A Faster Cutting Plane Method and its Implications for Combinatorial and Convex Optimization

Yin Tat Lee
MIT
yintat@mit.edu

Aaron Sidford
MIT
sidford@mit.edu

Sam Chiu-wai Wong
UC Berkeley
samcwong@berkeley.edu

Abstract

In this paper we improve upon the running time for finding a point in a convex set given a separation oracle. In particular, given a separation oracle for a convex set $K \subset \mathbb{R}^n$ that is contained in a box of radius R we show how to either compute a point in K or prove that K does not contain a ball of radius ϵ using an expected $O(n \log(nR/\epsilon))$ evaluations of the oracle and additional time $O(n^3 \log^{O(1)}(nR/\epsilon))$. This matches the oracle complexity and improves upon the $O(n^{\omega+1} \log(nR/\epsilon))$ additional time of the previous fastest algorithm achieved over 25 years ago by Vaidya [103] for the current value of the matrix multiplication constant $\omega < 2.373$ [110, 41] when $R/\epsilon = O(\mathsf{poly}(n))$.

Using a mix of standard reductions and new techniques we show how our algorithm can be used to improve the running time for solving classic problems in continuous and combinatorial optimization. In particular we provide the following running time improvements:

- **Submodular Function Minimization**: let n be the size of the ground set, M be the maximum absolute value of function values, and EO be the time for function evaluation.
  Our weakly and strongly polynomial time algorithms have a running time of $O(n^2 \log nM \cdot \mathrm{EO} + n^3 \log^{O(1)} nM)$ and $O(n^3 \log^2 n \cdot \mathrm{EO} + n^4 \log^{O(1)} n)$, improving upon the previous best of $O((n^4 \cdot \mathrm{EO} + n^5) \log M)$ and $O(n^5 \cdot \mathrm{EO} + n^6)$ respectively.
- **Matroid Intersection**: let n be the size of the ground set, r be the maximum size of independent sets, M be the maximum absolute value of element weight, and $\mathcal{T}_{\text{rank}}$ and $\mathcal{T}_{\text{ind}}$ be the time for each rank and independence oracle query.
  We obtain a running time of $O(nr\mathcal{T}_{\text{rank}} \log n \log(nM) + n^3 \log^{O(1)} nM)$ and $O(n^2\mathcal{T}_{\text{ind}} \log(nM) + n^3 \log^{O(1)} nM)$, achieving the first quadratic bound on the query complexity for the independence and rank oracles. In the unweighted case, this is the first improvement since 1986 for independence oracle.
- **Submