The KKT conditions for the Lagrangian imply

$$\begin{aligned}
\frac{w_i}{u_i} - \beta_i = 0 \quad \forall i \in \mathcal{A} \\
\beta_i\, v_{ij}\, e^{-\delta_j/w_i} - p_j + \theta_{ij} = 0 \quad \forall (i, j) \in \mathcal{A} \times \mathcal{G}.
\end{aligned}$$

Thus $u_i = w_i/\beta_i$ for each $i \in \mathcal{A}$ maximizes the Lagrangian. For the supremum over $\mathbf{x}, \mathbf{u}$ to stay finite, the second KKT condition is necessary and sufficient. Substituting these conditions in the Langrangian gives the following convex program.

$$\begin{aligned}
\inf_{\mathbf{p},\beta,\theta} \quad & f(\delta) + \sum_{j\in\mathcal{G}} p_j + \sum_{i\in\mathcal{A}} (w_i \log w_i - w_i) - \sum_{i\in\mathcal{A}} w_i \log \beta_i \\
& p_j = \beta_i\, v_{ij}\, e^{-\delta_j/w_i} + \theta_{ij} \quad \forall (i, j) \in \mathcal{A} \times \mathcal{G}
\end{aligned}$$

$$\mathbf{p}, \beta, \theta \geq 0$$

Observe that we can remove θ 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

$$\begin{aligned} \inf_{\mathbf{r},\gamma} \quad & f(\delta) + \sum_{j\in\mathcal{G}} e^{r_j} + \sum_{i\in\mathcal{A}} w_i\,\gamma_i + \sum_{i\in\mathcal{A}} (w_i \log w_i - w_i) \\ & r_j + \gamma_i + \frac{\delta_j}{w_i} \geq \log v_{ij} \quad \forall (i,j) \in \mathcal{A}\times\mathcal{G}. \end{aligned}$$

As (Weighted-Primal) involves an infimum over δ, whenever $\delta_j < 0$, we can increase it to $\delta_j = 0$ without increasing the value of $f(\delta)$ and maintaining feasibility in the above program. Using this observation and taking an infimum over δ gives (Weighted-Dual). $\square$

Lemma 24 *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 (Weighted-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

$$\begin{aligned} L(\mathbf{r},\gamma,\delta,\hat{\mathbf{b}},\tau) &= \sum_{j\in\mathcal{G}} e^{r_j} + \sum_{i\in\mathcal{A}} w_i\,\gamma_i + \sum_{j\in\mathcal{G}} \delta_j + \sum_{i,j} \hat{b}_{ij}(\log v_{ij} - r_j - \gamma_i - \frac{\delta_j}{w_i}) \\ &\quad - \sum_{j\in\mathcal{G}} \delta_j\tau_j + \sum_{i\in\mathcal{A}} (w_i \log w_i - w_i) \\ &= \sum_{j\in\mathcal{G}} (e^{r_j} - (\sum_{i\in\mathcal{A}} \hat{b}_{ij}) r_j) + \sum_{i\in\mathcal{A}} \gamma_i (w_i - \sum_{j\in\mathcal{G}} \hat{b}_{ij}) + \sum_{j\in\mathcal{G}} \delta_j (1 - \tau_j - \sum_{i\in\mathcal{A}} \frac{\hat{b}_{ij}}{w_i}) \\ &\quad + \sum_{i,j} \hat{b}_{ij} \log v_{ij} + \sum_{i\in\mathcal{A}} (w_i \log w_i - w_i). \end{aligned}$$

The Lagrange dual of (Weighted-Dual) is given by

$$g(\hat{\mathbf{b}}, \tau) = \inf_{\delta \geq 0, \mathbf{r}, \gamma} L(\mathbf{r}, \gamma, \delta, \hat{\mathbf{b}}, \tau). \tag{4.3}$$

One can verify that Slater's condition is satisfied by (Weighted-Dual). So, the supremum of (4.3) with $\mathbf{b}, \tau \geq 0$ 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 γ, δ 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

$$\begin{aligned} \sup_{\hat{\mathbf{b}}, \tau} \quad & \sum_{i,j} \hat{b}_{ij} \log v_{ij} - \sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} \hat{b}_{ij} \log\left(\sum_{i \in \mathcal{A}} \hat{b}_{ij}\right) + \sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} \hat{b}_{ij} + \sum_{i \in \mathcal{A}} (w_i \log w_i - w_i) \\ & \sum_{j \in \mathcal{G}} \hat{b}_{ij} = w_i \\ & \sum_{i \in \mathcal{A}} \frac{\hat{b}_{ij}}{w_i} = 1 - \tau_j \\ & \hat{\mathbf{b}}, \tau \geq \mathbf{0}. \end{aligned}$$

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 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 25 *The two programs* (NCVX-Weighted) *and* (CVX-Weighted) *are relaxations of 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 $V_i = \sum_{j \in \sigma^{-1}(i)} v_{ij}$. 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 $\sum_{j \in \mathcal{G}} b_{ij} = 1$ 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 $\sum_{i \in \mathcal{A}} w_i \log V_i$, which is exactly $\text{NSW}(\sigma)$. The key observation is that for each $j \in \mathcal{G}$

$$\sum_{i \in \mathcal{A}} w_i b_{ij} = \frac{w_{\sigma(j)} v_{\sigma(j)j}}{V_{\sigma(j)}}.$$

With this, we can complete the calculations.

$$\begin{aligned} f_{\text{cvx}}(\mathbf{b}) &= \sum_{j \in \mathcal{G}} \frac{w_{\sigma(j)} v_{\sigma(j)j}}{V_{\sigma(j)}} \log v_{\sigma(j)j} - \sum_{j \in \mathcal{G}} \frac{w_{\sigma(j)} v_{\sigma(j)j}}{V_{\sigma(j)}} \log \left(\frac{w_{\sigma(j)} v_{\sigma(j)j}}{V_{\sigma(j)}} \right) + \sum_{i \in \mathcal{A}} w_i \log w_i \\ &= \sum_{j \in \mathcal{G}} \frac{w_{\sigma(j)} v_{\sigma(j)j}}{V_{\sigma(j)}} \log \left(\frac{V_{\sigma(j)}}{w_{\sigma(j)}} \right) + \sum_{i \in \mathcal{A}} w_i \log w_i \\ &= \sum_{i \in \mathcal{A}} w_i \sum_{j \in \sigma^{-1}(i)} \frac{v_{ij}}{V_i} \log \left(\frac{V_i}{w_i} \right) + \sum_{i \in \mathcal{A}} w_i \log w_i \end{aligned}$$

$$= \sum_{i \in \mathcal{A}} w_i \log \left(\frac{V_i}{w_i} \right) + \sum_{i \in \mathcal{A}} w_i \log w_i = \sum_{i \in \mathcal{A}} w_i \log V_i,$$

where the last line follows from definition of V_i.

Similarly, we have

$$\begin{aligned} f_{\text{ncvx}}(\mathbf{b}) &= \sum_{j \in \mathcal{G}} \frac{w_{\sigma(j)} v_{\sigma(j)j}}{V_{\sigma(j)}} \log v_{\sigma(j)j} - \sum_{j \in \mathcal{G}} \frac{w_{\sigma(j)} v_{\sigma(j)j}}{V_{\sigma(j)}} \log \left(\frac{v_{\sigma(j)j}}{V_{\sigma(j)}} \right) \\ &= \sum_{j \in \mathcal{G}} \frac{w_{\sigma(j)} v_{\sigma(j)j}}{V_{\sigma(j)}} \log V_{\sigma(j)} = \sum_{i \in \mathcal{A}} w_i \sum_{j \in \sigma^{-1}(i)} \frac{v_{ij}}{V_i} \log V_i \\ &= \sum_{i \in \mathcal{A}} w_i \log V_i. \end{aligned}$$

□

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 Section 4.5.1 we used it to derive (CVX-Weighted).

Lemma 26 *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

$$\sum_{i \in \mathcal{A}} w_i \log \left(\sum_{j \in \mathcal{G}} x_{ij} \; v_{ij} \; y_j^{1/w_i} \right) = \sum_{i \in \mathcal{A}} w_i \log V_i = \text{NSW}(\sigma).$$

Now we will show that the infimum over all y 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 $\sum\limits_{j=1}^{m} c_j = 1$, and $0 \leq p \leq q$, the weighted power mean inequality states that for any $\mathbf{z} \in \mathbf{R}^m_{\geq 0}$,

$$\left(\sum_{j=1}^{m} c_j z_j^p \right)^{1/p} \leq \left(\sum_{j=1}^{m} c_j z_j^q \right)^{1/q}. \tag{4.5}$$

This inequality follows from Jensen's inequality.

For each $i \in \mathcal{A}$, define $q_i = \frac{1}{w_i}$ and $c_j^{(i)} = \frac{v_{ij}}{\sum\limits_{j \in S_i} v_{ij}}$ for every $j \in S_i$. Since $q_i = \frac{1}{w_i} \geq 1$, using equation (4.5), we get

$$w_i \log \left(\frac{\sum_{j \in S_i} v_{ij}\, y_j^{1/w_i}}{\sum\limits_{j \in S_i} v_{ij}} \right) \geq \log \left(\frac{\sum_{j \in S_i} v_{ij}\, y_j}{\sum\limits_{j \in S_i} v_{ij}} \right)$$

for each agent i. Next, we need to apply Lemma 18, the same key inequality used in [43].

$$\sum_{i \in \mathcal{A}} \log \left(\sum_{j \in S_i} v_{ij} y_j \right) \geq \sum_{i \in \mathcal{A}} \log \left(\sum_{j \in S_i} v_{ij} \right)$$

A complete proof is given in the background material within Section 4.2.

Applying the lemma and summing over the previous inequality gives

$$\sum_{i \in \mathcal{A}} w_i \log \left(\frac{\sum_{j \in S_i} v_{ij}\, y_j^{1/w_i}}{\sum\limits_{j \in S_i} v_{ij}} \right) \geq \sum_{i \in \mathcal{A}} \log \left(\frac{\sum_{j \in S_i} v_{ij}\, y_j}{\sum\limits_{j \in S_i} v_{ij}} \right) \geq 0.$$

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 appeared in Section 4.5.1, 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 19 *For any any feasible* $\mathbf{b} \in \mathcal{P}(\mathcal{A}, \mathcal{G})$ *and positive 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.*

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

$$\begin{aligned}
f_{\text{cvx}}(\mathbf{b}) - f_{\text{ncvx}}(\mathbf{b}) &= \sum_{i\in\mathcal{A}} w_i \log w_i - \sum_{j\in\mathcal{G}}\sum_{i\in\mathcal{A}} w_i\, b_{ij} \log\left(\sum_{i'\in\mathcal{A}} w_{i'}\, b_{i'j}\right) \\
&\quad + \sum_{i,j} w_i\, b_{ij} \log\left(\sum_{i'\in\mathcal{A}} b_{i'j}\right) \\
&= \sum_{i\in\mathcal{A}} w_i \log w_i - \sum_{j\in\mathcal{G}}\sum_{i\in\mathcal{A}} w_i\, b_{ij} \log\left(\frac{\sum_{i'\in\mathcal{A}} w_{i'}\, b_{i'j}}{\sum_{i'\in\mathcal{A}} b_{i'j}}\right) \\
&= \sum_{i\in\mathcal{A}} w_i \log w_i + \sum_{j\in\mathcal{G}}\sum_{i\in\mathcal{A}} w_i\, b_{ij} \log n - \sum_{j\in\mathcal{G}} \mu(j) \log\left(\frac{\mu(j)}{\theta(j)}\right) \\
&= \sum_{i\in\mathcal{A}} w_i \log(n w_i) - \sum_{j\in\mathcal{G}} \mu(j) \log\left(\frac{\mu(j)}{\theta(j)}\right) \qquad \text{(using } \textstyle\sum_j b_{ij} = 1\text{)} \\
&= D_{\text{KL}}(\mathbf{w}\,||\,\mathbf{u}) - D_{\text{KL}}(\mu\,||\,\theta).
\end{aligned}$$

As $D_{KL}(\mu, \theta) \geq 0$, the above equation implies

$$f_{\text{cvx}}(\mathbf{b}) - f_{\text{ncvx}}(\mathbf{b}) \leq D_{\text{KL}}(\mathbf{w}\,||\,\mathbf{u}).$$

For the lower bound, it suffices to show that $D_{\text{KL}}(\mu\,||\,\theta) \leq D_{\text{KL}}(\mathbf{w}\,||\,\mathbf{u})$. To see this, we expand the definition:

$$\begin{aligned}
D_{\text{KL}}(\mu\,||\,\theta) &= \sum_{j\in\mathcal{G}} \left(\sum_{i\in\mathcal{A}} w_i b_{ij}\right) \log\left(\frac{n\sum_{i\in\mathcal{A}} w_i b_{ij}}{\sum_{i\in\mathcal{A}} b_{ij}}\right) \\
&= \log(n) + \sum_{j\in\mathcal{G}} \left(\sum_{i\in\mathcal{A}} w_i b_{ij}\right) \log\left(\frac{\sum_{i\in\mathcal{A}} w_i b_{ij}}{\sum_{i\in\mathcal{A}} b_{ij}}\right) \\
&= \log(n) + \sum_{j\in\mathcal{G}} \left(\sum_{i\in A} b_{ij}\right) \left(\sum_{i\in\mathcal{A}} \frac{b_{ij}}{\sum_{i\in A} b_{ij}} w_i\right) \log\left(\sum_{i\in\mathcal{A}} \frac{b_{ij}}{\sum_{i\in A} b_{ij}} w_i\right) \\
&\leq \log(n) + \sum_{j\in\mathcal{G}} \left(\sum_{i\in A} b_{ij}\right) \left(\sum_{i\in\mathcal{A}} \frac{b_{ij}}{\sum_{i\in A} b_{ij}}\right) w_i \log(w_i) \\
&= \log(n) + \sum_{j\in\mathcal{A}} w_i \log(w_i) \sum_{j\in\mathcal{G}} b_{ij}
\end{aligned}$$

$$= \log(n) + \sum_{i \in \mathcal{A}} w_i \log(w_i) = D_{\mathrm{KL}}(\mathbf{w} \,||\, \mathbf{u}).$$

Here, the only inequality uses the convexity of $x \log(x)$, and the last equality follows from the feasibility of $\mathbf{b}$. $\square$

4.7 Algorithm Overview

For ease of notation, given any feasible point $\mathbf{b} \in \mathcal{P}(\mathcal{A}, \mathcal{G})$, we use vector $\mathbf{q} \in \mathbf{R}^{|\mathcal{G}|}$ 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 $f_{\mathrm{ncvx}}(\mathbf{b}, \mathbf{q})$ and $f_{\mathrm{cvx}}(\mathbf{b}, \mathbf{q})$ to also denote the objective $f_{\mathrm{ncvx}}(\mathbf{b})$ and $f_{\mathrm{cvx}}(\mathbf{b})$, respectively. With a slight abuse of notation, we define

$$f_{\mathrm{ncvx}}(\mathbf{b}, \mathbf{q}) := \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{G}} w_i \, b_{ij} \log v_{ij} - \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{G}} w_i \, b_{ij} \log q_j.$$

for any $\mathbf{b} \in \mathcal{P}(\mathcal{A}, \mathcal{G})$ and its projection $\mathbf{q} \in \mathbf{R}^{|G|}$.

The main result of this section is to provide an approximation algorithm which relies on a subroutine which rounds acyclic feasible solutions. Our main algorithm, Algorithm 6, begins by finding the optimal solution $\overline{\mathbf{b}}$ to the convex program (CVX-Weighted). It then constructs another feasible point, $\mathbf{b}^{\mathrm{forest}}$, in support of $\overline{\mathbf{b}}$ such that the support graph of $\mathbf{b}^{\mathrm{forest}}$ is a forest and f_{ncvx} at $\mathbf{b}^{\mathrm{forest}}$ is at least f_{ncvx} at $\overline{\mathbf{b}}$. In the final step, the algorithm rounds $\mathbf{b}^{\mathrm{forest}}$ to an integral solution using Algorithm 7. In the next section we prove Theorem 25, which establishes a bound on the rounding error incurred during Algorithm 7.

Algorithm 6 Approximation Algorithm for Weighted Nash Social Welfare

Input. NSW instance $(\mathcal{A}, \mathcal{G}, \mathbf{v}, \mathbf{w})$

$\overline{\mathbf{b}} \leftarrow$ optimal solution of (CVX-Weighted)

$\overline{\mathbf{q}} \leftarrow$ vector in $\mathbf{R}^{|\mathcal{G}|}$ with $\overline{q}_j = \sum_{i \in \mathcal{A}} \overline{b}_{ij}$

$(\mathbf{b}^{\text{forest}}, \mathbf{q}^{\text{forest}}) \leftarrow$ acyclic solution in support of $\overline{\mathbf{b}}$ such that $f_{\text{ncvx}}(\mathbf{b}^{\text{forest}}) \geq f_{\text{ncvx}}(\overline{\mathbf{b}})$

$\sigma \leftarrow$ output of Algorithm 7 with input $(\mathcal{A}, \mathcal{G}, \mathbf{v}, \mathbf{w}, \mathbf{b}^{\text{forest}}, \mathbf{q}^{\text{forest}})$

Return σ

Lemma 20, 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 [71, 42] 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). The reason for this difference is that if we fix the sum $\sum_{i \in A} b_{ij}$ for each $j \in \mathcal{G}$, the program (NCVX-Weighted) becomes linear in the b variables, and the constraints are still simple assignment constraints.

Lemma 20 *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 $G_{\text{supp}}(\overline{\mathbf{b}})$ 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 $\overline{\mathbf{b}}$ on this cycle while ensuring the value of $\overline{\mathbf{q}}$ does not change. If $\overline{\mathbf{q}}$ is fixed, then $f_{\text{ncvx}}(\cdot, \overline{\mathbf{q}})$ is linear in the input, and as a result, we can *cancel* the cycle by considering the following vector. Define $\boldsymbol{\delta} \in \mathbf{R}^{|\mathcal{A}| \times |\mathcal{G}|}$ 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 $j \in \mathcal{G}$,

$$\sum_{i \in \mathcal{A}} \left(\bar{b}_{ij} + \varepsilon \delta_{ij} \right) = \sum_{i \in \mathcal{A}} \bar{b}_{ij} = \bar{q}_j.$$

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 19 to bound the difference between f_{cvx} and f_{ncvx}.

By combining Lemma 20 with Lemma 19, we obtain the following corollary.

Corollary 1 *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 19 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 20 to get an acyclic solution $\mathbf{b}^{\text{forest}}$, in the support of $\overline{\mathbf{b}}$ such that

$$f_{\text{ncvx}}(\bar{\mathbf{b}}) \leq f_{\text{ncvx}}(\mathbf{b}^{\text{forest}}).$$

Finally, using the other side of Lemma 19, 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 7, we give the proof of Theorem 4, which now follows directly from Theorem 25 and Corollary 1, as outlined below.

Proof of Theorem 4 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 6, respectively. Let $\sigma^\star$ be the assignment returned by Algorithm 7 on input $(\mathbf{b}^{\text{forest}}, \mathbf{q}^{\text{forest}})$. By Theorem 25, 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 1 and (ii) follows from Lemma 25. □

4.8 Rounding an Acyclic Solution by Pruning and Matching

It remains to show that we can efficiently round an acyclic feasible solution without too much additional loss. In this section, we prove Theorem 25 by establishing properties of support-restricted optimal solutions of (CVX-Weighted), and using them to round solutions in Algorithm 7.

Algorithm 7 Algorithm for Rounding an Acyclic Solution

Input. NSW instance $(\mathcal{A}, \mathcal{G}, \mathbf{v}, \mathbf{w})$, acyclic solution $(\mathbf{b}, \mathbf{q}) \in \mathcal{P}(\mathcal{A}, \mathcal{G})$

$(\mathbf{b}^\star, \mathbf{q}^\star) \leftarrow$ optimal solution of (CVX-Weighted) restricted to the support of $(\mathbf{b}, \mathbf{q})$

$F^\star \leftarrow G_{\text{supp}}(\mathbf{b}^\star)$ with every tree rooted at an agent node

$\widetilde{F} \leftarrow$ Forest obtained by removing edges between item j and its children in $F^\star$ whenever $q_j^\star < 1/2$ **(Pruning)**

$L_i^\star \leftarrow$ set of leaf children of agent i in $\widetilde{F}$ and let $L^\star = \cup_i L_i^\star$

$M^\star \leftarrow$ matching between $\mathcal{A} \to \mathcal{G} \backslash L^\star$ in $\widetilde{F}$ which maximizes weight function $w_{\widetilde{F}}(M) := \sum_{i \in \mathcal{A}} w_i \log\left(v_{iM(i)} + \sum_{j \in L_i^\star} v_{ij}\right)$ [1]

$\sigma^\star \leftarrow$ assignment of $\mathcal{G}$ to $\mathcal{A}$ with $\sigma^\star(j) = i$ if $j \in L_i^\star \cup M^\star(i)$ **(Matching)**

Return $\sigma^\star$

In the first step, Algorithm 7 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).

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 $f_{\text{cvx}}(\mathbf{b}^\star)$. This bound is summarized in the following lemma.

[1] If agent i in unmatched in M, we let $v_{iM(i)} = 0$

Lemma 27 *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 27 relies on the stability properties of optimal solutions to the convex relaxation (CVX-Weighted), as outlined in Section A.1.

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 A.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.

The results of the matching step are summarized in the following lemma.

Lemma 28 *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 A.2. With these two guarantees, we are ready to prove the main result for rounding acyclic solutions.

Proof of Theorem 25 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 27 on input $(\mathbf{b}^\star, \mathbf{q}^\star)$. Since Lemma 27 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 27, 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 28 with input $(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}})$.

Using Lemma 28 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 7 satisfies

$$\begin{aligned}\text{NSW}(\sigma^\star) = \sum_{i\in\mathcal{A}} w_i \log\left(v_{iM^\star(i)} + \sum_{j\in L_i^\star} v_{ij}\right) &\overset{(i)}{\geq} \sum_{i\in\mathcal{A}} w_i \log\left(v_{iM(i)} + \sum_{j\in L_i^\star} v_{ij}\right) \\ &\overset{(ii)}{\geq} f_{\text{ncvx}}(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}}) - \log 2 - \frac{1}{2e} \\ &\overset{(iii)}{\geq} f_{\text{cvx}}(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}}) - D_{\text{KL}}(\mathbf{w}\,||\,\mathbf{u}) - \log 2 - \frac{1}{2e} \\ &\overset{(iv)}{\geq} f_{\text{cvx}}(\mathbf{b}^\star, \mathbf{q}^\star) - D_{\text{KL}}(\mathbf{w}\,||\,\mathbf{u}) - 2\log 2 - \frac{1}{2e} \\ &\overset{(v)}{\geq} f_{\text{cvx}}(\mathbf{b}, \mathbf{q}) - D_{\text{KL}}(\mathbf{w}\,||\,\mathbf{u}) - 2\log 2 - \frac{1}{2e}.\end{aligned}$$

Here, (i) follows from the optimality of $M^\star$, (ii) follows from Lemma 28, (iii) follows from Lemma 19, (iv) follows from Lemma 27, and (v) follows from the optimality of $\mathbf{b}^\star$. $\square$

4.9 Recent Related Work and Open Questions

After the initial release of this work there has been substantial progress improving the approximation factor. For the case of additive [73] and even the more general submodular valuations [74] there are now constant factor approximation algorithms. Their approach is based on rounding a fractional solution to the configuration formulation of the problem, and the algorithm itself is quite simple. The configuration 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 $\exp\left(D_{\mathrm{KL}}(\mathbf{w} \,||\, \mathbf{u})\right)$ 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.

CHAPTER 5

MAXIMIZING THE MINIMUM EIGENVALUE IN CONSTANT DIMENSION

This chapter is based on joint work with Aditi Laddha and Mohit Singh [79] which appeared in Operations Research Letters.

In an instance of a minimum eigenvalue problem, we are given a collection of n vectors $v_1, \ldots, v_n \in \mathbf{R}^d$, and the goal is to pick a subset $B \subseteq [n]$ of given vectors to maximize the minimum eigenvalue of the matrix $\sum_{i \in B} v_i v_i^\top$. The selected set B must satisfy additional constraints such as cardinality, partition, or more generally matroid constraints. While much of the focus in previous works [13, 1, 2] has been on cardinality constraints, in this work, we consider general matroid constraints. The generality of matroid constraints allows us to model the algorithmic version of the Kadison Singer problem [80, 45] as well as the Santa Claus allocation problem [16, 81, 17, 82, 19] as a special case of the minimum eigenvalue problem.

The "discrepancy" formulation of the Kadison-Singer problem (shown to be equivalent to the original formulation in [83] and proved in [80]) states that given a set of vectors $v_1, \ldots, v_m \in \mathbf{R}^d$ with $\|v_i\| \leq \alpha$ and $\sum_i v_i v_i^\top = I_d$, there exists a partition of $[m]$ into two subsets S_1, S_2, such that for every j, $\sum_{i \in S_j} v_i v_i^\top$ spectrally approximates $I_d/2$ to an additive factor of $O(\alpha)$. Algorithmically, finding such a partition is equivalent to solving an instance of the minimum eigenvalue maximization problem under partition matroid constraints (see Section 5.3.1).

Another classical application of the minimum eigenvalue problem arises in the area of optimal design of experiments in statistics [3, 13]. The goal in the design of experiments is to select a subset of vectors S from a given list of vectors $\{v_1, \ldots, v_n\}$ such that certain measures of the covariance matrix $\left(\sum_{i \in S} v_i v_i^\top\right)^{-1}$ are small. In particular, minimizing the maximum eigenvalue of the covariance matrix, classically known as the E-design prob-

lem in statistics, is exactly the minimum eigenvalue problem. While much of the previous work has focused on the case when the selected set of measurements S must satisfy cardinality constraints, our work generalizes this problem to be studied under general matroid constraints.

In this chapter, we present an approximation algorithm for the minimum eigenvalue problem for all matroids. We use the randomized rounding technique of pipage rounding to give a polynomial time approximation scheme (PTAS) when the dimension is constant.

Theorem 5 *For any $\epsilon > 0$ there is an $O\left(n^{O(d\log(d)/\epsilon^2)}\right)$-time algorithm which, given a collection of vectors $v_1, \ldots, v_n \in \mathbf{R}^d$ and a matroid $\mathcal{M} = ([n], \mathcal{I})$ returns a set $B \in \mathcal{I}$ such that with probability at least $1 - d^{-4}$*

$$\lambda_{\min}\left(\sum_{i\in B} v_i v_i^\top\right) \geq (1-\epsilon)\cdot \max_{B^\star\in\mathcal{I}} \lambda_{\min}\left(\sum_{i\in B^\star} v_i v_i^\top\right).$$

Our result generalizes to give a PTAS (for constant dimension) when the objective is a general matrix function satisfying certain technical properties. In particular, this implies that a similar result as in Theorem 5 is achievable when the objective is to maximize the determinant of $\sum_{i\in B} v_i v_i^\top$ or to minimize any norm of the eigenvalues of $(\sum_{i\in B} v_i v_i^\top)^{-1}$.

Theorem 6 *Suppose we have a collection of vectors $\mathcal{V} = (v_1, \ldots, v_n) \in \mathbf{R}^d$, and a matroid $\mathcal{M} = ([n], \mathcal{I})$. Let $f : \mathbb{S}_d^+ :\to \mathbb{R}$ be a concave, monotone, and homogeneous function given with a value and first order oracle. For any $\epsilon > 0$, there is an $O\left(n^{O(d\log(d)/\epsilon^2)}\right)$-time randomized algorithm, which takes $(\mathcal{V}, \mathcal{M}, f)$ as input and returns a set $B \in \mathcal{I}$ such that with probability at least $1 - d^{-4}$,*

$$f\left(\sum_{i\in B} v_i v_i^\top\right) \geq (1-\epsilon)\cdot \max_{B^\star\in\mathcal{I}} f\left(\sum_{i\in B^\star} v_i v_i^\top\right).$$

Although Theorem 6 is stated in terms of maximizing concave functions, our algorithm can also be applied to minimize monotone and homogeneous convex functions (e.g.,

$\mathrm{trace}(\sum_{i\in B} v_i v_i^\top)^{-1})$ by considering the natural convex relaxation of the function over the matroid base polytope and using the same rounding strategy.

5.1 Technical Overview

The first natural direction is to construct a convex programming relaxation for the problem and aim to apply randomized rounding methods to it.

$$\begin{aligned} \max \quad & \lambda_{\min}(X) \\ & X = \sum_{i=1}^{n} x_i \cdot v_i v_i^\top \\ & x \in \mathcal{P}(\mathcal{M}) \\ & x \geq 0 \end{aligned} \tag{CP}$$

Here, $\mathcal{P}(\mathcal{M})$ denotes the matroid base polytope of $\mathcal{M}$. Unfortunately, this direct approach faces problems as this natural relaxation has an unbounded integrality gap even in very special cases (see Section 5.5). The main challenge is the presence of *long* vectors that contribute significantly towards the optimum solution.

A natural way to formalize the contribution of a vector is to consider its *leverage score*. Indeed, if T denotes the optimum solution and $A_T = \sum_{i\in T} v_i v_i^\top$, let $l_i = v_i^\top A_T^{-1} v_i$ be the leverage score of v_i and let $S = \{i \in T : l_i \geq \frac{\epsilon^2}{\log d}\}$ be the set of vectors in the optimum solution with large leverage scores. The bound $\frac{\epsilon^2}{\log d}$ is chosen to allow randomized rounding methods to work (see Corollary 3 for details). Using the simple fact that the sum of leverage scores of all vectors in the optimum solution is exactly d, it follows that $|S| \leq \frac{d \log d}{\epsilon^2}$. Thus we could easily enumerate all such subsets S in time $n^{O(d \log d/\epsilon^2)}$. For each such guess S, we include S in our solution and solve the convex program. We then apply the randomized rounding method to the solution of the convex program. Unfortunately, the challenge lies in ensuring that the convex program not only selects the vectors in S (this can be easily done by setting their indicator variable to one) but also avoids selecting all vectors *not* in T

that have a large leverage score. The latter is crucial for the randomized rounding approach to work effectively. Unfortunately, since we did not guess T, we have no way to insist that we do not pick these vectors in the convex program.

To address this problem, we present a structural lemma that enables us to compute the leverage score as given by matrix $A_S = \sum_{i \in S} v_i v_i^\top$. The lemma shows that there are few vectors with large leverage scores, even when using A_S instead of A_T. Since S is a subset of T, we know that $A_S \preceq A_T$ and hence $A_S^{-1} \succeq A_T^{-1}$. Therefore, the leverage scores with respect to A_S are larger. Nevertheless, we still show a similar bound in the following lemma.

Lemma 29 *For any set T and a set of vectors $\{v_i : i \in T\}$ in $\mathbb{R}^d$ such that $\sum_{i \in T} v_i v_i^\top$ is invertible, there exists a subset $S \subseteq T$ such that $|S| = O(d \log(d)/\epsilon^2)$, $A_S = \sum_{i \in S} v_i v_i^\top$ is invertible, and for all $i \in T \backslash S$,*

$$v_i^\top A_S^{-1} v_i \leq \frac{\epsilon^2}{10 \log(d)} .$$

With the help of Lemma 29, we can now guess the set S and insist that the convex program includes all these vectors in the chosen subset. More importantly, it allows us to insist that all vectors v_i not in S such that $v_i^\top A_S^{-1} v_i > \frac{\epsilon^2}{10 \log(d)}$ not be included in the chosen solution. The last step can be done since we have guessed the set S. This allows us to apply the randomized rounding approach to the convex programming solution.

There are some points worth mentioning about the randomized rounding approach. When the constraint matroid is a partition matroid, randomized rounding is a natural choice: for each part, the convex programming solution can be interpreted as a probability distribution over vectors in that part. Independently, for each part, pick one of the vectors with probability given by the convex programming solution. A simple application of the matrix Chernoff bound and the fact that leverage scores are all small due to Lemma 29 gives us the desired result. Due to the simplicity of the approach for partition matroids as well

as the applicability of these constraints, we first prove the result for partition matroids in Section 5.3. We also show the application of our result to obtain an algorithmic version of the Kadison-Singer problem [80] for constant dimension. We slightly improve the run time compared to the recent work [45].

Corollary 2 *Suppose we are given collection of vectors $\mathcal{U} = (u_1, \ldots, u_n) \in \mathbf{R}^d$ with $\|u_i\|^2 \leq \alpha$ for any $i \in [n]$ and $\sum_{i=1}^n u_i u_i^\top = I_d$ and a constant $c > 0$ such that there exists a set T^* satisfying*

$$\left(\frac{1}{2} - c\sqrt{\alpha}\right) I_d \preceq \sum_{i \in T^*} u_i u_i^\top \preceq \left(\frac{1}{2} + c\sqrt{\alpha}\right) I_d \,.$$

For any $\epsilon > 0$, there exists a randomized algorithm such which given $\mathcal{U}$ and c as input, returns a set T such that

$$(1-\epsilon) \cdot \left(\frac{1}{2} - c\sqrt{\alpha}\right) I_d \preceq \sum_{i \in T} u_i u_i^\top \preceq (1+\epsilon) \cdot \left(\frac{1}{2} + c\sqrt{\alpha}\right) I_d \,,$$

with probability at least $1 - O(d^{-4})$. The run time of the algorithm is $O(n^{O(d \log d/\epsilon^2)})$.

For general matroids, a straightforward application of randomized rounding does not work since it will not ensure that the chosen set is an independent set in the matroid. Instead, we use *pipage rounding* for general matroids, which involves randomly walking in the matroid polytope to return a vertex while ensuring that the output solution has even better concentration than is given by independent randomized rounding. To show these concentration results, we build on the work of Harvey and Olver [44] and give lower tail bounds on the distribution obtained via pipage rounding in Lemma 5.

5.2 Related Work

The minimum eigenvalue problem with partition constraints can be interpreted as a generalization of the max-min allocation problem. In the case of cardinality constraints, it

can also model problems from experimental design and spectral sparsification. We give an overview of prior work for these special cases.

Max-min allocation and Santa Claus: In the max-min allocation problem, we are given a set $[d]$ of agents and a set $[n]$ of items where agent $j \in [d]$ has valuation $h_{ij} \geq 0$ for item i. The goal is to select an assignment $\sigma : [n] \to [d]$ which maximizes

$$\min_{j \in [d]} \sum_{i:\sigma(i)=j} h_{ij}.$$

This can be seen as a special case of the minimum eigenvalue problem with partition constraints.

Bansal and Sviridenko [15] introduced the configuration LP as a relaxation for the max-min allocation problem but showed that it has an integrality gap of $\Omega(\sqrt{n})$ [15]. Asadpour and Saberi [16] gave a rounding scheme for the same LP, which achieves an $O(\sqrt{n}\log^3 n)$-approximation. This was later improved by Chakrabarty et al. [82] to an $\tilde{O}(n^{\epsilon})$-approximation for any $\epsilon \in \Omega(\log\log n/\log n)$ by iteratively constructing new instances with smaller integrality gap.

A further special case is the Santa Claus problem where each item i has an intrinsic value $H_i \geq 0$ such that $h_{ij} \in \{0, H_i\}$ for all players $j \in [d]$. Here, Bansal and Sviridenko [15] used the configuration LP to find an $O(\log\log n/\log\log\log n)$-approximation. Feige [17] non-constructively showed a constant upper bound on the integrality gap of the configuration LP for the Santa Claus problem by iteratively applying the Lovász Local Lemma. The current best bound is due to Haxell and Szabó [18], who used new topological techniques to show that the integrality gap is at most 3.534. Bounds on the integrality gap do not immediately lead to efficient approximation algorithms, but Davies et al. [19] recently gave an algorithm for a more general setting that can be used to achieve a $(4+\epsilon)$-approximation for the Santa Claus problem.

Experimental Design (E-optimal Design): Even with cardinality constraints (uniform matroid of rank k), the minimum eigenvalue problem is NP-hard [84]. Allen-Zhu et al. [13] showed that it is possible to deterministically find a $(1-\epsilon)$-approximation so long as $k \geq \Omega(d/\epsilon^2)$ by rounding the natural convex relaxation. They also conjectured that this requirement was necessary. This conjecture was confirmed in [1], where they showed an integrality gap instance for the convex relaxation. Recently Lau and Zhou [2] have built on the regret minimization framework from [13] to show that a modified local search algorithm with a "smoothed" objective works as long as there is a near-optimal solution with a good condition number.

Since our algorithm also gives approximations for other objectives we can also make comparisons to other experimental design problems. In D-design the objective is select a collection of k vectors to maximize $f : X \mapsto \det(X)$ and in A-design the goal is to minimize $f : X \mapsto \mathrm{trace}(X^{-1})$. In both cases there are combinatorial $(1+\epsilon)$-approximations when $k \geq \Omega(d/\epsilon)$ [14, 2]. This requirement on k is weaker than if prior results for E-design are extended directly. The approximations are much worse for k close to d. For example a d-approximation for A-design when $k = d$ [1].

Spectral Sparsification and Kadison-Singer. The problem of rounding the natural convex programming relaxation for the minimum eigenvalue problem is closely related to spectral sparsification [85] and the Kadison-Singer problem [80]. In spectral sparsification [85], the goal is to pick a small subset of vectors $S \subseteq [n]$ such that $\sum_{i \in S} w_i v_i v_i^\top$ spectrally approximates $\sum_{i \in [n]} v_i v_i^\top$ for some weights w_i. In the cardinality constrained minimum eigenvalue problem, rounding the convex programming solution involves finding a small set S, such that $\sum_{i \in S} v_i v_i^\top$ spectrally approximates $\sum_{i \in [n]} x_i v_i v_i^\top$, where the weights x_i form the solution to the convex relaxation. Indeed [13] essentially build on this connection to obtain their results for the E-design problem discussed earlier. The Kadison-Singer problem [80] is closely related to the minimum eigenvalue problem under a partition ma-

troid constraint. We utilize this connection in Corollary 2 to give an algorithmic version of the Kadison-Singer problem for constant dimensions. More generally, the Kadison-Singer problem can be reformulated as showing that the integrality gap of the natural relaxation of the minimum eigenvalue problem under partition matroid constraints is at most $1/(1-\epsilon)$ if the length of each vector is at most $O(\epsilon)$.

5.3 The Algorithm for Partition Matroids

To highlight the main idea of our algorithm, we first prove Theorem 5 for the special case of partition matroid. Let $\mathcal{M} = (E, \mathcal{I})$ be a partition matroid where $E = P_1 \cup \cdots \cup P_k$ be a disjoint union of parts with each part containing n elements, and we have a collection of vectors v_{ij} for $i \in [k]$ and $j \in P_i$. The goal is to select an element $\sigma(i) \in P_i$ for each i to maximize $\lambda_{\min}\left(\sum_{i=1}^{k} v_{i\sigma(i)} v_{i\sigma(i)}^{\top}\right)$.

We can construct the natural convex relaxation of this problem as follows. For each $i \in [k]$ and $j \in P_i$, we add a decision variable x_{ij} which represents whether we select the vector v_j from part P_i, i.e., if $\sigma(i) = j$. Then we get the convex program

$$\begin{aligned} \max \quad & \lambda_{\min}(X) \\ & X = \sum_{i=1}^{k} \sum_{j \in P_i} x_{ij} \cdot v_{ij} v_{ij}^{\top} \\ & \sum_{j \in P_i} x_{ij} = 1, \quad \forall i \in [k] \\ & x \geq 0 \end{aligned}$$

The constraint $\sum_{j \in P_i} x_{ij} = 1$ ensures that we have a probability distribution over the possible assignments within each part in the optimal solution.

Given an optimal solution $x^{\star}$ with value OPT, a natural rounding strategy is to round independently within each part. Following this rounding strategy, we get a rank 1 random

matrix $\mathbf{M}_i$ for each part P_i with

$$\mathbf{P}(\mathbf{M}_i = v_{ij}v_{ij}^\top) = x^\star_{ij}\,, \quad \forall j \in P_i\,.$$

The following matrix concentration inequality bounds the probability of failure of this rounding strategy.

Theorem 28 *[58, Theorem 5.1.1] Consider independent random matrices* $\mathbf{M}_1, \ldots, \mathbf{M}_k \in \mathbb{S}_d^+$. *Set*

$$\mu_{\min} = \lambda_{\min}\left(\mathbf{E}\left[\sum_{i=1}^k \mathbf{M}_i\right]\right)\,.$$

If $\lambda_{\max}(\mathbf{M}_i) \leq R$ *for all* $i \in [k]$ *then*

$$\mathbf{P}\left(\lambda_{\min}\left(\sum_{i=1}^k \mathbf{M}_i\right) < (1-\epsilon)\mu_{\min}\right) \leq d \cdot \exp\left(\frac{-\epsilon^2\mu_{min}}{2R}\right).$$

If we round according to the optimal solution $x^\star$ then

$$\mathbf{E}\left[\sum_{i=1}^k \mathbf{M}_i\right] = \sum_{i=1}^k \sum_{j\in P_i} x^\star_{ij} v_{ij} v_{ij}^\top.$$

So $\mu_{\min} = OPT$, and since for our particular case M_i are rank 1,

$$R = \max_i \lambda_{\max}(M_i) = \max_{ij} \|v_{ij}\|^2.$$

To bound the failure probability, we want $R \approx \epsilon^2/\log(d)$, which in turn requires that $\max_{ij}\|v_{ij}\|^2 = O(\epsilon^2/\log(d))$. This is a very strong assumption on an instance.

The plan is to "guess" a suitable change of basis such that all the vectors in the support of our optimal solution have a small norm. This will be useful because of the following standard, but slightly more flexible, version of the preceding matrix concentration inequality.

Corollary 3 *Consider independent random matrices $\mathbf{M}_1, \ldots, \mathbf{M}_k \in \mathbb{S}_d^+$ and let A be an arbitrary positive definite matrix. Define $\mu_{\min} := \lambda_{\min}\left(A^{-1/2}\,\mathbf{E}\left[\sum_{i=1}^k \mathbf{M}_i\right] A^{-1/2}\right)$. If $\lambda_{\max}(A^{-1/2}\mathbf{M}_i A^{-1/2}) \leq R$ for all $i \in [k]$ then*

$$\mathbf{P}\left(\sum_{i=1}^k \mathbf{M}_i \not\succeq (1-\epsilon)\mu_{\min} \cdot A\right) \leq d \cdot \exp\left(\frac{-\epsilon^2 \mu_{min}}{2R}\right).$$

Proof This is a simple calculation, using the fact the the semidefinite order is preserved under conjugation.

$$\begin{aligned}
\mathbf{P}\left(\sum_{i=1}^k \mathbf{M}_i \not\succeq (1-\epsilon)\mu_{\min} \cdot A\right) &= \mathbf{P}\left(\sum_{i=1}^k A^{-1/2}\mathbf{M}_i A^{-1/2} \not\succeq (1-\epsilon)\mu_{\min} \cdot I\right) \\
&= \mathbf{P}\left(\lambda_{\min}\left(\sum_{i=1}^k A^{-1/2}\mathbf{M}_i A^{-1/2}\right) < (1-\epsilon)\mu_{\min}\right) \\
&\leq d \cdot \exp\left(\frac{-\epsilon^2 \mu_{min}}{2R}\right).
\end{aligned}$$

□

Again, since $\mathbf{M}_i$ is rank 1 for our case, we have $R = \max_{i\in[k]}\ \lambda_{\max}(A^{-1/2}\mathbf{M}_i A^{-1/2}) = \max_{i,j}\ \ v_{ij}^\top A^{-1} v_{ij}$. So, to use this corollary, we first need to find a matrix A such that $v_{ij}^\top A^{-1} v_{ij} = O(\epsilon^2/\log(d))$ for all $[i] \in [k], j \in P_i$. We will only need to consider matrices A of a specific form that uses the input vectors.

Given a subset $S \subseteq E$, we define $A_S := \sum_{(i,j)\in S} v_{ij} v_{ij}^\top$, and consider the set of long vectors in the norm induced by A_S: $L(S) := \left\{(i,j) \in E\backslash S : v_{ij}^\top A_S^{-1} v_{ij} > \frac{\epsilon^2}{10\log(d)}\right\}$. For a fixed set S, the following convex program ensures that S is included in the solution and no

"long" vectors from $L(S)$ are included in the solution.

$$\begin{aligned} \max \quad & \lambda_{\min}(X) \\ & X = \sum_{i=1}^{k} \sum_{j \in P_i} x_{ij} \cdot v_{ij} v_{ij}^\top \\ & \sum_{j \in P_i} x_{ij} = 1, \ \forall i \in [k] \\ & x_{ij} = 0, \ \forall (i,j) \in L(S) \\ & x_{ij} = 1, \ \forall (i,j) \in S \\ & x \geq 0 \end{aligned} \tag{CP(S)}$$

Because of the extra constraints excluding "long" vectors, we could now use the flexible matrix concentration inequalities to randomly round the optimal solution.

But, it is not clear that there is a good choice of S for which the convex program CP(S) is still a relaxation of the original problem. Lemma 29, which we restate here for the reader's convenience, shows that there exists a suitable set S that is not too large.

Lemma 29 *For any set T and a set of vectors $\{v_i : i \in T\}$ in $\mathbb{R}^d$ such that $\sum_{i \in T} v_i v_i^\top$ is invertible, there exists a subset $S \subseteq T$ such that $|S| = O(d \log(d)/\epsilon^2)$, $A_S = \sum_{i \in S} v_i v_i^\top$ is invertible, and for all $i \in T \backslash S$,*

$$v_i^\top A_S^{-1} v_i \leq \frac{\epsilon^2}{10 \log(d)}.$$

The proof of this lemma is inspired by the local search algorithm of [14]. It may not be apparent why a subset satisfying the conditions of Lemma 29 should exist. However, in the proof, we show that any subset of T that is locally optimal with respect to a local search criteria for *determinant maximization* satisfies the guarantees of Lemma 29.

Proof (of Lemma 29) We consider the local search process of [14]. Starting with a set S of size ℓ such that $A = \sum_{i \in S} v_i v_i^\top$ is invertible, we apply the following update rule. For any $j \in T \backslash S$ and $i \in S$, if $\det(A) < \det(A - v_i v_i^\top + v_j v_j^\top)$, update $S = \{S \backslash \{i\}\} \cup \{j\}$

and iterate.

Let $S \subseteq T$ be a locally optimal (under single element swaps) solution for this process (such an S corresponds to the locally optimal solution determinant maximization problem subject to the cardinality constraint $|S| \leq \ell$), and let $A = \sum_{i \in S} v_i v_i^\top$. More concretely, this means that for all $i \in S$ and $j \in T \backslash S$,

$$\det(A) \geq \det(A - v_i v_i^\top + v_j v_j^\top)\,.$$

We calculate the determinant on the right-hand side using the matrix determinant lemma,

$$\begin{aligned}
&\det(A - v_i v_i^\top + v_j v_j^\top) \\
&\qquad = \det\left(A + \begin{bmatrix} v_i & v_j \end{bmatrix} \begin{bmatrix} -v_i & v_j \end{bmatrix}^\top\right) \\
&\qquad = \det(A) \cdot \det\left(I_2 + \begin{bmatrix} -v_i & v_j \end{bmatrix}^\top A^{-1} \begin{bmatrix} v_i & v_j \end{bmatrix}\right) \\
&\qquad = \det(A) \cdot \left((1 - v_i^\top A^{-1} v_i)(1 + v_j^\top A^{-1} v_j) + (v_i^\top A^{-1} v_j)^2\right)\,.
\end{aligned}$$

So local optimality implies that for every $i \in S$ and $j \notin S$, $(1 - v_i^\top A^{-1} v_i)(1 + v_j^\top A^{-1} v_j) + (v_i^\top A^{-1} v_j)^2 \leq 1$. Rearranging this inequality we get

$$v_j^\top A^{-1} v_j - (v_i^\top A^{-1} v_i) \cdot (v_j^\top A^{-1} v_j) + (v_i^\top A^{-1} v_j)^2 \leq v_i^\top A^{-1} v_i\,. \tag{5.1}$$

Note that

$$\begin{aligned}
\sum_{i \in S} v_i^\top A^{-1} v_i &= \langle A, A^{-1} \rangle = d, \quad \text{and} \\
\sum_{i \in S} (v_i^\top A^{-1} v_j)^2 &= v_j^\top A^{-1} \left(\sum_{i \in S} v_i v_i^\top\right) A^{-1} v_j = v_j^\top A^{-1} v_j.
\end{aligned}$$

So for a fixed $j \in T \backslash S$, summing equation (5.1) over all $i \in S$ implies $\ell \cdot v_j^\top A^{-1} v_j - d \cdot$

$v_j^\top A^{-1} v_j + v_j^\top A^{-1} v_j \leq d$. Rearranging, we see that for any $j \in T \backslash S$,

$$v_j^\top A^{-1} v_j \leq \frac{d}{\ell - d + 1} = \frac{\epsilon^2}{10 \log(d)},$$

where the last equality follows by choosing $\ell = 10d \log(d)/\epsilon^2 + d - 1$. □

We will apply this lemma to the case when $T = \{v_{i\sigma^\star(i)} : i \in [k]\}$ where σ^* is the choice function that maximizes the minimum eigenvalue, i.e., when T contains the vectors from an optimal integral assignment. In particular, we get the following corollary.

Lemma 30 *There is a subset $S \subseteq E$ such that $|S| = O(d \log(d)/\epsilon^2)$, and the convex program CP(S) is a relaxation for the minimum eigenvalue problem.*

As d is a constant, the size of the set S we search for is also constant. Thus, there are at most $O(n^{O(d \log(d)/\epsilon^2)})$ possible choices for S. We will consider each choice in turn to guess the correct set. Note that trying every set of the appropriate size will be the dominant factor in determining the algorithm's runtime.

The following lemma proves that for any fixed subset S, rounding the optimal solution to CP(S) gives a good approximation to the optimal value of CP(S).

Lemma 31 *Let $S \subseteq E$ be an independent set, and let $x^\star$ be the optimal solution to CP(S). Then rounding randomly in each part outputs an assignment $\sigma : [k] \to E$ with $\sigma(i) \in P_i$ such that*

$$\mathbf{P}\left[\lambda_{\min}\left(\sum_{i=1}^{k} v_{i\sigma(i)} v_{i\sigma(i)}^\top\right) < (1-\epsilon) \cdot \lambda_{\min}\left(\sum_{i=1}^{k} \sum_{j \in P_i} x_{ij}^\star \, v_{ij} v_{ij}^\top\right)\right] < d^{-4}.$$

Proof Let $X = \sum_{i=1}^{k} \sum_{j \in P_i} x_{ij}^\star \, v_{ij} v_{ij}^\top$. The matrix $X^\star$ contains $v_{ij} v_{ij}^\top$ with coefficient 1 for every $(i,j) \in S$. Thus $X \succeq \sum_{i \in S} v_i v_i^\top$, so $v_j^\top X^{-1} v_j \leq \frac{\epsilon^2}{10 \log(d)}$ for all $j \notin L \cup S$. For the purposes of the analysis, for every $j \in S$ we can replace the vector v_i with $(v_i^\top X^{-1} v_i) \cdot \sqrt{10 \log(d)/\epsilon^2}$ copies of the same vector scaled down to have squared-length

at most $\epsilon^2/(10\log(d))$ with respect to X. Since all elements of S get value 1 in $x^\star$, we can similarly extend the vector $x^\star$ so that it has a 1 in all the copied entries. Since these values of x^* are deterministic, nothing changes about the resulting distribution over matrices, but we can now assume that $v_{ij}^\top X^{-1} v_{ij} < \epsilon^2/(10\log(d))$ for all (i,j) in the support of $x^\star$.

Next, define random matrices $\mathbf{M}_1, \ldots, \mathbf{M}_k$ such that for any $i \in [k]$, $\mathbf{P}\left(M_i = v_{ij} v_{ij}^\top\right) = x_{ij}^\star$ for all $j \in P_i$. We then apply Corollary 3 with $A = X$ on random matrices $\mathbf{M}_1, \ldots, \mathbf{M}_k$. Since $x^\star$ is not supported on L,

$$R = \max_i \ \lambda_{\max}(X^{-1/2}\mathbf{M}_i X^{-1/2}) \leq \frac{\epsilon^2}{10\log(d)}\,.$$

In addition, as $\mathbf{E}\left(\sum_i M_i\right) = X$ by definition, we have $\mu_{\min} = 1$.

Thus, if $\sigma : [k] \to E$ is the choice function obtained by independent rounding,

$$\mathbf{P}_\sigma\left[\sum_{i=1}^{k} v_{i\sigma(i)} v_{i\sigma(i)}^\top \not\succeq (1-\epsilon)X\right] \leq d \cdot \exp(-5\log(d)) = d^{-4}.$$

We conclude that $\lambda_{\min}\left(\sum_{i=1}^k v_{i\sigma(i)} v_{i\sigma(i)}^\top\right) \geq (1-\epsilon)\lambda_{\min}(X)$ with probability at least $1 - d^{-4}$. □

Combining this lemma with the earlier guarantee that there exists a set S of reasonable size such that CP(S) is a relaxation, we get the following algorithm: try all possible choices for the set S and return the solution with the best objective.

Algorithm 8 Algorithm to find an approximation to OPT

1: **Input**: Partition matroid $\mathcal{M}$ with k parts $P_1, \ldots, P_k$

2: **for** each $S \subseteq [n]$ such that $|S| = 10d\log(d)/\epsilon^2 + d - 1$ **do**

3: $x^* \leftarrow$ optimal solution of CP(S) for matroid $\mathcal{M}$

4: For each $i \in [k]$, assign $\sigma_S(i) = j$ with probability x^*_{ij}

5: **end for**

6: Return the choice function σ_S which maximizes $\lambda_{\min}\left(\sum_i v_{i\sigma_S(i)} v_{i\sigma_S(i)}^\top\right)$ over all choices of S

Proof (of Theorem 5 for Partition Matroids) By Lemma 30 there is a set $S \subseteq E$ with $|S| = O(d\log d/\epsilon^2)$ such that CP(S) is a relaxation.

Let x^* be the optimal value of CP(S) and let $\sigma^\star$ be the choice function of the optimal basis for the minimum eigenvalue problem. Since CP(S) is a relaxation, we have $\lambda_{\min}\left(\sum_{i=1}^k v_{i\sigma^*(i)} v_{i\sigma^*(i)}^\top\right) \leq \lambda_{\min}\left(\sum_{ij} x^*_{ij} v_{ij} v_{ij}^\top\right)$.

Lemma 31 implies that with high probability, the choice function obtained by rounding x^*, σ_S, is a good approximation to CP(S). So combining Lemma 31 with the previous inequality gives

$$\begin{aligned}\lambda_{\min}\left(\sum_{i=1}^k v_{i\sigma_S(i)} v_{i\sigma_S(i)}^\top\right) &\geq (1-\epsilon)\cdot\lambda_{\min}\left(\sum_{ij} x^*_{ij} v_{ij} v_{ij}^\top\right) \\ &\geq (1-\epsilon)\cdot\lambda_{\min}\left(\sum_{i=1}^k v_{i\sigma^*(i)} v_{i\sigma^*(i)}^\top\right),\end{aligned}$$

with probability at least $1 - d^{-4}$. Since we iterate over all choice functions in step 6 of Algorithm 8, we will output a choice function σ which is at least as good as σ_S with the same probability. □

5.3.1 Application: Algorithmic Kadison-Singer Problem

The Kadison-Singer conjecture was resolved in [80] using the following theorem which can be interpreted as a generalization of Weaver's conjecture [83].

Theorem 29 *[80, Corollary 1.5 with $r = 2$] Let $u_1, \ldots, u_m \in \mathbf{R}^d$ be vectors such that $\sum_{i=1}^m u_i u_i^\top = I$ and $\|u_i\|^2 \leq \alpha$ for all i. There exists a set $T \subseteq [m]$ such that*

$$\left(\frac{1}{2} - 3\sqrt{\alpha}\right) I_d \preceq \sum_{i \in T} u_i u_i^\top \preceq \left(\frac{1}{2} + 3\sqrt{\alpha}\right) I_d.$$

Their proof is based on analyzing interlacing families of polynomials and does not lead to an efficient algorithm to find such a subset T.

In [45], they introduce an algorithmic form of the Kadison-Singer problem, which asks to find such a subset assuming it exists. For a constant $c > 0$ and a set of vectors $u_1, \ldots, u_m \in \mathbf{R}^d$ such that $\|u_i\|^2 \leq \alpha$, $\sum_{i=1}^m u_i u_i^\top = I$ where there exists a subset $T \subseteq [m]$ satisfying

$$\left(\frac{1}{2} - c\sqrt{\alpha}\right) I \preceq \sum_{i \in T} u_i u_i^\top \preceq \left(\frac{1}{2} + c\sqrt{\alpha}\right), \tag{5.2}$$

the goal is actually to find a set $T \subseteq [m]$ which satisfies the above condition. This problem is FNP-hard when $c = 1/(4\sqrt{2})$ for general values of d [45, Theorem 2].

Their main result [45, Theorem 1] is an algorithm with running time

$$O\left(\binom{m}{k} \cdot \text{poly}(m, d)\right) \text{ for } k = O\left(\frac{d}{\epsilon^2} \log(d) \log\left(\frac{1}{c\sqrt{\alpha}}\right)\right),$$

which returns a set $T' \subseteq [m]$ such that

$$(1 - \epsilon)\left(\frac{1}{2} - c\sqrt{\alpha}\right) I \preceq \sum_{i \in T'} u_i u_i^\top \preceq (1 + \epsilon)\left(\frac{1}{2} + c\sqrt{\alpha}\right) I, \tag{5.3}$$

In this section, we will show how to use the rounding technique for partition matroids to give a simpler algorithm that achieves the same guarantee with the same run time, except

we save the small dependence on $\log(1/c\sqrt{\alpha})$ in the exponent.

The main idea of the proof is not new; the central construction is essentially the same as the one used in [80] to prove Corollary 1.5 and deduce Weaver's conjecture from their main result. We make a few aesthetic modifications to more closely match our prior setup.

Proof (of Corollary 2) Given vectors $u_1, \ldots, u_m \in \mathbf{R}^d$, we construct an instance of the minimum eigenvalue with partition constraints as follows. Let $E = \{1, 2\} \times [m]$, with m parts $P_1, \ldots, P_m$ so that $P_i = \{(i, 1), (i, 2)\}$ for $i \in [m]$. For each $i \in [m]$ define the vectors

$$v_{i1} = \begin{bmatrix} u_i \\ 0 \end{bmatrix} \in \mathbf{R}^{2d}, \text{ and } v_{i2} = \begin{bmatrix} 0 \\ u_i \end{bmatrix} \in \mathbf{R}^{2d}.$$

To see how v and u are related, note that for any $\delta \in [0, 1/2)$ there is a choice function $\sigma : [m] \to \{1, 2\}$ such that

$$\left(\frac{1}{2} - \delta\right) I_{2d} \preceq \sum_{i=1}^{m} v_{i\sigma(i)} v_{i\sigma(i)}^\top \tag{5.4}$$

if and only if there is a set $T \subseteq [m]$ such that

$$\left(\frac{1}{2} - \delta\right) I \preceq \sum_{i \in T} u_i u_i^\top \preceq \left(\frac{1}{2} + \delta\right) I. \tag{5.5}$$

Given σ satisfying (5.4), let $X_1 := \sum_{i:\sigma(i)=1} u_i u_i^\top$ and $X_2 := \sum_{i:\sigma(i)=2} u_i u_i^\top$. Then X_1 and X_2 are respectively the first and second diagonal $d \times d$ block of $\sum_{i=1}^m v_{i\sigma(i)} v_{i\sigma(i)}^\top$. Therefore $\left(\frac{1}{2} - \delta\right) I_{2d} \preceq \sum_{i=1}^m v_{i\sigma(i)} v_{i\sigma(i)}^\top$ if and only if $X_1 \succeq \left(\frac{1}{2} - \delta\right) I_d$ and $X_2 \succeq \left(\frac{1}{2} - \delta\right) I_d$. In addition, since $X_1 + X_2 = I_d$, this is equivalent to

$$\left(\frac{1}{2} - \delta\right) I_d \preceq \sum_{i:\sigma(i)=1} u_i u_i = I_d - X_2 \preceq \left(\frac{1}{2} + \delta\right) I_d.$$

We then use Algorithm 8 to find a $(1 - \epsilon)$ approximate solution $\sigma : [m] \to \{1, 2\}$ to input $\mathcal{M}$ and vectors v_{ij}. Since we assume there is a set T satisfying (5.2), Theorem 5

implies that with probability at least $1 - O(d^{-4})$, Algorithm 8 will return a choice function σ^* such that $(1-\epsilon)\left(\frac{1}{2} - c\sqrt{\alpha}\right) I_{2d} \preceq \sum_{i=1}^{m} v_{i\sigma^*(i)} v_{i\sigma^*(i)}^\top$, and we will return the set $T' = \{i \in [m] : \sigma^*(i) = 1\}$.

From the equivalence between (5.4) and (5.5), the set $T' = \{i \in [m] : \sigma(i) = 1\}$ satisfies (5.3)

$$(1-\epsilon)\left(\frac{1}{2} - c\sqrt{\alpha}\right) I_d \preceq \sum_{i \in T'} u_i u_i \preceq (1+\epsilon)\left(\frac{1}{2} + c\sqrt{\alpha}\right) I_d \,.$$

□

The resolution of the Kadison-Singer problem in [80] using the interlacing families of polynomials implies the following existential result about maximizing the minimum eigenvalue under partition matroid constraints.

Theorem 30 *[80, Theorem 1.4] For $\epsilon > 0$ and vectors $\{v_{ij}\}_{i\in[k], j\in[n]} \in \mathbf{R}^d$ with $\|v_{ij}\|^2 \leq \epsilon$ for all $i \in [k], j \in [n]$, if there exist $x_{ij} \geq 0$ such that*

$$\sum_{i=1}^{k}\sum_{j=1}^{n} x_{ij} \cdot v_{ij} v_{ij}^\top = I_d \quad \textit{and} \quad \sum_{j=1}^{n} x_{ij} = 1 \textit{ for all } i \in [k],$$

then there exists a choice function $\sigma : [k] \to [n]$ such that

$$(1-\sqrt{\epsilon})^2 \cdot I_d \preceq \sum_{i=1}^{k} v_{i\sigma(i)} v_{i\sigma(i)}^\top \preceq (1+\sqrt{\epsilon})^2 \cdot I_d.$$

We can state this result equivalently as an "existential" rounding result. When $\|v_{ij}\|^2 \leq \epsilon$, Theorem 30 implies that the integrality gap of the natural convex relaxation (CP) for the minimum eigenvalue problem with partition constraints is only $1/(1-\sqrt{\epsilon})^2$. It is an open problem to efficiently round the solution to the convex relaxation with comparable guarantees for any dimension d.

More generally, the problem of designing an approximation algorithm for the minimum

eigenvalue problem under partition or matroid constraints in arbitrary dimensions remains wide open. However, checking whether there is a solution with a non-zero objective can be solved in polynomial time solvable through matroid intersection. We have already discussed recent progress in the case of maximizing the determinant [62, 63, 22, 24, 64, 25], but it remains open whether those techniques can be utilized for the minimum eigenvalue problem.

5.4 General Matroid Constraints

In the general form of the problem, we are given a collection of vectors $v_1, \ldots, v_n \in \mathbf{R}^d$ and a matroid $\mathcal{M} = ([n], \mathcal{I})$, and the goal is to find a basis $B \in \mathcal{I}$ which maximizes $\lambda_{\min}\left(\sum_{i \in B} v_i v_i^\top\right)$. For background on matroids and matroid polytopes, refer back to Section 2.1 and Section 2.2.

For a general matroid, the idea of finding a linear transformation under which all elements in the optimal solution have a small norm generalizes easily. So we can use the same approach of first guessing a set $S \subseteq E$ on a reasonable size and then solving the convex relaxation of the problem conditioned on S being included in the solution.

Given a subset $S \subseteq [n]$, we can again set $A_S = \sum_{i \in S} v_i v_i^\top$, and consider the set of long vectors:

$$L(S) = \left\{ i \in [n] \backslash S : v_i^\top A_S^{-1} v_i > \frac{\epsilon^2}{10 \log(d)} \right\}.$$

For a matroid $\mathcal{M}$, let $\mathcal{P}(M) \subseteq [0,1]^n$ denote the matroid base polytope. Then the following

is the natural convex programming relaxation which excludes the "long" vectors.

$$\begin{aligned} \max \quad & \lambda_{\min}(X) \\ & X = \sum_{i=1}^{n} x_i \cdot v_{ij} v_{ij}^{\top} \\ & x \in \mathcal{P}(\mathcal{M}) \\ & x_i = 0, \ \forall i \in L(S) \\ & x_i = 1, \ \forall i \in S \\ & x \geq 0 \end{aligned} \tag{CP(S)}$$

This convex program can be solved in polynomial time (see Section 2.2). Just like in the partition case, Lemma 29 guarantees that there is a set S for which CP(S) is a relaxation for the minimum eigenvalue problem. As before, after solving CP(S), we can guarantee that all the vectors in the fractional support of the optimal solution will have a small norm with respect to A_S.

The challenge in extending the earlier approach to general matroid constraints comes from the rounding step. For a partition matroid, we could simply round the fractional optimum of CP(S) independently in each part to obtain a basis. However, for more general constraints, it is not so clear how to round a fractional solution to a basis.

Instead of rounding independently, we will use the technique of pipage rounding to find a basis. The following lemma is the lower-tail version of the same concentration inequality proved in [44]. The proof of this result can be found in the background material within Section 2.2.

Lemma 5 *Let $\mathcal{P}(\mathcal{M})$ be a matroid base polytope and $x \in \mathcal{P}(\mathcal{M})$. Let $M_1, \ldots, M_m$ be self-adjoint matrices that satisfy $\lambda_{\max}(M_i) \leq R$. Let $\mu = \lambda_{\min}\left(\sum_{i \in [n]} x_i M_i\right)$. If randomized pipage rounding (Algorithm 1) starts at x and outputs the extreme point $\hat{x} = \chi(B)$ of*

$\mathcal{P}(\mathcal{M})$, *then we have*

$$\mathbf{P}\left[\lambda_{\min}\left(\sum_{i\in B} M_i\right) \leq (1-\epsilon)\cdot\mu\right] \leq d\cdot\exp\left(\frac{-\epsilon^2\mu}{2R}\right).$$

We use this lemma to generalize our earlier approach to all matroids.

Lemma 32 *Let $S \subseteq E$ be an independent set in $\mathcal{M}$ and let $x^\star$ be the optimal solution to CP(S). Then pipage rounding starting at $x^\star$ outputs a basis B such that*

$$\mathbf{P}\left[\lambda_{\min}\left(\sum_{i\in B} v_i v_i^\top\right) < (1-\epsilon)\lambda_{\min}\left(\sum_{i\in[n]} x_i^\star\, v_i v_i^\top\right)\right] < d^{-4}.$$

The proof is identical to that of Lemma 31, except we use the matrix concentration inequality from Lemma 5.

Using this lemma, the following algorithm gives a $(1-\epsilon)$-approximation with high probability.

Algorithm 9 Algorithm to find an approximation to OPT

1: **for** each $S \subseteq [n]$ such that $|S| = 10d\log(d)/\epsilon^2 + d - 1$ **do**
2: Solve CP(S) to get optimal solution $x^\star$.
3: Let $B_S \leftarrow$ basis returned by Algorithm 1 for input $x^\star$
4: **end for**
5: Return the basis B_S with the best objective

5.4.1 Proof of Theorem 5 and Theorem 6

In this section we prove Theorem 6. The proof of Theorem 5 follows identically.

Proof (of Theorem 6) Let

$$B^* = \arg\max_{B\in\mathcal{I}} f\left(\sum_{i\in B} v_i v_i^\top\right) \quad \text{and let} \quad OPT = f\left(\sum_{i\in B^*} v_i v_i^\top\right).$$

Let $S \subseteq B^*$ such that $|S| = O(d \log d/\epsilon^2)$, $A = \sum_{(i,j)\in S} v_{ij} v_{ij}^\top$ is invertible, and for all $(i,j) \in B^* \backslash S$, $v_{ij}^\top A^{-1} v_{ij} \leq \epsilon^2/10d \log d$. By Lemma 29, such a set S exists.

Let x^* be the optimal solution and $X := \sum_{i\in E} x_i^* \cdot v_i v_i^\top$ of CP(S). Since the indicator vector of B^* satisfies the constraints of CP(S), $OPT = f\left(\sum_{i\in B^*} v_i v_i^\top\right) \leq f(X)$. Therefore, if B is the basis returned by pipage rounding, Lemma 5 implies that

$$\Pr\left[f\left(\sum_{i\in B} v_i v_i^\top\right) < (1-\epsilon)\cdot OPT\right] \leq \Pr\left[f\left(\sum_{i\in B} v_i v_i^\top\right) < (1-\epsilon)\cdot f(X)\right].$$

Since f is monotone and homogeneous, we have

$$\begin{aligned}
\Pr\left[f\left(\sum_{i\in B} v_i v_i^\top\right) < (1-\epsilon)\cdot f(X)\right] &= \Pr\left[f\left(\sum_{i\in B} v_i v_i^\top\right) < f((1-\epsilon)\cdot X)\right] \qquad (5.6)\\
&\leq \Pr\left[\sum_{i\in B} v_i v_i^\top \not\succeq (1-\epsilon)\cdot X\right]\\
&= \Pr\left[\sum_{i\in B} X^{-1/2} v_i v_i^\top X^{-1/2} \not\succeq (1-\epsilon)\cdot I\right]\\
&= \Pr\left[\lambda_{\min}\left(\sum_{i\in B} X^{-1/2} v_i v_i^\top X^{-1/2}\right) \leq (1-\epsilon)\right] \qquad (5.7)
\end{aligned}$$

Similar to the proof of Lemma 31, we will apply Lemma 5 to random matrices $M_i = v_i v_i^\top$ after appropriate transformations to ensure $R = O(\epsilon^2/\log d)$. First note that since $x_i^* = 1$ for any $i \in S$, we have $i \in B$ as pipage rounding does not change the integral elements of x^*. Therefore $\sum_{i\in S} v_i v_i^\top \preceq X$, and for any $i \in B\backslash S$,

$$\lambda_{\max}(X^{-1/2} v_i v_i^\top X^{-1/2}) = v_i^\top X^{-1} v_i \leq v_i^\top \left(\sum_{j\in S} v_j v_j^\top\right)^{-1} v_i \leq \frac{\epsilon^2}{10\log d}.$$

For any $i \in S$, $v_i X^{-1} v_i^\top \leq 1$, and since $x_i^* = 1$ we can replace $M_i = v_i v_i^\top$ with $r = 10\log d/\epsilon^2$ matrices $M_i^1, \ldots, M_i^r$ such that $M_i^j = \frac{\epsilon^2}{10\log d} v_i v_i^\top$. This ensures that $\lambda_{\max}(X^{-1/2} M_i^j X^{-1/2}) \leq \epsilon^2/10\log d$ for all $j \in [r]$. Applying Lemma 5 on random matri-

ces $\{M_i\}_{i\in B\setminus S}, \{M_i^j\}_{i\in S, j\in[r]}$ gives

$$\Pr\left[\lambda_{\min}(\sum_{i\in\tilde{B}} X^{-1/2} v_i v_i^\top X^{-1/2}) \leq (1-\epsilon)\right] \leq d^{-4}\,.$$

If B is the basis returned by Algorithm 9, then $f(B) \geq f(\tilde{B})$. Using the above inequality with (5.7) implies that $f(\sum_{i\in B} v_i v_i^\top) \geq (1-\epsilon)\cdot OPT$ with probability at least $1-d^{-4}$. □

5.5 Integrality Gap Example

Consider the vectors $v_1 = e_1, v_2 = e_1, v_3 = e_2, v_4 = e_3$ in $\mathbf{R}^3$ and a partition matroid $\mathcal{M} = ([4], \mathcal{I})$ defined by the bases $\{1,2,3\}, \{1,2,4\}$. The optimal value of maximizing the minimum eigenvalue for this instance is 0 as we are forced to pick v_1 and v_2 in any basis and they are linearly dependent.

The convex relaxation of maximizing the minimum eigenvalue for this instance is given by

$$\begin{aligned} \max \quad & \lambda_{\min}(X) \\ & X = x_1 \cdot v_1 v_1^\top + x_2 \cdot v_2 v_2^\top + x_3 \cdot v_3 v_3^\top + x_4 \cdot v_4 v_4^\top \\ & x_1 = 1, \quad \forall i \in [k] \\ & x_2 = 1, \quad \forall i \in [k] \\ & x_3 + x_4 = 1, \quad \forall i \in [k] \\ & x \geq 0 \end{aligned} \tag{CP}$$

The optimum of (CP) is attained when $x_1 = x_2 = 1$ and $x_3 = x_4$ which gives

$$X = 2e_1 e_1^\top + \frac{1}{2} e_2 e_2^\top + \frac{1}{2} e_3 e_3^\top\,.$$

So the optimal value of (CP) is $1/2$, whereas the true optimal is 0.

CHAPTER 6
NETWORK RECONFIGURATION

We will begin with the concrete problem of network reconfiguration, before showing it can be reduced to our general framework of subset selection with spectral objectives.

Power distribution networks are often built as highly connected networks of lines and busses, but are usually operated only along tree subgraphs (called radial networks) since this makes fault detection and overload protection easier [46]. In the network reconfiguration problem we are given a fixed graph $G = (V, E)$ where V is the set of busses (nodes) and E is the set of lines (edges). We will have a single substation $s \in V$ (source node) and all other nodes will be load busses (demand nodes) with given active power demands d_v for all $v \in V \backslash \{s\}$. Let f_e represent the power flow along line e. The goal is to find a spanning tree rooted at the substation s which minimizes the total energy loss. Following the work in [47] we make a few simplifying assumptions, but we will avoid adding the extra notation, and only explain these assumptions in plain English.

Assumption 1 *Voltage variation across the distribution network can be neglected.*

Assumption 2 *The impact of line losses on lines flows is negligible compared to the power demands at the busses.*

With these assumptions, we can model the reconfiguration problem in a much simpler mathematical way. Given a candidate solution spanning tree T, the power flow on each edge $e \in E$ will be equal to the total demand of the nodes in the subtree of descendant of that edge. If we denote by T_e the subtree not containing s in $T \backslash \{e\}$ then

$$f_e = \sum_{v \in T_e} d_v.$$

In terms of standard flow problems, the flow f is the unique flow on T which satisfies all the demands. We will also assume that every edge has unit resistance to simplify the presentation. In this case, the loss of edge e, denoted L_e, is exactly

$$L_e = f_e^2 = \left(\sum_{v \in T_e} d_v \right)^2 .$$

Given a spanning tree T we can take the sum over all edges in the tree, and write the loss minimization reconfiguration problem as follows.

Problem 2 *Let $G = (V, E)$ be a graph with a source node $s \in V$ and demands d_v at each $v \in V \backslash \{s\}$. Then the network reconfiguration problem is*

$$\min_T \sum_{e \in T} \left(\sum_{v \in T_e} d_v \right)^2 ,$$

where the minimum is over all spanning trees T.

This problem is called reconfiguration as opposed to just configuration, since the assumption is that we are starting with a network which has already been configured, and we want to open some switches and close others in order to balance load or minimize power loss. The above formulation was motivated by power loss reduction, but a simple change in the interpretation of the variables also allows it to model a load balancing objective for the power network. Thus, working with this more abstract mathematical formulation can give ways to solve multiple distinct reconfiguration problems for power networks.

The problem of network reconfiguration is NP-hard even in the case that all demands $d_v = 1$ for $v \in V \backslash \{s\}$, which can be shown via a reduction from the 3-partition problem [47]. Many different approaches have been proposed for the network reconfiguration problem since it was introduced in 1989 [49]. The first proposed algorithm was "branch exchange" which is essentially a local search algorithm on the space of spanning trees. Given an initial tree configuration T we consider the edges which are not in the tree. Adding

such an edge to the tree creates a unique cycle, and we delete one edge from this cycle so that we minimize the loss in the resulting tree. This process is repeated until there are no edges which we can update and decrease the objective, and the resulting tree configuration is output. When initially proposed in [49] this algorithm was accompanied by a table of test result which showed that it gives acceptable solutions, but no theoretical guarantees were provided.

In 2009 [86] a new methodology was proposed for the network reconfiguration problem based on a optimal power flow and Bender's decomposition approach. The decision (master) problem is a mixed integer program with a quadratic objective, and is used to determine the tree configuration. The second-stage problem is also non-linear and determines the feasibility of the master problem solution. They use an optimal power flow formulation for the second-stage problem and use it to provide the Bender cuts for the master problem. This method was tested on some practically-sized instances, and seems to converge after very few iterations and has a fast execution time, but again there are no theoretical guarantees.

A third approach [47] based on supermodular function minimization provides the first approximation guarantee on the "branch exchange" algorithm. Recall the objective for a tree T in the reconfiguration problem is

$$\sum_{e \in T} \left(\sum_{v \in T_e} d_v \right)^2 .$$

In [47] they extend this objective function to a function $g : 2^E \to \mathbb{R}$ such that $g(T)$ matches the objective on any tree, and such that $g(\cdot)$ is supermodular. Now the problem reduces to minimizing a supermodular function subject to a matroid base constraint. Unfortunately this problem is still NP-hard, even to approximate within any constant factor, and without the constraints. Instead they consider the function $A \mapsto g(E) - g(A)$ for $A \in 2^E$, which has been shifted and negated so that it is now a non-negative submodular function. Then it is

possible to use an "off-the-shelf" algorithm. It turns out that if you open up the algorithm it is essentially identical to the "branch exchange" proposed in [49]. This framework provides a multiplicative approximation factor depending on $g(E)$, which is a fairly weak guarantee, but was still the first theoretical bound.

Finally, in [48], they provide the first reasonable approximation guarantee for the reconfiguration problem. In particular, they show that any shortest path tree (with respect to the number of edges) is an n-approximate solution where $n = |V|$. They also give comparable lower bounds, which show that this argument cannot be improved; there are instances where a shortest path tree is always an $\Omega(n)$-approximate solution. They also propose new algorithms which apply to specialized instance, such as grid-like graphs, which do appear often in practical applications.

In this chapter, we begin by showing how the network reconfiguration problem can be modeled as a subset selection problem with a spectral objective. This naturally leads to a convex-programming relaxation. By replacing the matrix $dd^\top$ in the objective with other positive semidefinite matrices, we can also model the problem of finding a minimum routing cost tree [50, 51] or the problem of finding a low-stretch spanning tree [52]. This problem also generalizes A-design [1, 53], and we show that a straightforward rounding algorithm achieves an $O(n)$-approximation, matching the best known bound for the network reconfiguration problem [48].

Next, in Section 6.2, we investigate previous integer-nonlinear programming formulations for the problem. For two natural quadratic programs, we show that the fractional relaxation is actually equivalent to the simpler problem of finding the energy of the electrical flow in the full underlying graph.

In Section 6.3 we return to our spectral Laplacian-based formulation, and show that it is stronger than the previously-discussed quadratic programs.

Finally, in Section 6.4 we provide heuristic algorithms to compute upper and lower bounds on the network reconfiguration objective. The lower bounds are based on solving

the unweighted electrical flow problem, and solving the Laplacian-based formulation. The upper bounds come from combinatorial algorithms: greedy delete, which starts from the full graph and iteratively removes the edge that increases the objective the least; and local search, which starts at a spanning tree iteratively swaps edges in and out to find a better tree. We give a brief overview of the algorithms, and conclude with practical results and comparisons.

6.1 Formulation as a Spectral Objective

In this section we will show that the network reconfiguration problem can be reformulated to match our general subset selection framework. The final convex programming formulation can also model the case of general demands. Given a graph $G = (V, E)$ and a demand vector $d \in \mathbf{R}^V$ we would like to find a tree T and a flow $f \in \mathbf{R}^E$ supported only on T such that $\|f\|_2$ is minimized. For this to be a feasible problem we require that $\sum_{v \in V} d_v = 0$.

For every $e \in E$ let $v_e \in \mathbf{R}^V$ be a signed node-arc incidence vector, e.g. if $e = (i, j)$ then $v_e = \mathbb{1}_i - \mathbb{1}_j$ (the orientation will not matter in the final formulation). For a fixed spanning tree T let B_T be the $V \times E(T)$ matrix whose columns are the vectors v_e for $e \in E(T)$. Then the condition that f is a feasible flow using only edges from T can be written as $B_T f = d$. Finding a solution f with $\|f\|_2$ minimal now reduces to a least squares problem. Hence, the optimal solution will be when $f = B_T^+ d$ (B_T^+ is the pseudo-inverse of B_T).

Indeed, the column space of B_T is exactly $\{y \in \mathbf{R}^V : \mathbb{1}^\top y = 0\}$, since the columns are linearly independent, there are $n - 1$ of them, and they all lie in that $(n - 1)$-dimensional subspace. The demand vector p is also in the column space, so we can write $d = B_T x$ for some $x \in \mathbf{R}^{E(T)}$. Now $f = B_T^+ B_T x$, and we know that $B_T^+ B_T$ is the projection onto the rows pace of B_T. Thus any solution x to $B_T x = d$ can be written as $f + g$ where g is in the kernel of B_T. Since f and g would be orthogonal, this implies that $\|f + g\|_2 > \|f\|_2$ when $g \neq 0$, so we conclude that $f = B_T^+ d$ is the solution supported on $E(T)$ which minimizes

$\|f\|_2$.

Now that we have a linear-algebraic formula for f in terms of only our input data and the tree T, we can write the objective value this way as well. Once we have chosen a spanning tree T, the optimal value on that tree is

$$\|B_T^+ d\|_2 = (B_T^+ d)^\top B_T^+ d = d^\top (B_T^+)^\top B_T^+ d = d^\top (B_T B_T^\top)^+ d.$$

The matrix $B_T B_T^\top$ is the Laplacian of the tree T, and can also be expanded in terms of our node-arc incidence vectors v_e:

$$B_T B_T^\top = \sum_{e \in E(T)} v_e v_e^\top.$$

Hence, we have reduced the original flow problem to the problem of finding a basis T of the graphic matroid on G which minimizes

$$\left\langle dd^\top, \left(\sum_{e \in T} v_e v_e^\top \right)^+ \right\rangle.$$

While not directly a function of the eigenvalues of the Laplacian, if we replaced the matrix $dd^\top$ with the identity, the objective would be exactly $\Phi(\cdot) = \operatorname{Tr}((\cdot)^{-1})$. Hence, this formulation fits in the spectral-objective framework, using the graphic matroid of G as our constraint.

Moreover, using this as the objective, we get the following convex relaxation for the network reconfiguration problem:

$$\begin{aligned} \min \quad & \left\langle dd^\top, L^+ \right\rangle \\ & L = \textstyle\sum_{e \in E} x_e \cdot v_e v_e^\top \\ & x \in \mathrm{SPT}(G), \end{aligned} \qquad \text{(CP-Reconfiguration)}$$

where $\mathrm{SPT}(G)$ denotes the spanning tree polytope for the graph G.

It is clear that the feasible region is a polytope, so it is convex. It remains to check that the objective function is also convex, and we will also need access to its gradient if we want to optimize. It suffices to verify the following proposition.

Proposition 1 *If C is a symmetric positive semidefinite matrix, then the function $f(X) = \langle C, X^{-1} \rangle$ is convex for $X \succ 0$, and its directional derivative in the direction of a symmetric matrix Y is $-\langle X^{-1} C X^{-1}, Y \rangle$.*

Proof Starting at a positive definite matrix X we can calculate the directional derivatives of every order in the direction of a symmetric matrix Y. To do so, we need to understand how the function $\langle C, (X + tY)^{-1} \rangle$ behaves for t very close to zero. When t is close enough to zero, the inverse will actually exist, and we can write

$$(X + tY)^{-1} = X^{-1}(I + tYX^{-1})^{-1} = X^{-1} \sum_{n=0}^{\infty} (-1)^n t^n (YX^{-1})^n,$$

since this is a geometric series of matrices. Now we can plug this into the function f to see that

$$\begin{aligned} f(X + tY) &= \left\langle C, (X + tY)^{-1} \right\rangle \\ &= \sum_{n=0}^{\infty} (-1)^n t^n \left\langle C, X^{-1} (YX^{-1})^n \right\rangle. \end{aligned}$$

Hence, the first-order directional derivative of f in the direction Y is $-\langle C, X^{-1} Y X^{-1} \rangle$, and the second-order directional derivative is

$$\left\langle C, X^{-1} Y X^{-1} Y X^{-1} \right\rangle = \left\langle C, X^{-1/2} (X^{-1/2} Y X^{-1/2})^2 X^{-1/2} \right\rangle \geq 0,$$

since the matrices C and $X^{-1/2}(X^{-1/2} Y X^{-1/2})^2 X^{-1/2}$ are positive semi-definite. Since Y was an arbitrary symmetric matrix, this implies that f is convex at X. $\square$

6.1.1 A Pattern of Similar Objectives

In the previous section we showed that the network reconfiguration problem could be modeled as optimizing the function

$$\left\langle dd^\top, \left(\sum_{e\in T} v_e v_e^\top\right)^+ \right\rangle$$

subject to a spanning tree constraint.

To show that this leads to a convex programming formulation, we used the fact that the function $f(X) = \langle C, X^{-1}\rangle$ is convex for any positive semi-definite matrix C. So, it is natural to ask if there are any other choices for the matrix C that lead to interesting combinatorial optimization problems.

Minimum Routing Cost Spanning Trees Given an undirected graph with non-negative costs on the edges, the routing cost of a spanning tree is the sum over all the pairs of vertices in the graph of the distance between the pair in the tree. There is a classical 2-approximation for this problem [50] (consider the shortest path tree rooted at each vertex, and select the best such tree), and a more complicated PTAS [51].

If the edge costs are all 1, then this can be modeled in the above framework by selecting $C = n \cdot I$. Indeed, if $L_T = \sum_{e\in T} v_e v_e^\top$ then the distance between vertex i and vertex j in the tree T is exactly $(\mathbb{1}_i - \mathbb{1}_j)^\top L_T^+ (\mathbb{1}_i - \mathbb{1}_j)$ (we can handle general weights by a suitable scaling of the vectors v_e). This is because the unique solution to $B_t f = \mathbb{1}_i - \mathbb{1}_j$ will have ± 1 entries along the path from i to j in T, and 0 everywhere else. Thus, the total routing cost is

$$\left\langle \sum_{\text{pairs } \{i,j\}} (\mathbb{1}_i - \mathbb{1}_j)(\mathbb{1}_i - \mathbb{1}_j)^\top, L_T^+ \right\rangle = \left\langle nI - \mathbb{1}\mathbb{1}^\top, L_T^+ \right\rangle = \left\langle nI, L_T^+ \right\rangle,$$

where the last line follows from the fact that $\mathbb{1}$ is orthogonal to the column space of L_T.

Low-stretch Spanning Trees The following is a rephrasing of a result from [52]. Given a weighted graph with edge weights w_e, define the Laplacian of G to be $\tilde{L}_G = \sum_{e \in E} w_e \cdot v_e v_e^\top$. We consider a spanning tree T of G with the same edge weights. Since T is a tree, every pair of vertices in G is connected by a unique path in T. For an edge $e \in E$ let $e_1, \ldots, e_k$ be the edges along the unique path in T between the endpoints of e, and define the stretch of e to be

$$\mathrm{st}_T(e) = w(e) \left(\sum_{i=1}^{k} 1/w(e_i) \right).$$

The stretch of the graph is then defined to be

$$\mathrm{st}_T(G) = \sum_{e \in E} \mathrm{st}_T(e).$$

It is possible [87] to (very) efficiently construct spanning trees with total stretch

$$\mathrm{st}_T(G) \in O(m \log n \log\log n (\log\log\log n)^3).$$

In [52] they showed that

$$\mathrm{st}_T(G) = \left\langle \tilde{L}_G, \tilde{L}_T^+ \right\rangle,$$

where $L_T = \sum_{e \in E(T)} w_e \cdot v_e v_e^\top$. Hence, (if we allow weighted Laplacians) this also fits our framework with $C = \tilde{L}_G$. A similar connection was also made by [53]. They explained how the low-stretch spanning tree problem could be reformulated with $C = I$, by performing a change of basis on the vectors v_e.

The general problem We can write the more general convex program in the same way.

$$\begin{aligned} \min \ & \left\langle C, \left(\sum_{e \in E} x_e \cdot v_e v_e^\top \right)^{-1} \right\rangle \\ & x \in \mathrm{SPT}(G), \\ & x_e \geq 0, \text{ for all } e \in E. \end{aligned}$$

To capture the above problems in full generality we also need to allow the vectors v_e to be scaled by constants, but otherwise do not allow arbitrary linear transformations.

Using the above convex program, we can apply the following algorithm which is based on the proportional volume sampling proposed in [1]. First, we solve the convex relaxation to get an (approximate) optimal solution $x^\star \in \mathbf{R}^E$ and matrix $X = \left(\sum_{e\in E} x_e^\star \cdot v_e v_e^\top\right)$. We use this optimal solution to create a probability distribution over the feasible solutions, which we can then sample. In particular, we sample set $S \subseteq E$ with probability proportional to $\left(\prod_{e\in S} x_e^\star\right) \det\left(\sum_{e\in E} v_e v_e^\top\right)$. With this probability, we can directly bound the expected objective value of our sampled solution.

Proposition 2 *If $x^\star$ is an optimal solution to the convex program and we sample S with probability proportional to $\left(\prod_{e\in S} x_e^\star\right) \det\left(\sum_{e\in E} v_e v_e^\top\right)$ then*

$$\mathbf{E}_S\left[\left\langle C, \left(\sum_{e\in S} v_e v_e^\top\right)^{-1}\right\rangle\right] \leq d \cdot \left\langle C, X^{-1}\right\rangle,$$

where $X = \left(\sum_{e\in E} x_e^\star \cdot v_e v_e^\top\right)$.

Proof This is a direct, although long, calculation. We use $V_S V_S^\top$ to denote the matrix $\sum_{e\in S} v_e v_e^\top$, and use $E_{d-1}(M)$ to denote the sum of all $(d-1)\times(d-1)$ minors of a $d \times d$ matrix M. We will require a slight generalization of the Cauchy-Binet inequality 18, which we can apply to the function E_{d-1}; given a matrix B,

$$E_{d-1}(BB^\top) = \sum_{S:|S|=d-1} \det(B_S B_S^\top),$$

where B_S is the submatrix of B comprised of the columns in S. We will also use the notation $V_S(x)$ to denote the matrix with columns $\sqrt{x_i}\cdot v_i$ for all $i \in S$. The reason for this notation is so we can easily write down the simple equation

$$\left(\prod_{i\in S} x_i\right) \det(V_S V_S^\top) = \det(V_S(x) V_S(x)^\top),$$

We first prove the result in the case that $C \succ 0$, and then can deduce the positive semidefinite case by a limiting argument. First, if we simply expand the expectation

$$\begin{aligned}\mathbf{E}_S\left[\left\langle C, (V_S V_S^\top)^{-1}\right\rangle\right] &= \sum_{|S|=d} \left\langle C, (V_S V_S^\top)^{-1}\right\rangle \cdot \frac{\prod_{i\in S} x_i \det(V_S V_S^\top)}{\sum_{|S|=d}\prod_{i\in S} x_i \det(V_S V_S^\top)} \\ &= \frac{\sum_{|S|=d} \mathrm{Tr}\left(\left(C^{-1/2} V_S V_S^\top C^{-1/2}\right)^{-1}\right)\prod_{i\in S} x_i \det(V_S V_S^\top)}{\det(X)},\end{aligned}$$

where we have used Cauchy-Binet to collect the denominator into a single term. Now, for a $d\times$ matrix M with eigenvalues $\lambda_1, \ldots \lambda_d > 0$ we can write

$$\mathrm{Tr}(M^{-1}) = \sum_{i=1}^d \frac{1}{\lambda_i} = \frac{\sum_{S:|S|=d-1}\prod_{i\in S}\lambda_i}{\prod_{i=1}^d \lambda_i} = \frac{E_{d-1}(M)}{\det(M)}.$$

Using this fact, we can expand the trace:

$$\begin{aligned}\mathbf{E}_S\left[\left\langle C, (V_S V_S^\top)^{-1}\right\rangle\right] &= \frac{\sum_{|S|=d} \mathrm{Tr}\left(\left(C^{-1/2} V_S V_S^\top C^{-1/2}\right)^{-1}\right)\prod_{i\in S} x_i \det(V_S V_S^\top)}{\det(X)} \\ &= \frac{\sum_{|S|=d} \frac{E_{d-1}\left(C^{-1/2} V_S V_S^\top C^{-1/2}\right)}{\det\left(C^{-1/2} V_S V_S^\top C^{-1/2}\right)}\prod_{i\in S} x_i \det(V_S V_S^\top)}{\det(X)} \\ &= \frac{\det(C)}{\det(X)}\sum_{|S|=d} E_{d-1}\left(C^{-1/2} V_S V_S^\top C^{-1/2}\right)\prod_{i\in S} x_i\end{aligned}$$

where we have used the fact that $\det\left(C^{-1/2} V_S V_S^\top C^{-1/2}\right) = \det(C)^{-1}\det(V_S V_S^\top)$ to simplify, and then rearranged to clean up. Now we use the general version of Cauchy-Binet for E_{d-1} and swap the order of summation.

$$\begin{aligned}\mathbf{E}_S\left[\left\langle C, (V_S V_S^\top)^{-1}\right\rangle\right] &= \frac{\det(C)}{\det(X)}\sum_{|S|=d} E_{d-1}\left(C^{-1/2} V_S V_S^\top C^{-1/2}\right)\prod_{i\in S} x_i \\ &= \frac{\det(C)}{\det(X)}\sum_{|S|=d}\prod_{i\in S} x_i \sum_{\substack{R\subseteq S \\ |R|=d-1}} E_{d-1}\left(C^{-1/2} V_R V_R^\top C^{-1/2}\right)\end{aligned}$$

$$
\begin{aligned}
&= \frac{\det(C)}{\det(X)} \sum_{|R|=d-1} \prod_{i \in R} x_i E_{d-1}\left(C^{-1/2} V_R V_R^\top C^{-1/2}\right) \sum_{i \notin R} x_i \\
&\leq d \cdot \frac{\det(C)}{\det(X)} \sum_{|R|=d-1} \prod_{i \in R} x_i E_{d-1}\left(C^{-1/2} V_R V_R^\top C^{-1/2}\right) \\
&= (n-1) \cdot \frac{E_{d-1}\left(C^{-1/2} X C^{-1/2}\right)}{\det\left(C^{-1/2} X C^{-1/2}\right)} \\
&= d \cdot \mathrm{Tr}\left(\left(C^{-1/2} X C^{-1/2}\right)^{-1}\right) = d \cdot \left\langle C, X^{-1} \right\rangle .
\end{aligned}
$$

In the last line we use our previous expression for $\mathrm{Tr}(M^{-1})$ in reverse. The only inequality we use is that $\sum_{i \notin R} x_i \leq \sum_i x_i \leq d$.

In the case that C has low rank, as we used for the reconfiguration problem, we need the following limiting argument to finish. Let X be the optimal solution matrix to the convex program with objective C. For any $\epsilon > 0$

$$
\begin{aligned}
\mathbf{E}_S\left[\left\langle C, (V_S V_S^\top)^{-1} \right\rangle\right] &\leq \mathbf{E}_S\left[\left\langle C + \epsilon I, (V_S V_S^\top)^{-1} \right\rangle\right] \\
&\leq (n-1) \cdot \left\langle C + \epsilon I, X^{-1} \right\rangle \\
&= (n-1) \cdot \left\langle C, X^{-1} \right\rangle + (n-1)\epsilon \cdot \mathrm{Tr}\left(X^{-1}\right),
\end{aligned}
$$

and when $\epsilon \to 0$ we conclude that

$$
\mathbf{E}_S\left[\left\langle C, (V_S V_S^\top)^{-1} \right\rangle\right] \leq d \cdot \left\langle C, X^{-1} \right\rangle,
$$

even when C has low rank. □

Remark 1 *We only used the constraint $\sum_{e \in E} x_e \leq n-1$ from the spanning tree polytope in the above calculation. Thus, the same rounding will still work if we keep only the cardinality constraint.*

Note that with this sampling strategy, we will only select sets $S \subseteq E$ which are spanning

trees. This is because $\det\left(\sum_{e\in E} u_e u_e^\top\right) = 1$ *if* S *is a spanning tree and is zero otherwise. This is essentially a consequence of the Matrix-Tree theorem 19 since* $\sum_{e\in E} u_e u_e^\top$ *is matrix-similar to the laplacian of the graph with vertices* V *and edges* S*, after removing one row and edge.*

The approximation ratio from this algorithm is $O(n)$ which matches the previous best know approximation algorithms for the single source case [48]. The advantage of this technique is that is applies to the case of more general demands, where we can allow multiple source vertices.

6.2 The Trouble with Linear Formulations

In the previous section, we studied a general convex/semidefinite formulation for the reconfiguration problem. But, based on the simple combinatorial description with trees and flows, there are other natural formulations with simpler linear constraints and quadratic objectives. In this section we will describe a few such formulations, and show that they are all equivalent to the simplest of the proposed formulations.

The first obvious approach is using big-M constraints to enforce the spanning-tree condition. This leads to the following program, where B is any signed node-arc incidence matrix and $\mathrm{SPT}(G)$ is the spanning tree polytope for the graph G.

$$\begin{array}{ll} \min & \sum_{e\in E} f_e^2 \\ & Bf = d \\ & -Mx_e \le f_e \le Mx_e \quad \forall e \in E, \\ & x \in \mathrm{SPT}(G), \end{array} \tag{Big-M}$$

where we set M to be the sum of the positive demands: $M = \sum_{v:d_v>0} d_v = \frac{1}{2}\sum_{v\in V} |d_v|$.

Lemma 33 *Given a graph* $G = (V, E)$ *and demand vector* $d \in \mathbf{R}^V$ *satisfying* $\sum_{v\in V} d_v = 0$*, if we set* $M = \sum_{v:d_v>0} d_v$ *then the quadratic program* (Big-M) *is a relaxation for the*

network reconfiguration problem.

Proof Let T be a tree in G. Then there is a unique solution f_T to $Bf = d$ which is supported on T, and $\sum_{e \in E} f_{T,e}^2$ is exactly the network reconfiguration objective. Moreover, the largest flow value (after a change of sign) on any edge is at most $\sum_{v:d_v>0} d_v = M$, so we can set $x = \mathbb{1}_T$, and (f, x) will be a solution to (Big-M) with the correct objective value. □

Unfortunately, the big-M constraints do nothing to constrain the program, and this formulation reduces to the standard electrical flow problem with potentials d.

$$\begin{array}{ll} \min & \sum_{e \in E} f_e^2 \\ & Bf = d \end{array} \qquad \text{(Electrical-Flow)}$$

Lemma 34 *For any graph $G = (V, E)$ and demand vector $d \in \mathbf{R}^V$ satisfying $\sum_{v \in V} d_v = 0$, if $M = \sum_{v:d_v>0} d_v$ then the programs* (Big-M) *and* (Electrical-Flow) *are equivalent.*

Proof It is easy to see that the objective value for the program (Electrical-Flow) is at most that of (Big-M). We can take any solution to (Big-M) and convert it to a solution to (Electrical-Flow) with the same objective by dropping the x variables.

For the reverse inequality, suppose we have an optimal solution $f^\star$ to (Electrical-Flow), and assume, by possibly reorienting the edges, that $f^\star \geq 0$. Consider the polytope $\{f \in \mathbf{R}^E : Bf = d, f \geq 0\}$. The vertices of this polytope have acyclic support, and the recession cone is comprised of non-negative circulations. Since adding a non-negative circulation only increases the objective value, we know that $f^\star$ can be written as a convex combination of flows supported on trees. Since the flow on each tree is unique we can write

$$f^\star = \sum_{T \in \mathcal{T}} \lambda_T \cdot f_T,$$

where $\lambda_T \geq 0$ and $\sum_{T \in \mathcal{T}} \lambda_T = 1$, and f_T is the unique flow on T satisfying $Bf_T = d$.

Since the maximum amount of flow on any edge in f_T is at most $\sum_{v:d_v>0} d_v = M$, we know that $0 \leq f_{T,e} \leq M$ for all $e \in E$. Thus, if we set $x = \sum_{T \in \mathcal{T}} \lambda_T \cdot \mathbb{1}_T$, we see that $0 \leq f^\star_e \leq M x_e$ for all $e \in E$, by taking the convex combination of the corresponding inequalities. Finally, the pair $(f^\star, x)$ is a solution to (Big-M) with the same objective. $\square$

Motivated by various more intricate and stronger formulations for the Steiner tree problem [88] we may hope to construct a stronger formulation for the reconfiguration problem. We will jump directly to a version with all the additional trimmings. This formulation only works in the case of a single source vertex $r \in V$, so that $d_v \geq 0$ for all $v \in V \backslash \{r\}$. The idea is to decompose the flow into a sum of single-commodity flows, one for each possible destination vertex. We add constraints so that all the flows must be constrained by the same fractional tree, and to attempt to enforce a consistent orientation of the edges.

$$
\begin{aligned}
\min \quad & \sum_{e \in E} \left(\sum_{v \neq r} d_v (g^v_{ij} + g^v_{ji}) \right)^2 \\
& g^v(\delta^{\text{in}}(i)) - g^v(\delta^{\text{out}}(i)) = \begin{cases} 1 & i = v \\ -1 & i = s \\ 0 & \text{otherwise} \end{cases} \qquad \forall v \in V \backslash \{r\}, i \in V \\
& g^v_{ij} + g^u_{ji} \leq x_e \quad \forall e = \{i,j\} \in E, u, v \in V \backslash \{r\} \\
& g^v_{ij} \geq 0 \quad \forall v \in V \backslash \{r\}, \{i,j\} \in E \\
& x \in \text{SPT}(G),
\end{aligned}
$$

(Multi-Commodity)

Lemma 35 *Given a graph $G = (V, E)$, and a demand vector $d \in \mathbf{R}^V$ satisfying $\sum_{v \in V} d_v = 0$, and with a single source vertex $r \in V$ such that $d_r < 0$, the program* (Multi-Commodity) *is a relaxation for the network reconfiguration problem.*

Proof Let T be a spanning tree of G. For each $v \in V \backslash \{r\}$ there is a unique directed path P_{rv} from r to v in T, and we set $g^v = \mathbb{1}_{P_{rv}}$. This choice of g^v guarantees that the first set of constraints are satisfied. We construct a flow f_T supported on T by setting

$f_{T,e} = \sum_{v \neq r} d_v (g^v_{ij} + g^v_{ji})$ (note that only one of g^v_{ij} and g^v_{ji} will be non-zero). Let B be the signed node-arc incidence matrix with all edges oriented away from the root r. By summing the first set of constraints with the multipliers d_v we see that f_T satisfies $Bf_T = d$, and f is supported on T. Thus the objective is exactly the reconfiguration objective for the tree T.

Finally, we will set $x = \mathbb{1}_T$, and it remains to show that the second set of constraints are satisfied. Since we always chose the vectors g^v to be the indicator vectors of the directed rv-paths, we know that for all edges e there is a consistent direction such that $g^v_{ji} = 0$ for all $v \in V \backslash \{r\}$. Thus the second set of inequalities is always valid since $g^v_{ij} = 0$ if $e \notin T$ and $g^v_{ij} \leq 1 = x_e$ if $e \in T$. □

Even in this more complicated relaxation, with all the bells and whistles, the spanning-tree variables and constraints are not adding anything meaningful to the formulation.

Lemma 36 *For any graph $G = (V, E)$ and demand vector $d \in \mathbf{R}^V$ satisfying $\sum_{v \in V} d_v = 0$, and with a single source vertex $r \in V$ such that $d_r < 0$, the programs* (Multi-Commodity) *and* (Electrical-Flow) *are equivalent.*

Proof For the easy direction, we start with a solution to (Multi-Commodity) and construct a flow. To do this, we set $f_e = \sum_{v \neq r} d_v (g^v_{ij} - g^v_{ji})$ after fixing an orientation for the edge e. Then the vector f satisfies $Bf = d$. Moreover, since $d_v \geq 0$ for $v \in V \backslash \{r\}$, we know that $|f_e| \leq \sum_{v \neq r} d_v (g^v_{ij} + g^v_{ji})$, and thus the objective of the new solution to (Electrical-Flow) is at most the original objective in (Multi-Commodity).

For the reverse direction, we follow the same plan as was used in Lemma 34. Suppose we have an optimal solution $f^\star$ to (Electrical-Flow), and assume that $f^\star \geq 0$. The vertices of the polytope $\{f \in \mathbf{R}^E : Bf = d, f \geq 0\}$ have acyclic support, and the recession cone $\{f \in \mathbf{R}^E : Bf = 0, f \geq 0\}$ is comprised of non-negative circulations. Once again, we know that $f^\star$ can be written as a convex combination of flows supported on trees. Since the

flow on each tree is unique we can write

$$f^\star = \sum_{T \in \mathcal{T}} \lambda_T \cdot f_T,$$

Now, for each tree, we decompose the flow f_T into a collection of single commodity flows. For each tree T and $v \in V \backslash \{r\}$, let g_T^v be the indicator vector of the directed path from r to v in T. Since f was non-negative, all the orientations are consistent. As in Lemma 35 we know that the collection of vectors g_T^v satisfy the constraints of (Multi-Commodity) with the characteristic vector of T, and they also satisfy the additional constraint $f_{T,e} = \sum_{v \neq r} d_v(g_{T,ij}^v + g_{T,ji}^v)$, since only one of the $g_{T,ij}^v$ and $g_{T,ji}^v$ will be non-zero. Thus, we set $g^v = \sum_{T \in \mathcal{T}} \lambda_T \cdot g_T^v$ and $x = \sum_{T \in \mathcal{T}} \lambda_T \cdot \mathbb{1}_T$. Since all the constraints are convex, we see that g^v and x also satisfy the constraints of (Multi-Commodity), but now we also know that $f_e^\star = \sum_{v \neq r} d_v(g_{ij}^v + g_{ji}^v)$, so the objective in (Multi-Commodity) is identical to the original objective value of $f^\star$ in (Electrical-Flow). □

We can see that the proofs of Lemma 34 and Lemma 36 follow a similar pattern. The same general approach should work for any formulation with spanning tree variables, where the extra variables only appear in linear constraints. First, we take a solution to (Electrical-Flow) and decompose it as a linear combination of flows supported on trees. Then on each such tree, the flow is uniquely determined and we can find all any auxiliary variables using the same argument that the formulation was a relaxation. Since the constraints are linear, we can then take the convex combination of the auxiliary variables to get a full solution with the same objective.

6.3 More on the Laplacian-based Formulation

We can compare the results of the previous section with our earlier spectral formulation for the reconfiguration problem. We start with a proposition which puts the Laplacian-based formulation into a similar format to (Electrical-Flow).

Proposition 3 *Let $G = (V, E)$ be a graph and B a signed node-arc incidence matrix. For any non-negative vector $x \in \mathbf{R}^E$, if $L = \sum_{e \in E} x_e \cdot v_e v_e^\top$ then*

$$\left\langle dd^\top, L^+ \right\rangle = \min \left\{ \sum_{e \in E} \frac{f_e^2}{x_e} : Bf = d \right\}$$

Proof We can write $L = B\mathrm{Diag}(x)B^\top$, where $\mathrm{Diag}(x)$ is the diagonal matrix with entries x. Thus,

$$\begin{aligned}
\left\langle dd^\top, L^+ \right\rangle &= d^\top L^+ d \\
&= \left(d^\top (\mathrm{Diag}(x)^{1/2} B^\top)^+ \right) \left((B\mathrm{Diag}(x)^{1/2})^+ d \right) \\
&= \| (B\mathrm{Diag}(x)^{1/2})^+ d \|^2 \\
&= \min \left\{ \sum_{e \in E} g_e^2 : B\mathrm{Diag}(x)^{1/2} g = d \right\} \\
&= \min \left\{ \sum_{e \in E} \frac{f_e^2}{x_e} : Bf = d \right\},
\end{aligned}$$

where the final line follows from the change of variables $f_e = x_e^{1/2} \cdot g_e$. □

With this equivalence, we can show one side of the relationship between the relaxations (Electrical-Flow) and (CP-Reconfiguration).

Lemma 37 *Let $G = (V, E)$ be a graph and $d \in \mathbf{R}^v$ such that $\sum_{v \in V} d_v$. The convex program* (CP-Reconfiguration) *is at least as strong a relaxation as* (Electrical-Flow).

From our earlier discussion this also implies that (CP-Reconfiguration) is at least as strong as both linear formulations (Big-M) and (Multi-Commodity).

Proof Given any solution $x \in \mathrm{SPT}(G)$ let $L = \sum_{e \in E} x_e \cdot v_e v_e^\top$. By Proposition 3 there is a flow $f \in \mathbf{R}^E$ such that $Bf = d$ and

$$\left\langle dd^\top, L^+ \right\rangle = \sum_{e \in E} \frac{f_e^2}{x_e}.$$

We will use the same f as a solution to (Electrical-Flow), and note that $x \in \mathrm{SPT}(G)$ implies that $x_e \leq 1$ for all $e \in E$, so $\frac{f_e^2}{x_e} \geq f_e^2$. Thus, the objective value of (Electrical-Flow) can only be smaller than the objective of (CP-Reconfiguration). $\square$

Remark 2 *In the above proof we only used the constraint that $x_e \leq 1$ for all $e \in E$. Hence, the same argument will work for any constraint polytope for which these inequalities are valid.*

We would like to show that the Laplacian-based formulation can be stronger than the electrical flow objective. To do so we will need lower bounds on the objective, which can be obtained by constructing the dual program. One approach is to rewrite the convex program as a semidefinite program and appeal to standard conic duality arguments.

The program (CP-Reconfiguration) can be reformulated as a semidefinite program by using the Schur complement.

$$\begin{aligned} \min \quad & t \\ & \begin{bmatrix} t & d^\top \\ d & L \end{bmatrix} \succeq 0 \\ & L = \textstyle\sum_{e \in E} x_e \cdot v_e v_e^\top \\ & x \in \mathrm{SPT}(G). \end{aligned}$$

To make taking the dual easier we write the constraints explicitly in the standard minimiza-

tion form for semidefinite programming.

$$\begin{aligned}
\min \quad & t \\
& t \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + \textstyle\sum_{e \in e} x_e \begin{bmatrix} 0 & 0 \\ 0 & v_e v_e^\top \end{bmatrix} \succeq \begin{bmatrix} 0 & -d^\top \\ d^\top & 0 \end{bmatrix} \\
& x(E) = n - 1 \\
& x(E(S)) \leq |S| - 1, \quad \forall S \subseteq V \\
& x \geq 0.
\end{aligned}$$

We have a semidefinite dual variable $\begin{bmatrix} y & b^\top \\ b & Y \end{bmatrix}$ corresponding to the semidefinite constraint, a free dual variable λ, and a non-negative dual variable μ_S for each $S \subseteq V$.

$$\begin{aligned}
\max \quad & -2b^\top d - \lambda(n-1) - \textstyle\sum_{S \subseteq V} \mu_S \cdot (|S| - 1) \\
& y = 1 \\
& \left\langle Y, v_e v_e^\top \right\rangle - \lambda - \textstyle\sum_{S : e \in E(S)} \mu_S \leq 0, \quad \forall e \in E \\
& \begin{bmatrix} y & b^\top \\ b & Y \end{bmatrix} \succeq 0, \mu_S \geq 0.
\end{aligned}$$

Now we can eliminate the variables one at a time until we can write everything in terms of Y. First, we have a constraint that immediately forces $y = 1$. Thus, by taking a Schur complement, the constraint $\begin{bmatrix} y & b^\top \\ b & Y \end{bmatrix} \succeq 0$ is equivalent to

$$b^\top Y^+ b \leq 1.$$

Since b only appear in the objective linearly, it is completely determined once we have fixed the matrix Y. In the optimal solution b will be proportional to Yd, and in particular we will

select

$$b = \frac{-Yd}{\sqrt{d^\top Y Y^+ Y d}} = \frac{-Yd}{\sqrt{d^\top Y d}}.$$

With these two substitutions, the dual problem can be simplified to

$$\begin{array}{ll} \max & 2\sqrt{d^\top Y d} - \lambda(n-1) - \sum_{S \subseteq V} \mu_S \cdot (|S|-1) \\ & v_e^\top Y v_e \leq \lambda + \sum_{S: e \in E(S)} \mu_S \\ & Y \succeq 0, \mu_S \geq 0. \end{array}$$

Now, once we have fixed $Y \succeq 0$ will also try (partially) eliminate the λ and μ_S variables. Set $w_e = v_e^\top Y v_e$, then the optimal choice of λ and μ_S will come from the optimal solution to the following linear program

$$\begin{array}{ll} \min & \lambda(n-1) + \sum_{S \subseteq V} \mu_S \cdot (|S|-1) \\ & w_e \leq \lambda + \sum_{S: e \in E(S)} \mu_S \\ & \mu_S \geq 0. \end{array}$$

This is exactly the dual of the max-weight spanning tree problem with edge weights $w_e = v_e^\top Y v_e$, which we will denote by $\mathrm{maxST}(w)$. Thus, we can write the dual as

$$\begin{array}{ll} \max & 2\sqrt{d^\top Y d} - \mathrm{maxST}(v_e^\top Y v_e) \\ & Y \succeq 0. \end{array}$$

Remark 3 *As an alternative to semidefinite duality we can take the Lagrangian relaxation with respect to the constraint $L = \sum_{e \in E} x_e \cdot v_e v_e^\top$. Following this method, we get an equivalent program*

$$\begin{array}{ll} \min_x \max_Y & \left\langle dd^\top, L^+ \right\rangle + \left\langle Y, L - \sum_{e \in E} x_e \cdot v_e v_e^\top \right\rangle \\ & x \in \mathrm{SPT}(G). \end{array}$$

In this form, the variables Y and L are free, so we can use the first order optimality con-

ditions of the objective with respect to L to eliminate the Y variable. The gradient of the objective with respect to L is

$$-L^+dd^\top L^+ + Y,$$

so in the optimal solution we set $Y = L^+dd^\top L^+$. If we make this substitution the objective becomes

$$\left\langle dd^\top, L^+ \right\rangle + \left\langle L^+dd^\top L^+, L - \sum_{e\in E} x_e \cdot v_e v_e^\top \right\rangle = 2d^\top L^+ d - \sum_{e\in E} x_e \cdot (v_e^\top L^+ d)^2.$$

Note that this is equivalent to what we get if we make the same substitution for Y in the semidefinite dual above.

We can collect the results of this discussion in the following lemma, which we will use as our lower bound.

Lemma 38 *For any $n \times n$ positive semidefinite matrix Y, the optimal value of the relaxation* (CP-Reconfiguration) *is at least*

$$2\sqrt{d^\top Y d} - \mathrm{maxST}(v_e^\top Y v_e).$$

In particular, if L is a feasible solution to (CP-Reconfiguration) *then the duality gap is at most*

$$\mathrm{maxST}((v_e^\top L^+ d)^2) - d^\top L^+ d.$$

Proof The first part of the result follows from the duality transformations above, and weak duality for semidefinite programs. Given a feasible solution L to (CP-Reconfiguration) we can set $Y = L^+dd^\top L$. Then the primal objective is $d^\top L^+ d$ and the dual lower bound is

$$2\sqrt{d^\top L^+ dd^\top L^+ d} - \mathrm{maxST}(v_e^\top L^+ dd^\top L^+ v_e) = 2 \cdot d^\top L^+ d - \mathrm{maxST}((v_e^\top L^+ d)^2),$$

and the duality gap is at most the difference between the objective and the lower bound. □

To make this slightly more useful, it will help to understand what the edge weights $(v_e^\top L^+ d)^2$ are for a specific solution $L = \sum_{e \in E} x_e \cdot v_e v_e^\top$. The following lemma will also be useful later to help calculate the gradient of our objective function.

Lemma 39 *Let $G = (V, E)$ be a graph, B a signed node-arc incidence matrix, and d a demand vector. If $L = \sum_{e \in E} x_e \cdot v_e v_e^\top$, and $f \in \mathbf{R}^E$ is the flow minimizing $\sum_{e \in E} \frac{f_e^2}{x_e}$ subject to $Bf = d$ then*

$$v_e^\top L^+ d = \frac{f_e}{x_e}.$$

Proof We follow the same calculation as in Proposition 3. We can write $L = B\mathrm{Diag}(x)B$ and then

$$\begin{aligned} L^+ d &= (\mathrm{Diag}(x)^{1/2} B^\top)^+ (B\mathrm{Diag}(x)^{1/2})^+ d \\ &= (\mathrm{Diag}(x)^{1/2} B^\top)^+ g, \end{aligned}$$

where $g_e = \frac{f_e}{\sqrt{x_e}}$. Now, rearranging, we can write

$$(\mathrm{Diag}(x)^{1/2} B^\top)^+ g = \phi,$$

where $\phi \in \mathbf{R}^V$ satisfies

$$B^\top \phi = \mathrm{Diag}(x)^{-1/2} g$$

(ϕ will have minimum norm among all such vectors, but we do not need that fact). In particular, since the columns of B are exactly the vectors v_e, we know that

$$v_e^\top L^+ d = v_e^\top \phi = \frac{1}{\sqrt{x_e}} \frac{f_e}{\sqrt{x_e}} = \frac{f_e}{x_e}.$$

□

Remark 4 *Using this lemma it is easy to verify directly that the duality gap bound from Lemma 38 is always non-negative. Given $x \in \mathrm{SPT}(G)$ we know that*

$$\mathrm{maxST}(f_e^2/x_e^2) \geq \sum_{e \in E} x_e \cdot \frac{f_e^2}{x_e^2} = \sum_{e \in E} \frac{f_e^2}{x_e} = d^\top L^+ d.$$

6.3.1 A Gap Between Electrical Flow and the Convex Relaxation

In this section we will construct an example of a graph where there is a gap between the value of the (Electrical-Flow) and the value of (CP-Reconfiguration). Moreover, the later will be an exact formulation for the example we construct.

It is easy to construct examples where the electrical flow value is very far from the integral optimal solution. For example consider a graph with only two vertices s, t and k parallel edges between them, with demands $d_s = -1, d_t = +1$. In the optimal solution to (Electrical-Flow) we will put an equal amount of flow $\frac{1}{k}$ on each edge, resulting in an objective value of $k \cdot (1/k)^2 = 1/k$. The integral solution is just to take any one of the k edges and send all the flow along it, resulting in an integral optimal value of 1. From this example we can see that the gap can be as large as the number of edges in the graph. It turns out that (CP-Reconfiguration) is also exact for this graph.

In some ways this is a silly example, since we are taking advantage of the parallel edges. But we could try to create another example by subdividing each edge. This would create a graph with two vertices s, t connected by k distinct paths of length 2 (see Figure 6.1). If we keep the demands as $d_s = -1, d_t = +1$ and $d_v = 0$ otherwise, then the optimal integral solution is the length of the shortest path which is 2. For the electrical flow problem, we again split our flow evenly, and the resulting objective is $2k \cdot (1/k)^2 = 2/k$. Unfortunately, in this case the optimal value of (CP-Reconfiguration) is $\frac{4}{k+1}$ which is also far from the integral value.

Lemma 40 *If G is a graph consisting of k disjoint paths of length 2 between s and t (as in Figure 6.1) and $d_s = -1, d_t = +1$ and $d_v = 0$ otherwise, then the optimal value*

of (CP-Reconfiguration) *is* $\frac{4}{k+1}$.

Proof We set $x_e = \frac{k+1}{2k}$ on every edge. Since x is a constant vector, the optimal flow is to equally distribute the flow along the paths, setting $f_e = \frac{1}{k}$ on every edge. Thus, the primal objective value is

$$\sum_{e \in E} \frac{f_e^2}{x_e} = 2k \cdot \left(\frac{1}{k}\right)^2 \frac{2k}{k+1} = \frac{4}{k+1}.$$

To see that there is no better solution, we consider the duality gap from 38. By applying Lemma 39 we can calculate the maximum spanning tree with edge weights are $\left(\frac{f_e}{x_e}\right)^2$. All edges have the same weight, so

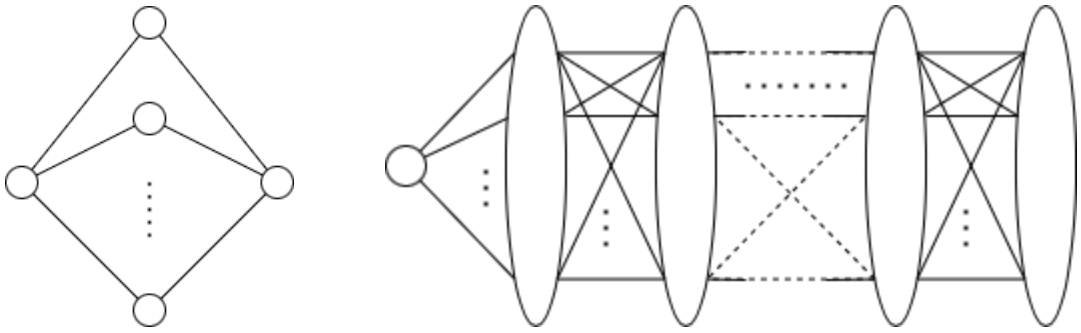
$$\text{maxST}((f_e/x_e)^2) = (k+1) \cdot \left(\frac{1}{k}\frac{2k}{k+1}\right)^2 = \frac{4}{k+1},$$

which confirms that the duality gap is zero. □

One reason for this is that we are not using the correct relaxation given the specific demand vector. We could improve the objective if we allowed the vector $x \in \mathbf{R}^E$ to be a fractional (s,t)-path instead of a fractional spanning tree (with this change we again get an exact formulation for the specific example). Unfortunately, this is not enough to save us. We can put arbitrarily small demands $\epsilon > 0$ at all the intermediate vertices $v \neq s,t$. In this case the correct formulation is the spanning tree model, and we again get an integrality gap of $\Omega(k)$, where k is the number of paths (and there are $k+2$ vertices total). This shows that, with arbitrary demands, the integrality gap of (CP-Reconfiguration) can be as bad as $\Omega(|V|)$.

For now we will restrict our attention to the case of unit demands where we have a single root vertex r such that $d_v = 1$ for all $v \neq r$ and $d_r = 1 - |V|$. In this case, we can show separation between the objective of (CP-Reconfiguration) and (Electrical-Flow).

Lemma 41 *Let G consist of a single root vertex r and k sets $V_1, \ldots, V_k$ each of k vertices such that r is adjacent to every vertex in V_1, and so that V_i and V_{i+1} are connected by a*



(a) Parallel paths (b) Complete-bipartite chain

Figure 6.1: Graph structure for integrality gap examples

complete bipartite graph for $i = 1, \ldots, k-1$ *(See Figure 6.1). If we have unit demands (*$d_r = -k^2$ *and* $d_v = 1$ *otherwise), then*

- *the optimal value of the reconfiguration problem is* $\frac{k^4(k+1)(2k+1)}{6} = \Theta(k^4)$,
- *the optimal value of* (CP-Reconfiguration) *is* $\frac{k^4(k+1)(2k+1)}{6} = \Theta(k^4)$, *and*
- *the optimal value of* (Electrical-Flow) *is* $k^3 + \frac{(k-1)k(2k-1)}{6} = \Theta(k^3)$.

Proof Let G be a complete-bipartite chain as in Figure 6.1 where there are k parts each of size k, as in the lemma statement.

Consider a spanning tree T which consist of k disjoint paths of length $k+1$ from r to distinct vertices in the final part V_k. The total reconfiguration objective for this tree is

$$k \cdot \left(k^2 + (k-1)^2 + \cdots + 2^2 + 1^2\right).$$

By analyzing the optimal value of (CP-Reconfiguration) we will see that it is not possible to do any better.

We will start with the fractional spanning tree where $x_e = 1$ for all edges incident at r, and $x_e = \frac{1}{k}$ otherwise. The optimal flow with these conductances is $f_e = k$ for all edges incident with r, and $f_e = \frac{k-i}{k}$ for all the edges between V_i and V_{i+1}. This way the net flow from V_i to V_{i+1} is $k^2 \cdot \frac{k-i}{k} = k(k-i)$, as required. It is easy to verify that this is the optimal

flow by constructing potentials which are constant within each part. The total objective value for this solution is

$$\sum_{e \in E} \frac{f_e^2}{x_e} = k \cdot k^2 + k^2 \cdot \sum_{i=1}^{k-1} k \cdot \frac{(k-i)^2}{k^2} = k \cdot \left(k^2 + (k-1)^2 + \cdots + 2^2 + 1^2\right).$$

This matches the previous integral solution, and we will show both are optimal by calculating the duality gap. By Lemma 39, this requires calculating the maximum spanning tree with weights $\frac{f_e^2}{x_e^2}$. This weight is k^2 on the edges incident with r, and $(k-i)^2$ for the edges between V_i and V_{i+1}. When we greedily construct the maximum spanning tree, we select exactly k edges from each cut and the resulting total weight is exactly

$$k \cdot \left(k^2 + (k-1)^2 + \cdots + 2^2 + 1^2\right).$$

By plugging this into the duality gap from Lemma 38 we conclude that this, and the previous integral solution are optimal.

Finally, for the electrical flow problem we use the same flow, which is still optimal with unit conductances. Its objective value is

$$\sum_{e \in E} f_e^2 = k \cdot k^2 + k^2 \cdot \sum_{i=1}^{k-1} \frac{(k-i)^2}{k^2} = k^3 + \left((k-1)^2 + (k-2)^2 + \cdots + 2^2 + 1^2\right).$$

The final formulas can be obtained by applying the equation for the sum of the first n squares. □

6.4 Lower-Bound Relaxations and Upper-Bound Heuristics

In this section we provide a collection of practically efficient algorithms which can be used to provide either upper or lower bounds on the network reconfiguration problem.

The lower bounds come from solving (Electrical-Flow) and (CP-Reconfiguration). The

first only requires solving a single linear system; given a signed node-arc incidence matrix B and demands d, the minimizer will be exactly the vector B^+d. The convex relaxation takes more work, but can be solved to relatively high accuracy using the Frank-Wolfe method. This approach is detailed in Section 6.4.1.

For the upper bounds, we have combinatorial algorithms which output a feasible spanning tree. The first is a greedy deletion algorithm, which is a standard technique for A-design [53], and is outlined in Section 6.4.2. Finally, we will discuss the local search approach to the network reconfiguration problem. This is one of the most commonly studied algorithms [47]. While it does seem to perform well in practice, we provide an example showing that the approximation gap can be quite large.

6.4.1 Frank-Wolfe for Solving CP-Reconfiguration

While the convex program (CP-Reconfiguration) can be reformulated as a semidefinite program and solved using an out-of-the-box conic programming solver, this quickly becomes impractical. Because of the special structure of the feasible region, we can use the Frank-Wolfe method to quickly find approximately optimal solutions. See [89] for a full survey of the method; we will borrow the main terminology and notation.

In the abstract, the Frank-Wolfe method is a way to solve constrained convex optimization problems when we have access to an oracle that can perform linear optimization over the feasible region. We are given a (potentially convex and smooth) function $g : \mathbf{R}^n \to \mathbf{R}$ and a convex feasible region $P \subseteq \mathbf{R}^n$. The method is summarized in Algorithm 10.

Algorithm 10 The Frank-Wolfe Algorithm

1: **Input**: $x_0 \in P$ and a convex function f

2: **for** $t = 0, \ldots$ **do**

3: $\quad v_t \leftarrow \operatorname{argmin}_{v \in P} \langle \nabla g(x_t), v \rangle$

4: $\quad \gamma_t \leftarrow \frac{1}{t+2}$

5: $\quad x_{t+1} \leftarrow (1 - \gamma_t) x_t + \gamma_t v_t$

6: **end for**

For a given feasible region P and convex function f, the Frank-Wolfe method requires that we are able to calculate the gradient of f, and also solve the optimization problem $\min_{v \in P} \langle c, v \rangle$ when $c = \nabla f(x)$. Concretely, for our problem, the feasible region is $P = \mathrm{SPT}(G)$ is the spanning tree polytope for a graph G, and

$$g(x) = \left\langle dd^\top, \left(\sum_{e \in E} x_e \cdot v_e v_e^\top \right)^+ \right\rangle.$$

Proposition 4 *If* $g : \mathbf{R}^E \to R$ *is given by* $f(x) = \left\langle dd^\top \left(\sum_{e \in E} x_e \cdot v_e v_e^\top \right)^+ \right\rangle$, *then* $\nabla g(x)_e = -(v_e^\top L_x^+ d)^2$, *where* $L_x = \sum_{e \in E} x_e \cdot v_e v_e^\top$.

Proof Let $y \in \mathbf{R}^E$ and consider the directional derivative of f in the direction of y. By Proposition 1, the directional derivative will be

$$- \left\langle L_x^+ dd^\top L_x^\top, \sum_{e \in E} y_e \cdot v_e v_e^\top \right\rangle = \sum_{e \in E} y_e \cdot \left\langle L_x^+ dd^\top L_x^\top, v_e v_e^\top \right\rangle = \sum_{e \in E} y_e \cdot (v_e^\top L_x^+ d)^2,$$

and the conclusion follows since y was arbitrary. □

This proposition gives a concrete way to calculate the gradient. Now, for any particular point $x \in \mathrm{SPT}(G)$ the optimization problem $\min_{v \in \mathrm{SPT}(G)} \langle \nabla g(x), v \rangle$ is equivalent to computing a maximum weight spanning tree with weights $(v_e^\top L_x^+ d)^2$. This can be done efficiently once we know the weights.

Finally, we provide a stopping condition for the method by maintaining a dual bound at each iteration. Given a threshold $\epsilon > 0$ we will stop iterating once the relative gap between the dual bound and the optimal objective value is at most $1 + \epsilon$.

Algorithm 11 Frank-Wolfe iteration for solving (CP-Reconfiguration)

Input: Graph $G = (V, E)$ and demand vector $d \in \mathbf{R}^V$ satisfying $\sum_{v \in V} d_v = 0$, and accuracy parameter $\epsilon > 0$.

$x_0 \leftarrow$ any fractional spanning tree in $\mathrm{SPT}(G)$.

$t \leftarrow 0$

$\mathrm{obj} \leftarrow \infty$

$\mathrm{dual} \leftarrow 0$

while $\mathrm{obj}/\mathrm{dual} > 1 + \epsilon$ **do**

  $f \leftarrow$ optimal solution to $\min \left\{ \sum_{e \in E} \frac{f_e^2}{x_e} : Bf = d \right\}$.

  $\mathrm{obj} \leftarrow \sum_{e \in E} \frac{f_e^2}{x_e}$.

  $T \leftarrow$ maximum spanning tree with edge weights $w_e := \frac{f_e^2}{x_e^2}$.

  $\mathrm{dual} \leftarrow \max(\mathrm{dual},\, 2 \cdot \mathrm{obj} - w(T))$

  $\gamma_t \leftarrow \frac{1}{t+2}$

  $t \leftarrow t + 1$

  $x_t = x_{t-1} + \gamma_t(\mathbb{1}_T - x_{t-1})$.

end while

Return x_t.

Recall from Proposition 3 that the objective value

$$g(x) = \min \left\{ \sum_{e \in E} \frac{f_e^2}{x_e} : Bf = d \right\},$$

where B is a node-arc incidence matrix for the underlying graph. This minimization problem can be solved by solving a single linear system to obtain the flow vector f. Using

Lemma 38 and Lemma 39 we can maintain the dual bound

$$2 \cdot g(x) - \text{maxST}\left(\frac{f_e^2}{x_e^2}\right).$$

Note that both terms were already computed as part of the iteration and linear optimization problem. Combining all these results, we get the version of the Frank-Wolfe method specialized to the case of the network reconfiguration problem in Algorithm 11.

6.4.2 Greedy Delete

The greedy deletion algorithm has been used in A-design when the number of vectors selected is at least the dimension. The network reconfiguration problem is at the extreme end of this range, but we hope that the special vector structure could lead to improvements. The idea is to start with every available edge already selected, and then iteratively delete the edge which results in the smallest increase to the objective.

To make computing the change in objective for each edge more efficient, we will use the Woodbury matrix identity.

Lemma 42 *Let A be an $n \times n$ matrix, and U, C, K be $n \times k$, $k \times k$ and $k \times n$ respectively. Then as long as the below matrices are invertible, we have the identity*

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}.$$

This identity will also be useful later in Section 6.4.3 to analyze the discrete local-optimality conditions of the network reconfiguration objective.

Lemma 43 *Given a matrix L and a vector v, so long as the $v^\top L^{-1} v \neq 1$ we have the identity*

$$\left(L - vv^\top\right)^{-1} = L^{-1} - \frac{L^{-1}vv^\top L^{-1}}{(1 - v^\top L^{-1} v)}$$

Proof We write the update in terms of the Woodbury matrix identity:

$$L - vv^\top = L + v(-v)^\top.$$

Thus

$$L - vv^\top = L^{-1} + L^{-1}v\left(1 - v^\top L^{-1}v\right)^{-1} vL^{-1} = L^{-1} + \frac{L^{-1}vv^\top L^{-1}}{(1 - v^\top L^{-1}v)}.$$

□

Using this lemma, we see that the absolute change in objective after deleting an edge e will be exactly

$$\left\langle dd^\top, L^{-1} + \frac{L^{-1}v_e v_e^\top L^{-1}}{(1 - v_e^\top L^{-1}v_e)} \right\rangle - \left\langle dd^\top, L^{-1} \right\rangle = \frac{(d^\top L^+ v_e)^2}{(1 - v_e^\top L^+ v_e)}.$$

Algorithm 12 Greedy Delete For Network Reconfiguration

Input: Graph $G = (V, E)$ and demand vector $d \in \mathbf{R}^V$ satisfying $\sum_{v \in V} d_v = 0$, and accuracy parameter $\epsilon > 0$

$L \leftarrow \sum_{e \in E} v_e v_e^\top$

while $|E| > |V| - 1$ **do**

$e \leftarrow \text{argmin}_{e \in E} \left\{ \frac{(d^\top L^+ v_e)^2}{(1 - v_e^\top L^+ v_e)} \right\}$

$L \leftarrow L - v_e v_e^\top$

$E \leftarrow E \backslash \{e\}$

end while

Output: E the edge set of a tree

In each iteration, we can calculate all the vectors $L^+ v_e$ by solving a linear system, and then use the solution to calculate the change in objective.

6.4.3 Local Search

In [47] they provided the first theoretical analysis of the natural local-search algorithm for the network reconfiguration problem. The approach was to extend the objective to a supermodular function, and then apply out-of-the-box approximation algorithms for maximizing a submodular function with a matroid base constraint. In this section we will attempt to directly analyze local-search using the local-optimality conditions and the convex relaxation, and provide a new lower bound. The result will be the following theorem, which shows that local search can result in a tree that is far from optimal.

Theorem 7 *There is a graph G on n vertices and a spanning tree T of G such that T is locally optimal for the network reconfiguration objective, and the gap between the objective of T and the optimal solution is*

$$\Omega\left(\frac{n}{(\log\log n)^2}\right).$$

For the sake of brevity we will first establish some notation for the leverage scores. For a fixed matrix L we define $\tau_{ij} = v_i^\top L^+ v_j$.

Lemma 44 *Given a matrix L and vectors v_i, v_j, so long as all the below matrices are invertible, we have the identity*

$$\begin{aligned}&(L + v_i v_i^\top - v_j v_j^\top)^{-1} \\ &\qquad = L^{-1} - \frac{L^{-1}\left((1-\tau_{jj})v_i v_i^\top + \tau_{ij} v_i v_j^\top + \tau_{ij} v_j u_i^\top - (1+\tau_{ii}) v_j v_j^\top\right) L^{-1}}{(1+\tau_{ii})(1-\tau_{jj}) + \tau_{ij}^2}.\end{aligned}$$

Proof First we rewrite the local update in the form of the Woodbury matrix identity (Lemma 42).

$$L + v_i v_i^\top - v_j v_j^\top = L + \begin{bmatrix} v_i & v_j \end{bmatrix} \begin{bmatrix} v_i & -v_j \end{bmatrix}^\top.$$

Thus,

$$(L + v_i v_i^\top - v_j v_j^\top)^{-1}$$
$$= L^{-1} - L^{-1} \begin{bmatrix} v_i & v_j \end{bmatrix} \left(I_2 + \begin{bmatrix} v_i & -v_j \end{bmatrix}^\top L^{-1} \begin{bmatrix} v_i & v_j \end{bmatrix} \right)^{-1} \begin{bmatrix} v_i & -v_j \end{bmatrix}^\top L^{-1}.$$

Now we can break it down into pieces.

$$I_2 + \begin{bmatrix} v_i & -v_j \end{bmatrix}^\top L^{-1} \begin{bmatrix} v_i & v_j \end{bmatrix} = I_2 + \begin{bmatrix} v_i^\top L^{-1} v_i & v_i^\top L^{-1} v_j \\ -v_j^\top L^{-1} v_i & -v_j^\top L^{-1} v_j \end{bmatrix}$$
$$= \begin{bmatrix} 1 + \tau_{ii} & \tau_{ij} \\ -\tau_{ij} & 1 - \tau_{jj} \end{bmatrix},$$

and so

$$\left(I_2 + \begin{bmatrix} v_i & -v_j \end{bmatrix}^\top L^{-1} \begin{bmatrix} v_i & v_j \end{bmatrix} \right)^{-1} = \frac{1}{(1+\tau_{ii})(1-\tau_{jj}) + \tau_{ij}^2} \begin{bmatrix} 1 - \tau_{jj} & -\tau_{ij} \\ \tau_{ij} & 1 + \tau_{ii} \end{bmatrix}.$$

Now note that

$$\begin{bmatrix} v_i & v_j \end{bmatrix} \begin{bmatrix} 1 - \tau_{jj} & -\tau_{ij} \\ \tau_{ij} & 1 + \tau_{ii} \end{bmatrix} \begin{bmatrix} v_i & -v_j \end{bmatrix}^\top = (1-\tau_{jj}) v_i v_i^\top + \tau_{ij} v_i v_j^\top + \tau_{ij} v_j v_i^\top - (1+\tau_{ii}) v_j v_j^\top,$$

and the claim follows by making all the necessary substitutions. □

Now we can state the complete local optimality conditions for our specific problem.

Lemma 45 *Let G be a graph and T be a spanning tree. Then T is locally optimal for the objective $d^\top L_T^+ d$ if and only if*

$$(1 + \tau_{ii})(d^\top L_T^+ v_j)^2 - 2\tau_{ij}(v_j^\top L_T^+ d)(d^\top L_T^+ v_i) \geq 0,$$

for all $i \notin T$ and $j \in T$ such that $T \cup \{i\} \backslash \{j\}$ is also a spanning tree.

Proof The tree T is locally optimal if and only if

$$\left\langle dd^\top, L_T^+ \right\rangle \leq \left\langle dd^\top, (L_T + v_i v_i^\top - v_j v_j^\top)^+ \right\rangle$$

for all $i \notin T$ and $j \in T$ such that $T \cup \{i\} \backslash \{j\}$ is also a spanning tree. We can expand the right-hand side using Lemma 44 to see that

$$\begin{aligned}
&\left\langle dd^\top, (L_T + v_i v_i^\top - v_j v_j^\top)^+ \right\rangle \\
&\quad \leq \left\langle dd^\top, L_T^+ \right\rangle - \frac{\left\langle dd^\top, L^+ \left((1-\tau_{jj}) v_i v_i^\top + \tau_{ij} v_i v_j^\top + \tau_{ij} v_j u_i^\top - (1+\tau_{ii}) v_j v_j^\top \right) L^+ \right\rangle}{(1+\tau_{ii})(1-\tau_{jj}) + \tau_{ij}^2},
\end{aligned}$$

By subtracting the first term, and multiplying through by the denominator, this is equivalent to

$$\begin{aligned}
0 &\leq \left\langle dd^\top, L^+ \left(-(1-\tau_{jj}) v_i v_i^\top - \tau_{ij} v_i v_j^\top - \tau_{ij} v_j u_i^\top + (1+\tau_{ii}) v_j v_j^\top \right) L^+ \right\rangle \\
&= (1+\tau_{ii})(d^\top L_T^+ v_j)^2 - 2\tau_{ij}(v_j^\top L_T^+ d)(d^\top L_T^+ v_i) - (1-\tau_{jj})(d^\top L_T^+ v_i)^2 \\
&= (1+\tau_{ii})(d^\top L_T^+ v_j)^2 - 2\tau_{ij}(v_j^\top L_T^+ d)(d^\top L_T^+ v_i),
\end{aligned}$$

where in the last line we use the fact that $\tau_{jj} = 1$. □

We can get a better understanding of the leverage scores using our specific problem structure.

Proposition 5 *Let G be a graph and T be a spanning tree. If $i \notin T$ and $j \in T$ are edges such that $T \cup \{i\} \backslash \{j\}$ is also a spanning tree then*

- $\tau_{ii} := v_i^\top L_T^+ v_i$ *is equal to the length of the unique path in T between the endpoints of edge i,*
- $\tau_{jj} := v_j^\top L_T^+ v_j = 1$,

- $\tau_{ij} := v_i^\top L_T^+ v_j = \pm 1$, *depending on the direction of the edge* j *in the fundamental cycle for edge* i,
- $d^\top L_T^+ v_i$ *is equal to the (signed) sum of flow values along the unique path in* T *between endpoints of edge* i, *and*
- $d^\top L_T^+ v_j$ *is equal to the (signed) flow value on the edge* j.

Proof To see this, note that $L_T = B_T B_T^\top$ where B_T is a signed node-arc incidence matrix for the tree T, so $\tau_{ij} = \left\langle (B_T^\top)^+ v_i, (B_T^\top)^+ v_j \right\rangle$ (with similar expressions for τ_{ii} and τ_{jj}). Since the unique solution to $B_T f = \mathbb{1}_t - \mathbb{1}_s$ is the (signed) edge-indicator vector of the path from s to t, $(B_T^\top)^+ v_i$ is an indicator vector for the path between the endpoints of edge i, while $(B_T^\top)^+ v_j$ is an indicator for the edge j itself. For the last two facts, recall that $(B_T^\top)^+ d$ is exactly the unique flow in T which satisfies the demands d. □

Given a tree T and an edge $i \notin T$, let $C_T(i)$ denote the fundamental cycle of T created by adding the edge i.

If T has only a single source vertex, then we can orient the edges of T so that they are directed away from the source. Then $d^\top L_T^+ v_j \geq 0$ for all $j \in T$ and $d L_T^+ v_i$ is equal to the total flow imbalance between the two paths from the source to the endpoints of edge i. These paths will only disagree for edges in the fundamental cycle $C_T(i)$. The orientation of T decomposes $C_T(i) \backslash \{i\}$ into two components, $C_T^+(i)$ and $C_T^-(i)$, which we have labeled so that $f(C_T^+(i)) \geq f(C_T^-(i))$. With this notation $d L_T^+ v_i = \pm (f(C_T^+(i)) - f(C_T^-(i))$, where the sign depends on the orientation of the edge i. This discussion leads to the following lemma.

Lemma 46 *Let* G *be a graph and* d *be a demand vector with single source node. A spanning tree* T *is locally optimal for the objective* $d^\top L_T^+ d$ *if and only if*

$$\min_{j \in C_T^+(i)} f_j \geq \frac{2 \cdot (f(C_T^+(i)) - f(C_T^-(i)))}{|C_T(i)|},$$

for all edges $i \notin T$.

Proof We begin by orienting all the edges of T away from the source vertex. From Lemma 45 we know that T is locally optimal if and only if

$$(1 + \tau_{ii})(d^\top L_T^+ v_j)^2 - 2\tau_{ij}(v_j^\top L_T^+ d)(d^\top L_T^+ v_i) \geq 0$$

for all $i \notin T$ and $j \in C_T(i)$. We always orient the edge $j \in C_T(i)$ so that it aligns with the orientation of T. Thus $d^\top L_T^+ v_j \geq 0$ for all $j \in C_T(i)$. Using this, we can rearrange and cancel the $d^\top L_T^+ v_j$ factors on either side to get the following equivalent condition:

$$(1 + \tau_{ii}) \cdot d^\top L_T^+ v_j \geq 2\tau_{ij} \cdot d^\top L_T^+ v_i.$$

Recall that τ_{ii} is the length of the unique path in T between the endpoints of i. Hence, $(1 + \tau_{ii}) = |C_T(i)|$. Note that reorienting the edge i does not affect the sign on the right-hand side, so we assume that i is directed so that it points from $C_T^-(i)$ to $C_T^+(i)$. Then $d\top L_T^+ v_i = f(C_T^+(i)) - f(C_T^-(i)) \geq 0$, and $\tau_{ij} = 1$ if and only if $j \in C_T^+(i)$. This implies that the above inequality trivially holds for all $j \in C_T^-(i)$, so it suffices to check that

$$|C_T(i)| f_j \geq 2 \cdot (f(C_T^+(i)) - f(C_T^-(i))),$$

for all $j \in C_T^+(i)$. The result follows by rearranging and taking the minimum on the left-hand-side. □

It is worth noting that the minimum will always be attained at the "lowest" edge, i.e. the edge which shares an endpoint with i. Because we have fixed our orientation the flow values are non-decreasing along any path from the root.

Observe that nothing about the structure of G, other than the presence of the edge i, is used in the proof. Thus, we can turn this lemma on its head, to start with a tree T, and

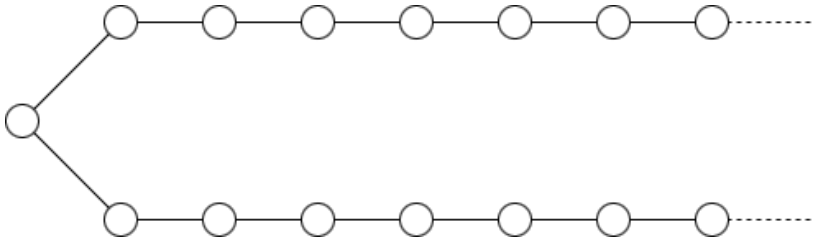


Figure 6.2: Two-path tree

construct a supergraph G such that T is still locally optimal by adding all the edges i which satisfy the above inequality.

The tree T we want to start with is a single root vertex, with two paths of equal length n.

Proposition 6 *Let T be a tree on $2n+1$ vertices with a single root vertex r and two paths of length n. If the demands are $d_v = 1$ for all $v \neq r$, and $d_r = -2n$ then the objective $d^\top L_T^+ d = \Theta(|T|^3)$.*

Proof On each path of length n the flow into the k-th vertex from the bottom is exactly k. Thus the total objective is

$$2 \cdot \sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{3} = \Theta(n^3).$$

□

If we add edges to the graph while maintaining that T is locally optimal, it turns out that, in the resulting graph, we will be able to find another spanning tree that looks like a complete binary tree.

Proposition 7 *Let T' be a complete binary tree with root r. If the demands are $d_v = 1$ for all internal vertices $v \neq r$ and $d_v \leq x$ for all leaves v of the tree, then the objective*

$$d^\top L_T^+ d = O(x^2 \cdot |T'|^2).$$

Proof Increasing the demand at the leaves only increases the flow values and final objective, so we may assume that every leaf has demand exactly x. Now, let k denote the depth of the tree T', and label the levels so that leaves are on level 0, and the root is the only vertex on level k.

Let a_ℓ denote the flow value on the edges into vertices on level ℓ. Then $a_0 = x$ and we have the recurrence $a_\ell = 2a_\ell + 1$ for $\ell = 1, \ldots, k-1$. Solving the recurrence, we see that $a_\ell = (x+1)2^\ell - 1$.

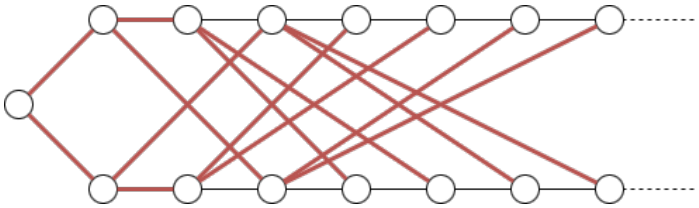
Now, there are $2^{k-\ell}$ total edges into vertices on level ℓ, so the final objective is

$$\begin{aligned}\sum_{\ell=0}^{k-1} 2^{k-\ell} a_\ell^2 &\leq (x+1)^2 \sum_{\ell=0}^{k-1} 2^{k-\ell}(2^\ell)^2 \\ &= (x+1)^2 \sum_{\ell=0}^{k-1} 2^{k+\ell} \\ &= (x+1)^2 2^k \sum_{\ell=0}^{k-1} 2^\ell \leq (x+1)^2 2^{2k}.\end{aligned}$$

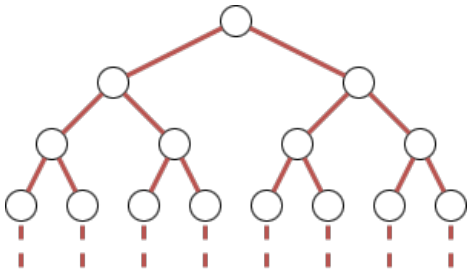
Since the number of vertices in T' is exactly 2^{k-1} we see that the objective is $O(x^2|T'|^2)$. □

Unfortunately, it will not be possible to get a complete binary tree that uses every vertex of the original tree T. Instead, we will can get $2/3|T|$ vertices into the binary tree, and then have paths of length at most $\log\log(n)$ hanging off the leaves to collect the remaining vertices. These paths of length $\log\log(n)$ contribute a negligible amount of flow, and if we ignore them we get a complete binary tree where the leaves have demand at most $\log\log(n)$. This tree has objective $O((\log\log n)^2 n^2)$ while the objective of the original tree T was $\Omega(n^3)$. Since we only added edges that keep T locally optimal this implies, that in the worst case, local search cannot achieve better than an $O\left(\frac{n}{(\log\log n)^2}\right)$-approximation.

In the rest of this section we will justify why we can find a complete binary tree with paths of length at most $\log\log n$ hanging off the leaves.



(a) Two-path tree with extra edges



(b) Unwound binary tree

Figure 6.3: Finding a binary tree while maintaining local-optimality

First, we will show that for the right choice of n we can find a complete binary tree that includes all vertices of T up to depth $2n/3$.

Lemma 47 *Let $n = 3 \cdot 2^{k-1} - 2$ and T be a tree with a root r and two paths of length n. After adding all edges such that T is still locally optimal, there exists a complete binary tree of depth $k+1$ with root r such that all vertices of distance at most $2^{k-1} - 1$ from r are internal vertices, and all vertices of distance between 2^{k-1} and $2^k - 1$ are leaves.*

Proof To begin, we add the edges from r to its two neighbors in T. Then we will show that for a vertex v at distance ℓ from r, we can add the edges from v to the vertices in the other path at distance 2ℓ and $2\ell + 1$ from r, and T will still be locally optimal. This will be true so long as $1 \leq \ell \leq 2^{k-1} - 1$, so all vertices in this range will be internal vertices, and the leaves will be vertices at distance between 2^{k-1} and $2^k - 1$ from r.

First, we will show that we can add the edge e from a vertex of depth ℓ to the vertex in the opposite path at distance 2ℓ. We do this by using Lemma 46. We need to calculate the length of the cycle, the imbalance, and the flow on the smallest edge.

The fundamental cycle will have length $|C_T(e)| = 3\ell + 1$, the smallest flow on the cycle is on the edge into the vertex at distance 2ℓ from r, with value $n - 2\ell + 1$, and the imbalance

is the sum of the flow values between the vertices at distance ℓ and at distance 2ℓ.

$$f(C_T^+(e)) - f(C_T^-(e)) = \sum_{i=\ell}^{2\ell-1}(n-i) = n\cdot\ell - \sum_{i=\ell}^{2i-1} i = n\cdot i - \frac{(3\ell-1)\ell}{2}.$$

By plugging this into the inequality from Lemma 46, we see that we can add this edge as long as

$$n - 2\ell + 1 \geq \frac{2n\ell - (3\ell-1)\ell}{3\ell+1}.$$

We can rearrange this to get the inequality

$$(n-2\ell+1)(3\ell+1) \geq 2n\ell - 3\ell^2 - \ell.$$

Multiplying out and collecting terms, we get a quadratic in ℓ:

$$0 \geq 3\ell^2 - (n-2)\ell - (n+1).$$

This inequality holds as long as

$$\ell \leq \frac{n-2+\sqrt{(n-2)^2+12(n+1)}}{6},$$

since the other root is negative. But $(n-2)^2 + 12(n+1) = (n+4)^2$ so we only need

$$\ell \leq \frac{n-2+n+4}{6} = \frac{n+1}{3},$$

which is true since $n \geq 3\ell + 1$ for $\ell \leq 2^{k-1} - 1$.

Now we must do the same calculation to show that we can add the edge from ℓ to $2\ell+1$. In this case, the fundamental cycle will have length $|C_T(e)| = 3\ell + 2$, the smallest flow on the cycle is on the edge into the vertex at distance $2\ell + 1$ from r, with value $n - 2\ell$, and the imbalance is the sum of the flow values between the vertices at distance ℓ and at distance

2ℓ:

$$f(C_T^+(e)) - f(C_T^-(e)) = n(\ell+1) - \frac{3\ell(\ell+1)}{2}.$$

Plugging this into Lemma 46, we need to verify that

$$n - 2\ell \geq \frac{2n(\ell+1) - 3\ell(\ell+1)}{3\ell+2}.$$

Multiplying through, we get the equivalent inequality

$$(3\ell+2)(n-2\ell) \geq 2n(\ell+1) - 3\ell(\ell+1).$$

Now we collect the coefficients of n to get the inequality

$$(3\ell+2-2\ell-2)n \geq 6\ell^2 + 4\ell - 3\ell^2 - 3\ell)\ell.$$

The coefficient of n is exactly i, so canceling this factor, we reduce to the inequality $n \geq 3\ell+1$, which we have already verified by the choice of n and range on ℓ. □

This shows that we can get $2/3$ of the vertices into a complete binary tree. It is left to show that the remaining vertices can be partitioned into disjoint paths of lengths $\log\log n$ ending at the leaves of this binary tree. To do this, we decompose the rest of the graph into a sequence of complete bipartite graphs.

Proposition 8 *Let G be a bipartite graph with bipartition $V(G) = A \cup B$ and decompose $A = \bigcup_{i=1}^{k} A_i$ and $B = \bigcup_{i=1}^{k} B_i$ such that $|A_i| = |B_i|$ for all $i \in [k]$, the pairs (A_i, B_{i+1}) and (B_i, A_{i+1}) induce complete bipartite graphs for $i \in [k-1]$ and $|A_i| \leq |A_{i+1}|$ for $i \in [k-1]$. Then the vertices of G can be partitioned into paths of length at most k where each path has one endpoint in either A_k or B_k.*

Proof Starting with the vertices in $A_1 \cup B_1$ we greedily pick vertices which are not on a path and use the complete bipartite subgraphs to construct a path of length at most k to A_k.

The condition that $|A_i| \leq |A_{i+1}|$ guarantees that we can never get stuck. $\square$

We start with A_1 and B_1 as the singleton sets consisting the vertices that are furthest from the root r. We can construct blocks as in the above proposition by setting A_2, and B_2 to be all vertices which can have an edge to either B_1 or A_1, and attempt to continue this process. We need to show that the size of the blocks grows fast enough that we only need $O(\log \log n)$ such blocks to reach the top $2/3$ of vertices, while maintaining the property that $|A_i| \leq |A_{i+1}|$.

Lemma 48 *Let T be a tree with root r, two paths of length n and unit demands. There is a constant $\alpha \in (0,1)$ such that for all $\epsilon \in (0,1)$ any vertex whose distance from r is in $[(1-\epsilon)n, (1-\epsilon^\alpha)n)$ can be made adjacent to all vertices in the opposite path whose distance from r is in $\left[(1-\epsilon^\alpha)n, (1-\epsilon^{\alpha^2})n\right)$ while keeping T locally optimal. In particular we can select $\alpha = 3/4$.*

Proof Let e be an edge between a vertex whose distance from r is in $[(1-\epsilon)n, (1-\epsilon^\alpha)n)$ and a vertex on the opposite path whose distance from r is in $[(1-\epsilon^\alpha)n, (1-\epsilon^{\alpha^2})n)$. We will show that e satisfies the condition of Lemma 46.

Among all such edges, the smallest the lowest flow edge on $C_T(e)$ can be is at least $\epsilon n + 1 \geq \epsilon n$, the smallest the length of the cycle can be is $(2 - \epsilon^\alpha - \epsilon^{\alpha^2})n + 1 \geq (2 - \epsilon^\alpha - \epsilon^{\alpha^2})n$, and the largest the imbalance can be is at most $\epsilon^{\alpha^2} n \cdot (\epsilon^{\alpha^2} - \epsilon)n$. Plugging these worst case values into Lemma 46, it suffices to show that

$$(\epsilon n)(2 - \epsilon^\alpha - \epsilon^{\alpha^2})n \geq 2(\epsilon^{\alpha^2} n)(\epsilon^{\alpha^2} - \epsilon)n.$$

We can divide both sides by n^2 to get an inequality that is independent of n

$$\epsilon(2 - \epsilon^\alpha - \epsilon^{\alpha^2}) \geq 2\epsilon^{\alpha^2}(\epsilon^{\alpha^2} - \epsilon).$$

Expanding and collecting terms on one side, it suffices to show that

$$2\epsilon - \epsilon^{\alpha+1} - \epsilon^{\alpha^2+1} - 2\epsilon^{2\alpha^2} + 2\epsilon^{\alpha^2+1} \geq 0.$$

If we set $\alpha = 3/4$ we are trying to prove that

$$2\epsilon - \epsilon^{7/4} + \epsilon^{25/16} - 2\epsilon^{18/16} \geq 0.$$

We make the substitution $x = \epsilon^{1/16}$ to reduce to the equivalent polynomial inequality

$$2x^{16} - x^{28} + x^{25} - 2x^{18} \geq 0.$$

This polynomial can be partially factored as

$$2x^{16} - x^{28} + x^{25} - 2x^{18} = x^{16}(1-x)(x^{11} + x^{10} + x^9 + 2x + 2),$$

which is clearly positive for any $x \in (0, 1)$. □

For notation, let $A(\epsilon)$ and $B(\epsilon)$ denote the sets of vertices whose distance from r is in the interval $[(1-\epsilon)n, (1-\epsilon^\alpha)n)$. We will use this to carefully construct our decomposition. Let A_1 and B_1 again be the singleton sets consisting of the vertices at distance n from the root. We then set $A_2 = A(1/n)$ and $B_2 = B(1/n)$. Inductively, if $A_i = A(\epsilon)$ we set $A_{i+1} = A(\epsilon^\alpha)$. Note that $|A(\epsilon)| = (\epsilon^\alpha - \epsilon)n$. The value $\epsilon^\alpha - \epsilon$ is only increasing in ϵ until ϵ reaches some small which depends on the exponent α. When $\alpha = 3/4$ the maximum of $\epsilon^{3/4} - \epsilon$ is attained at $(3/4)^{1/4} \approx 0.316$. As α tends to 1 the point at which the maximum is attained tends to $1/e$. To avoid these complications, we can simply stop the inductive process once $|A_i| \geq n/100$. This will occur after for $i = O(\log \log n)$. After this, we just set A_{i+1} to be the next $|A_i|$ vertices, until A_{i+1} is completely within the top $2/3$ vertices. This can only take at most 33 extra steps, which in any case gives paths of length $\log \log n$.

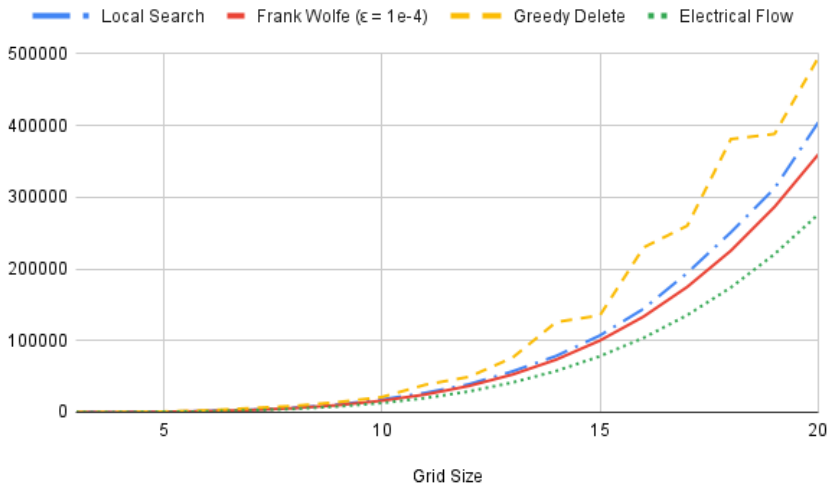


Figure 6.4: Solution values for selected algorithms on an $n \times n$ grid

Remark 5 *It is worth noting that the example graph obtained by adding only the edges from the binary tree to the two-path tree has maximum degree at most* 5 *and is planar.*

6.4.4 Summary of Computational Results

We can compare the performance of the four heuristics in terms of their running time and the upper and lower bounds they provide. The local search algorithm is completely combinatorial, and was implemented in Java. The remaining algorithms use linear system solvers as a subroutine, so were implemented in MATLAB.

To start, we tested the algorithms on $n \times n$ grids for n ranging from 3 to 20. In Figure 6.4 we can see the solution values grow with the size of the instance. On a complete grid we know that the optimal integral solution has objective value $\Theta(n^4 \log n)$ [48], which fits nicely between the local-search upper bound and Frank-Wolfe lower bound.

In Figure 6.5 we can see how the running times grow with the instance size. Because the linear-algebraic algorithms rely on MATLAB's linear system solvers, the running times are affected by various optimizations and heuristics, so can be slightly unpredictable. The combinatorial local-search algorithm consistently finishes in under a second.

Next, we investigate the behavior on larger instances. For the next tests, we start with

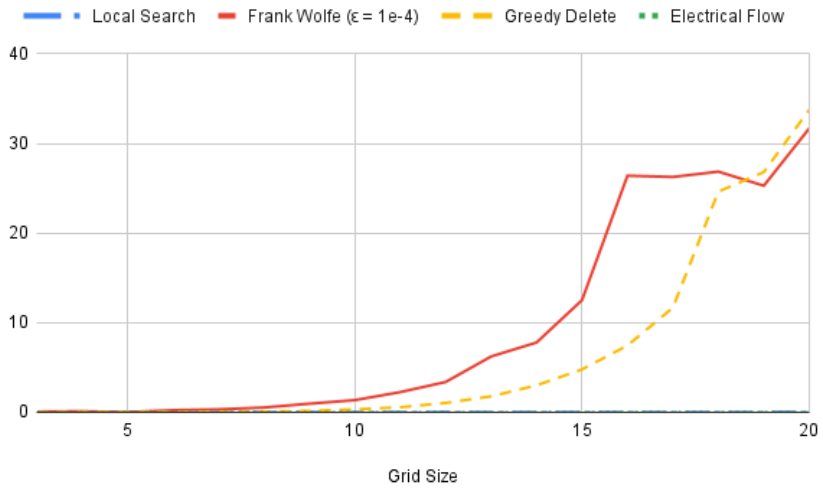


Figure 6.5: Runtime (in seconds) for selected algorithms on an $n \times n$ grid

a 30×30 grid with 900 vertices and 1740 edges, and subsample the edges so they are included with probability p. The solution values as we vary p are recorded in Figure 6.6, while the running times are recorded in Figure 6.7.

We can see that greedy delete starts to perform slightly better than local search once p is at most 0.7, but the running times remain drastically different. Since the performance of the local search depends on the initial tree it may still be preferable to repeat local search with multiple starting points.

From an efficiency perspective, we can see that the algorithms which require solving a linear system in each iteration are much slower, with an additional increase in running time as the graph becomes denser. Unsurprisingly, the electrical flow lower bound, which only solves a single linear system, is extremely fast, consistently finishing in under a second. Similarly, the combinatorial local search algorithm has consistent fast performance, even as the density changes.

6.5 Conclusion and Open Directions

While the network reconfiguration problem is NP-hard even with uniform demands, we have no hardness of approximation. It remains to find an algorithm which beat the naive

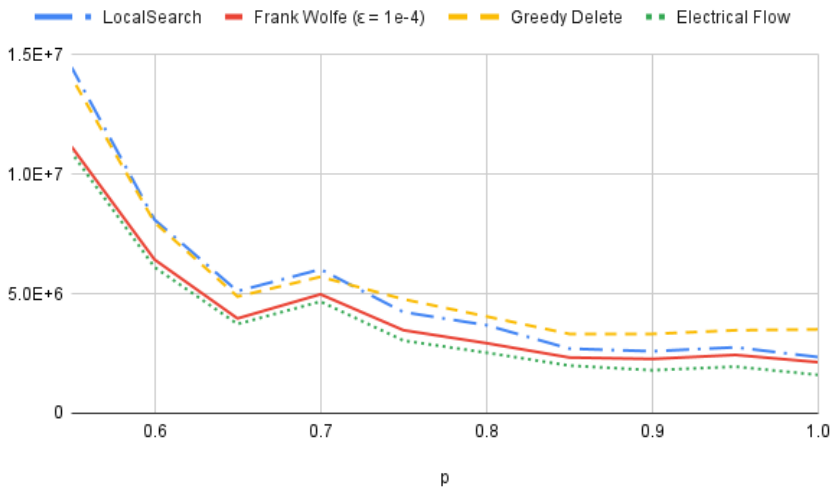


Figure 6.6: Solution values for selected algorithms on a 30×30 grid where each edge is samples with probability p

Figure 6.7: Runtime (in seconds) for selected algorithms on a 30×30 grid where each edge is samples with probability p

$O(n)$-approximation, or show that it is hard to do so. We have shown that the local search algorithm cannot perform much better that $O(n)$-approximation, although it continues to do well in practice.

A related questions is to understand the integrality gap for (CP-Reconfiguration) and devise a tighter rounding algorithm. While the integrality gap is linear in the general case, this requires demands which are not uniform.

Appendices

APPENDIX A

WEIGHTED NASH SOCIAL WELFARE ROUNDING ANALYSIS

A.1 Pruning Small Items

In this section, we prove Lemma 27 by establishing some properties of the set of (support restricted) optimal solutions of (CVX-Weighted) in Claim A.1.1 and Claim A.1.2.

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.

Claim A.1.1 *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* $(\mathbf{b}, \mathbf{q})$ *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 Claim A.1.2, 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.

Claim A.1.2 *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 Claim A.1.1 and Claim A.1.2, we use them to prove Lemma 27.

Proof of Lemma 27 By Claim A.1.2, 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 Claim A.1.1, the difference in objective from $(\mathbf{b}^\star, \mathbf{q}^\star)$ to $(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}})$ is bounded as follows

$$\begin{aligned} f_{\text{cvx}}(\mathbf{b}^\star, \mathbf{q}^\star) - f_{\text{cvx}}(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}}) \\ = \sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} w_i \, b_{ij}^{\text{pruned}} \log \left(\frac{\sum_{i' \in \mathcal{A}} w_{i'} \, b_{i'j}^{\text{pruned}}}{\sum_{i' \in \mathcal{A}} w_{i'} \, b_{i'j}^\star} \right). \end{aligned}$$

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).

$$f_{\text{cvx}}(\mathbf{b}^\star, \mathbf{q}^\star) - f_{\text{cvx}}(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}}) \leq \sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} w_i \, b_{ij}^{\text{pruned}} \log 2. \tag{A.1}$$

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 (A.1) completes the proof. □

In the rest of this section, we will provide proofs of the two component lemmas. The proof of Claim A.1.1 consists of considering the Lagrangian relaxation to investigate the optimality conditions.

Proof of Claim A.1.1 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

$$w_i \log v_{ij} = w_i + w_i \log \left(\sum_{i'\in\mathcal{A}} w_{i'} b^\star_{i'j}\right) + \lambda_i + \eta_j. \tag{A.2}$$

Now, expanding the difference between the two function values, we get

$$\begin{aligned} f_{\text{cvx}}(\mathbf{b}^\star,\mathbf{q}^\star) - f_{\text{cvx}}(\mathbf{b},\mathbf{q}) = & \sum_{i\in\mathcal{A}} \sum_{j\in\mathcal{G}} \left(b^\star_{ij} - b_{ij}\right) \cdot w_i \log v_{ij} \\ & - \sum_{j\in\mathcal{G}} \sum_{i\in\mathcal{A}} w_i\, b^\star_{ij} \log \left(\sum_{i'\in\mathcal{A}} w_{i'}\, b^\star_{i'j}\right) \\ & + \sum_{j\in\mathcal{G}} \sum_{i\in\mathcal{A}} w_i\, b_{ij} \log \left(\sum_{i'\in\mathcal{A}} w_{i'}\, b_{i'j}\right). \end{aligned} \tag{A.3}$$

Substituting the value of v_{ij} from equation (A.2) in equation (A.3) gives

$$
\begin{aligned}
&f_{\text{cvx}}(\mathbf{b}^\star, \mathbf{q}^\star) - f_{\text{cvx}}(\mathbf{b}, \mathbf{q}) \\
&\qquad = \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{G}} (b^\star_{ij} - b_{ij}) \left(w_i \log \left(\sum_{i' \in \mathcal{A}} w_{i'}\, b^\star_{i'j} \right) + \lambda_i + w_i + \eta_j \right) \\
&\qquad\quad - \sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} w_i\, b^\star_{ij} \log \left(\sum_{i' \in \mathcal{A}} w_{i'}\, b^\star_{i'j} \right) \\
&\qquad\quad + \sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} w_i\, b_{ij} \log \left(\sum_{i' \in \mathcal{A}} w_{i'}\, b_{i'j} \right) \\
&\qquad = \sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} w_i\, b_{ij} \log \left(\frac{\sum_{i' \in \mathcal{A}} w_{i'}\, b_{i'j}}{\sum_{i' \in \mathcal{A}} w_{i'}\, b^\star_{i'j}} \right) \\
&\qquad\quad + \sum_{i \in \mathcal{A}} (\lambda_i + w_i) \left(\sum_{j \in \mathcal{G}} b^\star_{ij} - \sum_{j \in \mathcal{G}} b_{ij} \right) \\
&\qquad\quad + \sum_{j \in \mathcal{G}} \eta_j \left(\sum_{i \in \mathcal{A}} b^\star_{ij} - \sum_{i \in \mathcal{A}} b_{ij} \right).
\end{aligned}
$$

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

$$
f_{\text{cvx}}(\mathbf{b}^\star, \mathbf{q}^\star) - f_{\text{cvx}}(\mathbf{b}, \mathbf{q}) = \sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} w_i\, b_{ij} \log \left(\frac{\sum_{i' \in \mathcal{A}} w_{i'}\, b_{i'j}}{\sum_{i' \in \mathcal{A}} w_{i'}\, b^\star_{i'j}} \right).
$$

□

It remains to verify Claim A.1.2. Notice that the statement of the claim can be interpreted purely as a feasibility problem where the extra conditions we desire are additional linear constraints. We will show that such a feasible solution exists by constructing an auxiliary flow problem, and verifying Hoffman's circulation condition.

Proof of Claim A.1.2 Let $(\mathbf{b}, \mathbf{q})$ be our initial acyclic feasible point in $\mathcal{P}(\mathcal{A}, \mathcal{G})$. Let F be a directed forest formed by $G_{\text{supp}}(\mathbf{b})$ when every tree is rooted at an arbitrary agent node. We want to show that there exists another feasible solution $(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}})$ such that $G_{\text{supp}}(\mathbf{b}^{\text{pruned}})$ is a subgraph of $G_{\text{supp}}(\mathbf{b})$, and with the following additional constraints:

- The
- $q_j \leq q_j^{\text{pruned}}$ for each item j with $q_j \geq 1/2$,
- each item with $q_j < 1/2$ is a leaf in $G_{\text{supp}}(\mathbf{b}^{\text{pruned}})$ connected to its parent in F, and
- for any $(i, j) \in \mathcal{A} \times \mathcal{G}$, $b_{ij}^{\text{pruned}} \leq \min\{1, 2 \cdot b_{ij}\}$.

To do this we will construct an auxiliary flow problem with lower- and upper-bound capacities in the edges, such that any feasible circulation corresponds to feasible solution with the desired properties.

The flow graph we construct will have two additional vertices: a super-source s and a super-sink t. The node s will be connected to all agents $i \in \mathcal{A}$, we will have all edges from $\mathcal{A}$ to $\mathcal{G}$, and then every item $j \in \mathcal{G}$ will be connected to t. We also add an additional edge from t to s to form our circulation problem.

To enforce the desired constraints, we must partition the items into two subsets

$$G_1 = \{j \in \mathcal{G} : q_j \geq 1/2\}, \quad \text{and}$$
$$G_2 = \{j \in \mathcal{G} : q_j < 1/2\}.$$

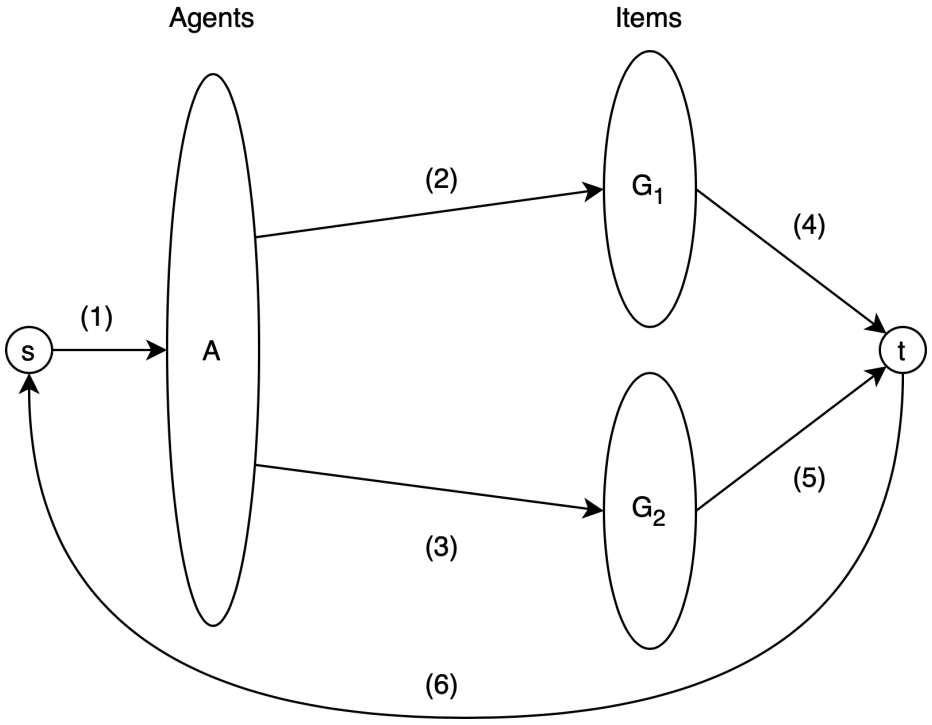


Figure A.1: Graph Skeleton for the Circulation Problem

Thus, the graph can be decomposed as in Figure A.1, and there are six types of edges. We will provide lower-bound capacities ℓ_e and upper-bound capacities u_e on each class of edges separately.

(1) For edges between s and $i \in \mathcal{A}$ set $\ell_{s,i} = 0$, and $u_{s,i} = 1$;

(2) for edges between $i \in \mathcal{A}$ and $j \in G_1$ set $\ell_{i,j} = 0$, and $u_{i,j} = \min\{1, 2b_{ij}\}$;

(3) for edges between $i \in \mathcal{A}$ and $j \in G_2$ set $\ell_{i,j} = 0$, and

$$u_{i,j} = \begin{cases} \min\{1, 2b_{ij}\} & \text{if } i \text{ is the parent of } j, \\ 0 & \text{otherwise;} \end{cases}$$

(4) for edges between $j \in G_1$ and t set $\ell_{j,t} = q_j$, and $u_{j,t} = 1$;

(5) for edges between $j \in G_2$ and t set $\ell_{j,t} = 0$, and $u_{j,t} = 1$; and

(6) for the edge from t to s set $\ell_{t,s} = |\mathcal{A}|$ and $u_{t,s} = \infty$.

Given a feasible circulation with these capacities, we can take the variables for the edges between $i \in \mathcal{A}$ and $j \in \mathcal{G}$ and it will be a feasible solution which satisfies the additional desired constraints.

Thus, it remains to show that there is a feasible circulation. To do this, we will use Hoffman's circulation condition [90], which states that a circulation exists if and only if for every subset C of the vertices $\ell(\delta^{\text{in}}(C)) \leq u(\delta^{\text{out}}(C))$. We will divide into four cases based on whether the vertices s and t are in the set C.

We begin by eliminating two simple cases. If $s \notin C$ and $t \notin C$, then $\ell(\delta^{\text{in}}(C)) = 0$, since the only edges with positive lower bounds are incoming to s or incoming to t. Since all the upper bounds are non-negative, the condition holds in this case. Second, if $s \notin C$ and $t \in C$, then $u(\delta^{\text{out}}(C)) = \infty$, since the edge from t to s is an outgoing edge. In this case the condition is always satisfied.

Next, consider the case that $s \in C$ and $t \in C$. Let $S = C \cap A$, $T_1 = C \cap G_1$ and $T_2 = C \cap G_2$. Then

$$\ell(\delta^{\text{in}}(C)) = \sum_{j \in G_1 \setminus T_1} q_j, \quad \text{and}$$

$$u(\delta^{\text{out}}(C)) = |\mathcal{A}| - |S| + \sum_{i \in S} \sum_{j \in G_1 \setminus T_1} \min\{1, 2b_{ij}\} + \sum_{i \in S} \sum_{\substack{j \in G_2 \setminus T_2 \\ j \text{ child of } i}} \min\{1, 2b_{ij}\}.$$

Notice that adding moving elements from $G_2 \setminus T_2$ to T_2 only decreases the upper bound, without changing the lower bound. So we may assume that $T_2 = G_2$. Then

$$u(\delta^{\text{out}}(C)) = |\mathcal{A}| - |S| + \sum_{i \in S} \sum_{j \in G_1 \setminus T_1} \min\{1, 2b_{ij}\}.$$

For a fixed set T_1 this upper bound is minimized when

$$S = \left\{ i \in \mathcal{A} : \sum_{j \in G_1 \setminus T_1} \min\{1, 2b_{ij}\} < 1 \right\}.$$

Thus, for all $i \in S$ and $j \in G_1 \backslash T_1$ we know that $\min\{1, 2b_{ij}\} = 2b_{ij} < 1$. So we can rewrite

$$\ell(\delta^{\text{in}}(C)) = \sum_{j \in G_1 \backslash T_1} \sum_{i \in \mathcal{A}} b_{ij}, \quad \text{and}$$
$$u(\delta^{\text{out}}(C)) = n - |S| + \sum_{i \in S} \sum_{j \in G_1 \backslash T_1} 2b_{ij}.$$

Thus, we can see that

$$\begin{aligned} u(\delta^{\text{out}}(C)) - \ell(\delta^{\text{in}}(C)) &= |\mathcal{A}| - |S| + \sum_{j \in G_1 \backslash T_1} \left(\sum_{i \in S} b_{ij} - \sum_{i \in \mathcal{A} \backslash S} b_{ij} \right) \\ &= |\mathcal{A} \backslash S| - \sum_{i \in \mathcal{A} \backslash S} \sum_{j \in G_1 \backslash T_1} b_{ij} + \sum_{i \in S} \sum_{j \in G_1 \backslash T_1} b_{ij} \\ &\geq \sum_{i \in S} \sum_{j \in G_1 \backslash T_1} b_{ij} \geq 0, \end{aligned}$$

where in the last line we use the fact that $\sum_{j \in G_1 \backslash T_1} b_{ij} \leq \sum_{j \in \mathcal{G}} b_{ij} = 1$ for all $i \in \mathcal{A}$.

Finally, we are left with the case that $s \in C$ and $t \notin C$. Again, let $S = C \cap A$, $T_1 = C \cap G_1$ and $T_2 = C \cap G_2$. Then

$$\begin{aligned} \ell(\delta^{\text{in}}(C)) &= |\mathcal{A}|, \quad \text{and} \\ u(\delta^{\text{out}}(C)) &= |\mathcal{A}| - |S| + \sum_{i \in S} \sum_{j \in G_1 \backslash T_1} \min\{1, 2b_{ij}\} \\ &\quad + \sum_{i \in S} \sum_{\substack{j \in G_2 \backslash T_2 \\ j \text{ child of } i}} \min\{1, 2b_{ij}\} + |T_1| + |T_2|. \end{aligned}$$

Consider removing an item j from T_2. Then $\sum_{i \in S} \sum_{\substack{j \in G_2 \backslash T_2 \\ j \text{ child of } i}} \min\{1, 2b_{ij}\}$ only gains a single term corresponding to the parent of j in the rooted tree (if it is in S), so the increase is at most 1. the $|T_2|$ term would decrease by 1 so the net decrease is non-negative, and we may assume that $T_2 = \emptyset$.

For a fixed set T_1, the upper bound is minimized when

$$S = \left\{ i \in \mathcal{A} : \sum_{j \in G_1 \setminus T_1} \min\{1, 2b_{ij}\} + \sum_{i \in S} \sum_{\substack{j \in G_2 \\ j \text{ child of } i}} \min\{1, 2b_{ij}\} < 1 \right\},$$

so again we can assume $\min\{1, 2b_{ij}\} = 2b_{ij}$ for all $i \in S$ and $j \in G_1 \setminus T_1$ or $j \in G_2$ is a child of i. Using this, we can write

$$\begin{aligned} & u(\delta^{\text{out}}(C)) - \ell(\delta^{\text{in}}(C)) \\ & \qquad = \sum_{i \in S} \left(\sum_{j \in G_1 \setminus T_1} 2b_{ij} + \sum_{\substack{j \in G_2 \\ j \text{ child of } i}} 2b_{ij} - \sum_{j \in \mathcal{G}} b_{ij} \right) + |T_1| \\ & \qquad = \sum_{i \in S} \left(\sum_{j \in G_1 \setminus T_1} b_{ij} + \sum_{\substack{j \in G_2 \\ j \text{ child of } i}} b_{ij} - \sum_{j \in T_1} b_{ij} - \sum_{\substack{j \in G_2 \\ j \text{ parent of } i}} b_{ij} \right) + |T_1|. \end{aligned}$$

We will show that this is non-negative attributing each term of the outer sum as charge to its agent $i \in S$, and then analyzing the total charge within the components of F induced by S and T_1. Consider one such component, which is comprised of agents in $\tilde{S} \subseteq S$ and items $\tilde{T}_1 \subseteq T_1$. The total charge of this component is then

$$\sum_{i \in \tilde{S}} \left(\sum_{j \in G_1 \setminus \tilde{T}_1} b_{ij} + \sum_{\substack{j \in G_2 \\ j \text{ child of } i}} b_{ij} - \sum_{j \in \tilde{T}_1} b_{ij} - \sum_{\substack{j \in G_2 \\ j \text{ parent of } i}} b_{ij} \right) + |\tilde{T}_1|.$$

We will show that this is non-negative. First, notice that the charge for each agent $i \in \tilde{S}$ is

$$\sum_{j \in G_1 \setminus \tilde{T}_1} b_{ij} + \sum_{\substack{j \in G_2 \\ j \text{ child of } i}} b_{ij} - \sum_{j \in \tilde{T}_1} b_{ij} - \sum_{\substack{j \in G_2 \\ j \text{ parent of } i}} b_{ij} \geq - \sum_{j \in \mathcal{G}} b_{ij} = -1,$$

so if $|\tilde{T}_1| \geq |S|$ we can immediately see that the total charge is non-negative.

So now consider the case that our component has $|\tilde{T}_1| < |S|$. Since we have a root in the full tree T, we can also identify a root of our component $F[\tilde{S} \cup \tilde{T}_1]$. First, we will consider the case that the root of the component is an agent in $\tilde{S}$. We will show that

$$\sum_{i \in \tilde{S}} \left(\sum_{j \in \mathcal{G} \setminus \tilde{T}_1} b_{ij} - \sum_{j \in \tilde{T}_1} b_{ij} \right) + |\tilde{T}_1| \geq 1.$$

Note that the only difference between this and the actual total charge is that we may have replaced the term $-b_{ij}$ for $j \in G_2$ the parent of i with $+b_{ij}$. But this can only happen for one vertex in the component, namely the root $r$if its parent $p(r)$ happen to be in G_2. Since $p(r) \in G_2$ we know that $b_{r,p(r)} \leq 1/2$, so the total true charge of the component would be

$$\sum_{i \in \tilde{S}} \left(\sum_{j \in \mathcal{G} \setminus \tilde{T}_1} b_{ij} - \sum_{j \in \tilde{T}_1} b_{ij} \right) + |\tilde{T}_1| - 2b_{r,p(r)} \geq 1 - 2(1/2) = 0.$$

It remains to verify the inequality. Indeed,

$$\begin{aligned} \sum_{i \in \tilde{S}} \left(\sum_{j \in \mathcal{G} \setminus \tilde{T}_1} b_{ij} - \sum_{j \in \tilde{T}_1} b_{ij} \right) + |\tilde{T}_1| &= \sum_{i \in \tilde{S}} \left(\sum_{j \in \mathcal{G}} b_{ij} \right) - 2 \sum_{j \in \tilde{T}_1} \left(\sum_{i \in \tilde{S}} b_{ij} \right) + |\tilde{T}_1| \\ &\geq |\tilde{S}| - 2|\tilde{T}_1| + |\tilde{T}_1| \geq 1, \end{aligned}$$

where in the last inequality we use the fact that $|\tilde{T}_1| < |S|$. Finally, consider the case that the root of our component $F[\tilde{S} \cup \tilde{T}_1]$ is an item $r \in \tilde{T}_1$. Then after deleting r from the tree we get a collection of components rooted at items. Each such component gives us a modified charge of at least 1, as before, but we must subtract off

$$2 \sum_{i \in \tilde{S}} b_{i,r}$$

to get the true charge. This time it is possible that the terms $b_{i,r}$ are greater than $1/2$, but we can cancer one copy of $b_{i,r}$ with the leftover charge of at least 1 from the component rooted

at i, and the remaining $-\sum_{i\in\tilde{S}} b_{i,r}$ can be canceled with the extra $+1$ we get from the fact that $r \in |\tilde{T}_1|$. $\square$

A.2 Fractional Matching and Analysis

In this section, we prove Lemma 28, which completes the proof of Theorem 25.

We establish Lemma 28 by proving two inequalities (in Claims A.2.1 and A.2.2) about the properties of f_{ncvx} at any feasible point whose support is a forest. Claim A.2.1 shows that f_{ncvx} can be upper bounded by a linear function in $\mathbf{b}$ while only losing a constant factor.

Claim A.2.1 *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*

$$\sum_{i\in\mathcal{A}} w_i \left(\sum_{j\notin S(i)} b_{ij} \log v_{ij} + \sum_{j\in S(i)} b_{ij} \log \left(\sum_{j\in S(i)} v_{ij} \right) \right) \geq f_{\text{ncvx}}(\mathbf{b}, \mathbf{q}) - \log 2 - \frac{1}{2e}.$$

Claim A.2.2 demonstrates how the linear function obtained in Claim A.2.1 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.

Claim A.2.2 *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 $G_{\text{support}}(\mathbf{b})$. 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 $\mathcal{G}\backslash(\cup_i S(i))$ *such that*

$$\sum_{i\in\mathcal{A}} w_i \log\left(v_{iM(i)} + \sum_{j\in S(i)} v_{ij}\right) \geq \sum_{i\in\mathcal{A}} w_i \left(\sum_{j\notin S(i)} b_{ij}\log v_{ij} + \sum_{j\in S(i)} b_{ij}\log\left(\sum_{j\in S(i)} v_{ij}\right)\right), \quad \text{(A.4)}$$

where $v_{iM(i)} = 0$ *if agent* i *is not matched in* M.

Claim A.2.1 and Claim A.2.2 together establish Lemma 28.

Proof of Lemma 28 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 $i \in \mathcal{A}$, $S(i)$ is a subset of the leaf node adjacent to agent i in $G_{\text{supp}}(\mathbf{b})$, and $S(i)$ contains all children of agent i with $q_j < 1/2$.

By Claim A.2.2 there exists a matching M in $G_{\text{supp}}(\mathbf{b})$ between vertices in $\mathcal{A}$ and $\mathcal{G}\backslash(\cup_i S(i))$ such that

$$\sum_{i\in\mathcal{A}} w_i \log\left(v_{iM(i)} + \sum_{j\in S(i)} v_{ij}\right) \geq \sum_{i\in\mathcal{A}} w_i \left(\sum_{j\notin S(i)} b_{ij}\log v_{ij} + \sum_{j\in S(i)} b_{ij}\log\left(\sum_{j\in S(i)} v_{ij}\right)\right).$$

Now by Claim A.2.1

$$\sum_{i\in\mathcal{A}} w_i \left(\sum_{j\notin S(i)} b_{ij}\log v_{ij} + \sum_{j\in S(i)} b_{ij}\log\left(\sum_{j\in S(i)} v_{ij}\right)\right) \geq f_{\text{ncvx}}(\mathbf{b}, \mathbf{q}) - \log 2 - \frac{1}{2e}.$$

Combining these two inequalities proves the lemma. □

In the rest of this section, we provide the proofs of Claim A.2.1 and Claim A.2.2.

The following lemma is an application of Gibbs's inequality and is used in the proof of

Claim A.2.1.

Lemma 49 *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 Claim A.2.1 Let $S := \cup_i S(i)$. Recall that

$$\begin{aligned} f_{\mathrm{ncvx}}(\mathbf{b}, \mathbf{q}) &= \sum_{i \in \mathcal{A}} w_i \sum_{j \in \mathcal{G}} b_{ij} \log v_{ij} - \sum_{i \in \mathcal{A}} w_i \sum_{j \in \mathcal{G}} b_{ij} \log q_j \\ &= \sum_{i \in \mathcal{A}} w_i \sum_{j \notin S(i)} b_{ij} \log v_{ij} - \sum_{i \in \mathcal{A}} w_i \sum_{j \notin S(i)} b_{ij} \log q_j \\ &\quad + \sum_{i \in \mathcal{A}} w_i \sum_{j \in S(i)} \left(b_{ij} \log v_{ij} - b_{ij} \log b_{ij} \right), \end{aligned} \tag{A.5}$$

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,

$$- \sum_{i \in \mathcal{A}} w_i \, b_{ij} \log q_j \leq \log 2 \sum_{i \in \mathcal{A}} w_i \, b_{ij}. \tag{A.6}$$

Plugging this bound into equation (A.5) gives

$$\begin{aligned} f_{\mathrm{ncvx}}(\mathbf{b}, \mathbf{q}) \leq & \sum_{i \in \mathcal{A}} w_i \sum_{j \notin S(i)} b_{ij} \log v_{ij} + \log 2 \sum_{i \in \mathcal{A}} w_i \sum_{j \notin S(i)} b_{ij} \\ & + \sum_{i \in \mathcal{A}} w_i \left(\sum_{j \in S(i)} b_{ij} \log v_{ij} - b_{ij} \log b_{ij} \right). \end{aligned} \tag{A.7}$$

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 (A.7) yields

$$\begin{aligned} f_{\text{ncvx}}(\mathbf{b}, \mathbf{q}) &\leq \sum_{i \in \mathcal{A}} w_i \sum_{j \notin S(i)} b_{ij} \log v_{ij} + \log 2 \sum_{i \in \mathcal{A}} w_i \\ &\quad + \sum_{i \in \mathcal{A}} w_i \left(\sum_{j \in S(i)} b_{ij} \log v_{ij} - b_{ij} \log b_{ij} - b_{ij} \log 2 \right) \\ &= \sum_{i \in \mathcal{A}} w_i \sum_{j \notin S(i)} (b_{ij} \log v_{ij}) + \log 2 \\ &\quad + \sum_{i \in \mathcal{A}} w_i \left(\sum_{j \in S(i)} b_{ij} \log v_{ij} - b_{ij} \log b_{ij} - b_{ij} \log 2 \right), \end{aligned} \tag{A.8}$$

where the last equation follows from $\sum_i w_i = 1$.

For each agent $i \in \mathcal{A}$, Corollary 49 implies that

$$\begin{aligned} &\sum_{j \in S(i)} b_{ij} \log v_{ij} - b_{ij} \log b_{ij} \\ &\qquad \leq \sum_{j \in S(i)} b_{ij} \log \left(\sum_{j \in S(i)} v_{ij} \right) - \sum_{j \in S(i)} b_{ij} \log \left(\sum_{j \in S(i)} b_{ij} \right). \end{aligned}$$

So, for any agent i,

$$\begin{aligned} &\sum_{j \in S(i)} b_{ij} \log v_{ij} - b_{ij} \log b_{ij} - b_{ij} \log 2 \\ &\qquad \leq \sum_{j \in S(i)} b_{ij} \log \left(\sum_{j \in S(i)} v_{ij} \right) - \sum_{j \in S(i)} b_{ij} \log \left(\sum_{j \in S(i)} b_{ij} \right) - \sum_{j \in S(i)} b_{ij} \log 2 \\ &\qquad \leq \sum_{j \in S(i)} b_{ij} \log \left(\sum_{j \in S(i)} v_{ij} \right) + \frac{1}{2e}, \end{aligned} \tag{A.9}$$

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 (A.9) in equation (A.8), 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 Proof of Claim A.2.2 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:

$$\begin{aligned} x_{ij} &:= b_{ij} \quad \forall i \in \mathcal{A}, j \in \{\mathcal{G} \backslash L\} \\ x_{i\ell_i} &:= \sum_{j \in S(i)} b_{ij} \quad \forall i \in \mathcal{A}. \end{aligned}$$

The left-hand side of equation (A.4) can be stated in terms of $\mathbf{x}$ as

$$\sum_{i\in\mathcal{A}} w_i \left(\sum_{j\notin S(i)} b_{ij} \log v_{ij} + \sum_{j\in S(i)} b_{ij} \log \left(\sum_{j\in S(i)} v_{ij} \right) \right) = \sum_{i\in\mathcal{A}} \sum_{j\in\widetilde{\mathcal{G}}} x_{ij}\, w_i \log v_{ij}. \quad \text{(A.10)}$$

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

$$\sum_{i\in\mathcal{A}} \sum_{j\in\widetilde{\mathcal{G}}} x_{ij}\, w_i \log v_{ij} \leq \sum_{i\in\mathcal{A}} w_i \log v_{i\widetilde{M}(i)}. \quad \text{(A.11)}$$

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

$$\sum_{i\in\mathcal{A}} w_i \log v_{i\widetilde{M}(i)} \leq \sum_{i\in\mathcal{A}} w_i \log \left(v_{iM(i)} + \sum_{j\in S(i)} v_{ij} \right). \quad \text{(A.12)}$$

Then equations (A.10), (A.11), and (A.12) together imply

$$\sum_{i\in\mathcal{A}} w_i \log \left(v_{iM(i)} + \sum_{j\in S(i)} v_{ij} \right) \geq \sum_{i\in\mathcal{A}} w_i \left(\sum_{j\notin S(i)} b_{ij} \log v_{ij} + \sum_{j\in S(i)} b_{ij} \log \left(\sum_{j\in S(i)} v_{ij} \right) \right).$$

□

A.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 7 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 13 Alternative Algorithm for Rounding an Acyclic Solution

Input. NSW instance $(\mathcal{A}, \mathcal{G}, \mathbf{v}, \mathbf{w})$, solution $(\mathbf{b}, \mathbf{q}) \in \mathcal{P}(\mathcal{A}, \mathcal{G})$

$(\mathbf{b}^\star, \mathbf{q}^\star) \leftarrow$ locally optimal acyclic solution of (NCVX-Weighted) starting from $(\mathbf{b}, \mathbf{q})$

$F^\star \leftarrow G_{\mathrm{supp}}(\mathbf{b}^\star)$ with every tree rooted at an agent node

Remove edges between item j and its children in $F^\star$ whenever $q_j^\star < 1/2$ to get forest $\widetilde{F}$ **(Pruning)**

$L_i^\star \leftarrow$ set of leaf children of agent i in $\widetilde{F}$ and let $L^\star = \cup_i \{L_i^\star\}$

$M^\star \leftarrow$ matching between $\mathcal{A} \to \mathcal{G} \backslash L^\star$ in $\widetilde{F}$ which maximizes weight function $w_{\widetilde{F}}(M) := \sum_{i \in \mathcal{A}} w_i \, \log \left(v_{iM(i)} + \sum_{j \in L_i^\star} v_{ij} \right)$

$\sigma^\star \leftarrow$ assignment of $\mathcal{G}$ to $\mathcal{A}$ with $\sigma^\star(j) = i$ if $j \in \{L_i^\star \cup M^\star(i)\}$ **(Matching)**

Return $\sigma^\star$

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 25 in the analysis.

Theorem 31 *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 7) which returns an assignment $\sigma : \mathcal{G} \to \mathcal{A}$ such that

$$\mathrm{NSW}(\sigma) \geq f_{\mathrm{ncvx}}(\mathbf{b}) - 2\log 2 - \frac{1}{2e}.$$

Note that, other than the running time, this result would imply Theorem 25 as well since Lemma 19 states that $f_{\mathrm{ncvx}}(\mathbf{b}) \geq f_{\mathrm{cvx}}(\mathbf{b}) - D_{\mathrm{KL}}(\mathbf{w} \,||\, \mathbf{u})$.

Looking into the proof of Theorem 25, the only thing which we need to change is to provide an alternative to Claim A.1.1 in terms of the non-convex objective. Claim A.1.1 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 50 *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*

$$f_{\mathrm{ncvx}}(\mathbf{b}^\star, \mathbf{q}^\star) - f_{\mathrm{ncvx}}(\mathbf{b}, \mathbf{q}) \leq \sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} w_i \, b_{ij} \log \left(\frac{\sum_{i \in \mathcal{A}} w_i \, b_{ij}}{\sum_{i \in \mathcal{A}} w_i \, b_{ij}^\star} \right).$$

Before providing a proof of this lemma, we will use it to prove Theorem 31. The proof is essentially identical to the proof of Theorem 25

Proof of Theorem 31 Let $(\mathbf{b}^\star, \mathbf{q}^\star)$ be an acyclic solution of (NCVX-Weighted) which is locally-optimal, let $\widetilde{F}$ denote the forest obtained after pruning $G_{\mathrm{supp}}(\mathbf{b}^\star)$. Let $L_i^\star$ denote the set of leaf children of agent i in $\widetilde{F}$.

Let $(\mathbf{b}^{\mathrm{pruned}}, \mathbf{q}^{\mathrm{pruned}})$ be the pruned feasible solution guaranteed by Claim A.1.2 on input $(\mathbf{b}^\star, \mathbf{q}^\star)$. We are guaranteed that $q_j^\star \leq q_j^{\mathrm{pruned}}$ for each item j with $q_j^\star \geq 1/2$, 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 $(i, j) \in \mathcal{A} \times \mathcal{G}$, $b_{ij}^{\mathrm{pruned}} \leq \min\{1, 2 \cdot b_{ij}^\star\}$. Using Lemma 50 the difference in

objective is bounded

$$f_{\text{ncvx}}(\mathbf{b}^\star, \mathbf{q}^\star) - f_{\text{ncvx}}(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}}) \leq \sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} w_i \, b_{ij}^{\text{pruned}} \log \left(\frac{\sum_{i \in \mathcal{A}} w_i \, b_{ij}^{\text{pruned}}}{\sum_{i \in \mathcal{A}} w_i \, b_{ij}^\star} \right)$$
$$\leq \sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} w_i \, b_{ij}^{\text{pruned}} \log 2 = \log 2$$

Using Lemma 28 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

$$\sum_{i \in \mathcal{A}} w_i \log \left(v_{iM(i)} + \sum_{j \in L_i^\star} v_{ij} \right) = \sum_{i \in \mathcal{A}} w_i \log \left(v_{iM(i)} + \sum_{j \in S(i)} v_{ij} \right)$$
$$\geq f_{\text{ncvx}}(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}}) - \log 2 - \frac{1}{2e}.$$

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 13 satisfies

$$\begin{aligned}
\text{NSW}(\sigma^\star) = \sum_{i \in \mathcal{A}} w_i \log \left(v_{iM^\star(i)} + \sum_{j \in L_i^\star} v_{ij} \right) &\geq \sum_{i \in \mathcal{A}} w_i \log \left(v_{iM(i)} + \sum_{j \in L_i^\star} v_{ij} \right) \\
&\geq f_{\text{ncvx}}(\mathbf{b}^{\text{pruned}}, \mathbf{q}^{\text{pruned}}) - \log 2 - \frac{1}{2e} \\
&\geq f_{\text{ncvx}}(\mathbf{b}^\star, \mathbf{q}^\star) - 2 \log 2 - \frac{1}{2e},
\end{aligned}$$

where the second inequality follows from Lemma 28, and the last inequality follows from our earlier bound on the difference in objective between the two solutions. □

A.3.1 Technical Lemmas

In this section we provide a proof of Lemma 50. 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 51 *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 $\lambda_i, \gamma_j \in \mathbf{R}$ 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 52 *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 51 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

$$\begin{aligned}
&f_{\text{ncvx}}(\mathbf{b}^\star, \mathbf{q}^\star) - f_{\text{ncvx}}(\mathbf{b}, \mathbf{q}) \\
&\quad = \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{G}} w_i (b_{ij}^\star - b_{ij}) \left(\log q_j^\star + \frac{\lambda_i}{w_i} + \frac{\sum_{i \in \mathcal{A}} w_i b_{ij}}{w_i q_j^\star} + \frac{\eta_j}{w_i}\right) \\
&\qquad + \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{G}} w_i b_{ij} \log q_j - w_i b_{ij}^\star \log q_j^\star \\
&\quad = \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{G}} w_i b_{ij} \left(\log q_j - \log q_j^\star\right) + \sum_{i \in \mathcal{A}} \lambda_i \left(\sum_{j \in \mathcal{G}} b_{ij}^\star - \sum_{j \in \mathcal{G}} b_{ij}\right) \\
&\qquad + \sum_{j \in \mathcal{G}} \left(\eta_j + \frac{\sum_{i \in \mathcal{A}} w_i b_{ij}^\star}{q_j^\star}\right) \left(\sum_{i \in \mathcal{A}} b_{ij}^\star - \sum_{i \in A} b_{ij}\right).
\end{aligned}$$

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

$$f_{\text{ncvx}}(\mathbf{b}^\star, \mathbf{q}^\star) - f_{\text{ncvx}}(\mathbf{b}, \mathbf{q}) = \sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{G}} w_i b_{ij} \left(\log q_j - \log q_j^\star\right) + \sum_{j\in\mathcal{G}} \left(\eta_j + \frac{\sum_{i\in\mathcal{A}} w_i b_{ij}^\star}{q_j^\star}\right)\left(q_j^\star - q_j\right).$$

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 50 Using Lemma 52 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

$$f_{\text{ncvx}}(\mathbf{b}^\star, \mathbf{q}^\star) - f_{\text{ncvx}}(\mathbf{b}, \mathbf{q}) \leq \sum_{j\in\mathcal{G}}\sum_{i\in\mathcal{A}} w_i\, b_{ij} \log\left(\frac{\sum_{i\in\mathcal{A}} w_i\, b_{ij}}{\sum_{i\in\mathcal{A}} w_i\, b^\star_{ij}}\right).$$

□

A.3.2 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 [91], and is also similar to the approach used for weakly convex functions [92]. 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 14 Gradient Descent with Bregman Divergence

Input. Functions f, and g such that g and $f + g$ are convex, $\delta > 0$ and initial $x_0 \in K$

$k \leftarrow 0$

while $\|\nabla f(x_i)\| > \delta$ **do**

$x_{k+1} \leftarrow \text{argmin}_{x \in K} f(x) + D_g(x, x_k)$

$k \leftarrow k + 1$

end while

Return x_k.

We can solve the internal minimization problem, since the objective is convex.

Lemma 53 *If f and g are functions such that g and $f + g$ are convex, then for any fixed choice of $y \in \mathbf{R}^n$, 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 convexity. □

After sufficiently many iteration, we are guaranteed to find a point x where $\nabla f(x)$ is close to zero.

Lemma 54 *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 $\rho^* = \min_i \rho_i$.

$$\max_{\mathbf{b}} \quad f_{\text{ncvx}}(\mathbf{b}) := \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{G}} w_i \, b_{ij} \log v_{ij} - \sum_{j \in \mathcal{G}} \sum_{i \in \mathcal{A}} w_i \, b_{ij} \log \left(\sum_{i \in \mathcal{A}} b_{ij} \right)$$

$$\text{s.t.} \quad \sum_{j \in \mathcal{G}} b_{ij} = 1 \quad \forall i \in \mathcal{A}$$

$$\sum_{i \in \mathcal{A}} b_{ij} \leq 1 \quad \forall j \in \mathcal{G}$$

$$b_{ij} \geq 0 \quad \forall (i,j) \in \mathcal{A} \times \mathcal{G}$$

$$\frac{1}{n} \sum_{i \in \mathcal{A}} b_{ij} \geq \frac{1}{n^2} \rho^* \quad \forall (i,j) \in \mathcal{A} \times \mathcal{G}$$

$$\sum_{i \in \mathcal{A}} w_i b_{ij} \geq \frac{\min_i w_i}{n} \rho^* \quad \forall (i,j) \in \mathcal{A} \times \mathcal{G}.$$

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

$$\frac{1}{n} \sum_{i \in \mathcal{A}} b_{ij} = \frac{1}{n} \frac{v_{\sigma(j)j}}{V_{\sigma(j)}} \geq \frac{1}{n^2} \rho^*, \text{ and}$$

$$\sum_{i \in \mathcal{A}} w_i b_{ij} = w_i \frac{v_{\sigma(j)j}}{V_{\sigma(j)}} \geq \frac{\min_i w_i}{n} \rho^*.$$

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 2

$$D_g(b, b') = D_{g^*}(\nabla g(b'), \nabla g(b)).$$

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 3

$$\nabla g^*(x) = (\nabla g)^{-1}(x).$$

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 55 *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,

$$\frac{\partial}{\partial b_{ij}} D_{\mathrm{KL}}(\mu \,||\, \theta) = w_i \log \mu_j(b) + w_i - w_i \log \theta_j(b) - \frac{1}{n} \frac{\mu_j(b)}{\theta_j(b)}$$

$$= w_i \log \frac{\mu_j(b)}{\theta_j(b)} + w_i - \frac{1}{n}\frac{\mu_j(b)}{\theta_j(b)}.$$

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:

$$\begin{aligned}
1 &= \sum_{i' \in A} \frac{\partial y_{i'}}{\partial x_i} \cdot \frac{\partial}{\partial y_{i'}} \left(w_i \log \frac{\mu(y)}{\theta(y)} + w_i - \frac{1}{n}\frac{\mu(y)}{\theta(y)} \right) \\
&= \sum_{i' \in A} \frac{\partial y_{i'}}{\partial x_i} \cdot \left(\frac{w_i w_{i'}}{\mu(y)} - \frac{w_i}{n\theta(y)} - \frac{w_{i'}\theta(y) - \frac{1}{n}\mu(y)}{n\theta(y)^2} \right) \\
&= \sum_{i' \in A} \frac{\partial y_{i'}}{\partial x_i} \cdot \left(\frac{w_i w_{i'}}{\mu(y)} - \frac{w_i + w_{i'}}{n\theta(y)} + \frac{\mu(y)}{n^2\theta(y)^2} \right),
\end{aligned}$$

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 32 *Algorithm 14 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.*

REFERENCES

[1] A. Nikolov, M. Singh, and U. T. Tantipongpipat, "Proportional volume sampling and approximation algorithms for a-optimal design," *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1369–1386, 2019.

[2] L. C. Lau and H. Zhou, "A local search framework for experimental design," in *Proceedings of the Thirty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '21, Virtual Event, Virginia: Society for Industrial and Applied Mathematics, 2021, pp. 1039–1058, ISBN: 9781611976465.

[3] F. Pukelsheim, *Optimal design of experiments*. SIAM, 2006.

[4] H. Li, S. Patterson, Y. Yi, and Z. Zhang, "Maximizing the number of spanning trees in a connected graph," *IEEE Transactions on Information Theory*, 2019.

[5] L. C. Lau and H. Zhou, "A spectral approach to network design," *SIAM Journal on Computing*, vol. 51, no. 4, pp. 1018–1064, 2022. eprint: https://doi.org/10.1137/20M1330762.

[6] P. H. Chan, L. C. Lau, A. Schild, S. C.-W. Wong, and H. Zhou, "Network design for s-t effective resistance," *ACM Trans. Algorithms*, vol. 18, no. 3, Oct. 2022.

[7] J. B. Barbanel and A. D. Taylor, *The Geometry of Efficient Fair Division*. Cambridge University Press, 2005.

[8] S. J. Brams and A. D. Taylor, *Fair Division: From Cake-Cutting to Dispute Resolution*. Cambridge University Press, 1996.

[9] F. Brandt, V. Conitzer, U. Endriss, J. Lang, and A. D. Procaccia, *Handbook of Computational Social Choice*. Cambridge University Press, 2016.

[10] J. Robertson and W. Webb, *Cake-cutting algorithms: Be fair if you can*. CRC Press, 1998.

[11] J. Rothe, *Economics and Computation: An Introduction to Algorithmic Game Theory, Computational Social Choice, and Fair Division*, 1st. Springer Publishing Company, Incorporated, 2015, ISBN: 3662479036.

[12] H. P. Young, *Equity: In Theory and Practice*. Princeton University Press, 1994, ISBN: 9780691043197.

[13] Z. Allen-Zhu, Y. Li, A. Singh, and Y. Wang, "Near-optimal discrete optimization for experimental design: A regret minimization approach," *Mathematical Programming*, vol. 186, Nov. 2017.

[14] V. Madan, M. Singh, U. Tantipongpipat, and W. Xie, "Combinatorial algorithms for optimal design," in *Proceedings of the Thirty-Second Conference on Learning Theory*, Jun. 2019, pp. 2210–2258.

[15] N. Bansal and M. Sviridenko, "The santa claus problem," in *Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing*, ser. STOC '06, Seattle, WA, USA: Association for Computing Machinery, 2006, pp. 31–40, ISBN: 1595931341.

[16] A. Asadpour and A. Saberi, "An approximation algorithm for max-min fair allocation of indivisible goods," in *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*, ser. STOC '07, San Diego, California, USA: Association for Computing Machinery, 2007, pp. 114–121, ISBN: 9781595936318.

[17] U. Feige, "On allocations that maximize fairness," in *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '08, San Francisco, California: Society for Industrial and Applied Mathematics, 2008, pp. 287–293.

[18] P. Haxell and T. Szabó, "Improved integrality gap in max-min allocation: Or topology at the north pole," in *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2023, pp. 2875–2897. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9781611977554.ch109.

[19] S. Davies, T. Rothvoss, and Y. Zhang, "A tale of santa claus, hypergraphs and matroids," in *Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '20, Salt Lake City, Utah: Society for Industrial and Applied Mathematics, 2020, pp. 2748–2757.

[20] A. Brown, A. Laddha, M. Pittu, M. Singh, and P. Tetali, "Determinant maximization via matroid intersection algorithms," in *63rd IEEE Foundations of Computer Science (FOCS)*, 2022. arXiv: 2207.04318 [cs.DS].

[21] A. Brown, A. Laddha, M. Pittu, and M. Singh, *Efficient determinant maximization for all matroids*, 2022. arXiv: 2211.10507 [cs.DS].

[22] A. Nikolov and M. Singh, "Maximizing determinants under partition constraints," in *ACM symposium on Theory of computing*, 2016, pp. 192–201.

[23] N. Anari and S. O. 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*, ACM, 2017, pp. 384–396.

[24] N. Anari, S. O. Gharan, and C. 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)*, IEEE, 2018, pp. 35–46.

[25] V. Madan, A. Nikolov, M. Singh, and U. Tantipongpipat, *Maximizing determinants under matroid constraints*, 2020.

[26] J. F. Nash Jr, "The bargaining problem," *Econometrica: Journal of the econometric society*, pp. 155–162, 1950.

[27] M. Kaneko and K. Nakamura, "The nash social welfare function," *Econometrica: Journal of the Econometric Society*, pp. 423–435, 1979.

[28] S. Chae and H. Moulin, "Bargaining among groups: An axiomatic viewpoint," *International Journal of Game Theory*, vol. 39, Feb. 2004.

[29] I. Caragiannis, D. Kurokawa, H. Moulin, A. D. Procaccia, N. Shah, and J. Wang, "The unreasonable fairness of maximum nash welfare," *ACM Transactions on Economics and Computation (TEAC)*, vol. 7, no. 3, pp. 1–32, 2019.

[30] S. Barman, S. K. Krishnamurthy, and R. Vaish, "Finding fair and efficient allocations," in *Proceedings of the 2018 ACM Conference on Economics and Computation*, 2018, pp. 557–574.

[31] J. C. Harsanyi and R. Selten, "A generalized nash solution for two-person bargaining games with incomplete information," *Management Science*, vol. 18, no. 5, P80–P106, 1972.

[32] E. Kalai, "Nonsymmetric nash solutions and replications of 2-person bargaining," *International Journal of Game Theory*, vol. 6, pp. 129–133, 1977.

[33] A. Laruelle and F. Valenciano, "Bargaining in committees as an extension of nash's bargaining theory," *Journal of Economic Theory*, vol. 132, no. 1, pp. 291–305, 2007.

[34] H. Fu, R. Kleinberg, and R. 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*, ser. EC '12, Valencia, Spain: Association for Computing Machinery, 2012, p. 586, ISBN: 9781450314152.

[35] H. Houba, G. Laan, and Y. Zeng, "Asymmetric nash solutions in the river sharing problem," *Strateg. Behav. Environ.*, vol. 4, Mar. 2013.

[36] 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*, vol. 17, no. 5, pp. 709–729, Oct. 2017.

[37] J. Garg, P. Kulkarni, and R. Kulkarni, "Approximating nash social welfare under submodular valuations through (un) matchings," in *Proceedings of the fourteenth annual ACM-SIAM symposium on discrete algorithms*, SIAM, 2020, pp. 2673–2687.

[38] J. Garg, E. Husić, and L. A. Végh, "Approximating nash social welfare under rado valuations," in *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, 2021, pp. 1412–1425.

[39] J. Garg, E. Husić, W. Li, L. A. Végh, and J. Vondrák, "Approximating nash social welfare by matching and local search," in *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, 2023, pp. 1298–1310.

[40] M. Chakraborty, A. Igarashi, W. Suksompong, and Y. Zick, "Weighted envy-freeness in indivisible item allocation," *ACM Trans. Econ. Comput.*, vol. 9, no. 3, Aug. 2021.

[41] W. Suksompong, "Weighted fair division of indivisible items: A review," *Inf. Process. Lett.*, vol. 187, no. C, Jan. 2025.

[42] R. Cole *et al.*, "Convex program duality, fisher markets, and nash social welfare," in *Proceedings of the 2017 ACM Conference on Economics and Computation*, ser. EC '17, Cambridge, Massachusetts, USA: Association for Computing Machinery, 2017, pp. 459–460, ISBN: 9781450345279.

[43] N. Anari, S. O. Gharan, A. Saberi, and M. Singh, "Nash Social Welfare, Matrix Permanent, and Stable Polynomials," in *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, C. H. Papadimitriou, Ed., ser. Leibniz International Proceedings in Informatics (LIPIcs), vol. 67, Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017, 36:1–36:12, ISBN: 978-3-95977-029-3.

[44] N. J. Harvey and N. Olver, "Pipage rounding, pessimistic estimators and matrix concentration," in *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, SIAM, 2014, pp. 926–945.

[45] B. Jourdan, P. Macgregor, and H. Sun, *Is the algorithmic kadison-singer problem hard?* 2022. arXiv: 2205.02161 [cs.CC].

[46] K. Sathish Kumar and S. Naveen, "Power system reconfiguration and loss minimization for a distribution systems using "catfish pso" algorithm," *Frontiers in Energy*, vol. 8, no. 4, 434, p. 434, 2014.

[47] A. Khodabakhsh *et al.*, "A submodular approach for electricity distribution network reconfiguration," in *Hawaii International Conference on System Sciences*, 2018.

[48] S. Gupta, A. Khodabakhsh, H. Mortagy, and E. Nikolova, "Electrical Flows over Spanning Trees," *arXiv e-prints*, arXiv:1909.04759, arXiv:1909.04759, Sep. 2019. arXiv: 1909.04759 [cs.DS].

[49] M. E. Baran and F. F. Wu, "Network reconfiguration in distribution systems for loss reduction and load balancing," *IEEE Power Engineering Review*, vol. 9, no. 4, pp. 101–102, 1989.

[50] R. T. Wong, "Worst-case analysis of network design problem heuristics," *SIAM Journal on Algebraic Discrete Methods*, vol. 1, no. 1, pp. 51–63, 1980. eprint: https://doi.org/10.1137/0601008.

[51] B. Y. Wu, G. Lancia, V. Bafna, K.-M. Chao, R. Ravi, and C. Y. Tang, "A polynomial-time approximation scheme for minimum routing cost spanning trees," *SIAM Journal on Computing*, vol. 29, no. 3, pp. 761–778, 2000. eprint: https://doi.org/10.1137/S009753979732253X.

[52] D. Spielman and J. O. Woo, "A note on preconditioning by low-stretch spanning trees," Apr. 2009.

[53] H. Avron and C. Boutsidis, "Faster subset selection for matrices and applications," *SIAM Journal on Matrix Analysis and Applications*, vol. 34, no. 4, pp. 1464–1499, 2013. eprint: https://doi.org/10.1137/120867287.

[54] C. Brezovec, G. Cornuéjols, and F. Glover, "Two algorithms for weighted matroid intersection," *Math. Program.*, vol. 36, no. 1, pp. 39–53, Oct. 1986.

[55] A. Schrijver, *Combinatorial optimization: polyhedra and efficiency*. Springer Science & Business Media, 2003, vol. 24.

[56] S. Krogdahl, "A combinatorial proof for a weighted matroid intersection algorithm," *Computer Science Report*, vol. 17, 1976.

[57] W. H. Cunningham, "Testing membership in matroid polyhedra," *Journal of Combinatorial Theory, Series B*, vol. 36, pp. 161–188, 1984.

[58] J. A. Tropp, "An introduction to matrix concentration inequalities," *Foundations and Trends® in Machine Learning*, vol. 8, no. 1-2, pp. 1–230, 2015.

[59] L. G. Khachiyan, "Rounding of polytopes in the real number model of computation," *Mathematics of Operations Research*, vol. 21, no. 2, pp. 307–320, 1996.

[60] N. Anari, S. O. Gharan, A. Saberi, and M. Singh, "Nash social welfare, matrix permanent, and stable polynomials," *In Proceedings of Conference on Innovations in Theoretical Computer Science*, 2016.

[61] A. Kulesza and B. Taskar, "Determinantal point processes for machine learning," *Foundations and Trends® in Machine Learning*, vol. 5, no. 2–3, pp. 123–286, 2012.

[62] A. Nikolov, "Randomized rounding for the largest simplex problem," in *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, ser. STOC '15, Portland, Oregon, USA: Association for Computing Machinery, 2015, pp. 861–870, ISBN: 9781450335362.

[63] M. D. Summa, F. Eisenbrand, Y. Faenza, and C. Moldenhauer, "On largest volume simplices and sub-determinants," in *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '15, San Diego, California: Society for Industrial and Applied Mathematics, 2015, pp. 315–323.

[64] M. Singh and W. Xie, "Approximate positive correlated distributions and approximation algorithms for D-optimal design," *In Proceedings of SODA*, 2018.

[65] A. Schrijver, *Theory of Linear and Integer Programming*. Wiley-Interscience, 2000.

[66] L. Comtet, *Advanced combinatorics, enlarged ed., d*, 1974.

[67] F. Brenti, "Unimodal polynomials arising from symmetric functions," *Proceedings of the American Mathematical Society*, vol. 108, no. 4, pp. 1133–1141, 1990.

[68] A. Brown, A. Laddha, M. R. Pittu, and M. Singh, "Approximation algorithms for the weighted nash social welfare via convex and non-convex programs," in *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2024, pp. 1307–1327. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9781611977912.52.

[69] N.-T. Nguyen, T. T. Nguyen, M. Roos, and J. Rothe, "Computational complexity and approximability of social welfare optimization in multiagent resource allocation," *Autonomous agents and multi-agent systems*, vol. 28, no. 2, pp. 256–289, 2014.

[70] E. Lee, "Apx-hardness of maximizing nash social welfare with indivisible items," *Information Processing Letters*, vol. 122, pp. 17–20, 2017.

[71] R. Cole and V. Gkatzelis, "Approximating the nash social welfare with indivisible items," in *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, 2015, pp. 371–380.

[72] W. Li and J. 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)*, IEEE, 2022, pp. 25–36.

[73] Y. Feng and S. Li, *A note on approximating weighted nash social welfare with additive valuations*, 2024. arXiv: 2404.15607 [cs.GT].

[74] Y. Feng, Y. Hu, S. Li, and R. Zhang, *Constant approximation for weighted nash social welfare with submodular valuations*, Nov. 2024.

[75] E. Eisenberg and D. Gale, "Consensus of subjective probabilities: The pari-mutuel method," *The Annals of Mathematical Statistics*, vol. 30, no. 1, pp. 165–168, 1959.

[76] V. Shmyrev, "An algorithmic approach for searching an equilibrium in fixed budget exchange models," in Jan. 2006, pp. 217–235, ISBN: 978-3-540-31405-9.

[77] J. 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*, ser. STOC '10, Cambridge, Massachusetts, USA: Association for Computing Machinery, 2010, pp. 291–300, ISBN: 9781450300506.

[78] M. Sion, "On general minimax theorems.," *Pacific Journal of Mathematics*, vol. 8, no. 1, pp. 171–176, 1958.

[79] A. Brown, A. Laddha, and M. Singh, "Fast algorithms for maximizing the minimum eigenvalue in fixed dimension," *Operations Research Letters*, vol. 57, p. 107 186, 2024.

[80] A. W. Marcus, D. A. Spielman, and N. Srivastava, "Interlacing families ii: Mixed characteristic polynomials and the kadison–singer problem," *Annals of Mathematics*, vol. 182, pp. 327–350, 2015.

[81] A. Asadpour, U. Feige, and A. Saberi, "Santa claus meets hypergraph matchings," *ACM Trans. Algorithms*, vol. 8, no. 3, 2012.

[82] D. Chakrabarty, J. Chuzhoy, and S. Khanna, "On allocating goods to maximize fairness," in *Proceedings of the 2009 50th Annual IEEE Symposium on Foundations of Computer Science*, ser. FOCS '09, USA: IEEE Computer Society, 2009, pp. 107–116, ISBN: 9780769538501.

[83] N. Weaver, "The kadison–singer problem in discrepancy theory," *Discrete mathematics*, vol. 278, no. 1-3, pp. 227–239, 2004.

[84] A. Çivril and M. Magdon-Ismail, "On selecting a maximum volume sub-matrix of a matrix and related problems," *Theoretical Computer Science*, vol. 410, pp. 4801–4811, 2009.

[85] J. D. Batson, D. A. Spielman, and N. Srivastava, "Twice-ramanujan sparsifiers," in *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, ser. STOC '09, Bethesda, MD, USA: Association for Computing Machinery, 2009, pp. 255–262, ISBN: 9781605585062.

[86] H. Khodr and J. Martínez-Crespo, "Integral methodology for distribution systems reconfiguration based on optimal power flow using benders decomposition technique," *Generation, Transmission & Distribution, IET*, vol. 3, pp. 521–534, Jul. 2009.

[87] I. Abraham, Y. Bartal, and O. Neiman, "Nearly tight low stretch spanning trees," in *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, 2008, pp. 781–790.

[88] M. X. Goemans and Y.-S. Myung, "A catalog of steiner tree formulations," *Networks*, vol. 23, no. 1, pp. 19–28, 1993. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/net.3230230104.

[89] S. Pokutta, "The frank-wolfe algorithm: A short introduction," *Jahresbericht der Deutschen Mathematiker-Vereinigung*, vol. 126, Dec. 2023.

[90] A. J. HOFFMAN, "Some recent applications of the theory of linear inequalities to extremal combinatorial analysis," in *Selected Papers of Alan J Hoffman*, pp. 244–258. eprint: https://www.worldscientific.com/doi/pdf/10.1142/9789812796936_0025.

[91] G. Lan, *First-order and stochastic optimization methods for machine learning*. Springer, 2020, vol. 1.

[92] D. Davis and D. Drusvyatskiy, "Stochastic model-based minimization of weakly convex functions," *SIAM Journal on Optimization*, vol. 29, no. 1, pp. 207–239, 2019. eprint: https://doi.org/10.1137/18M1178244.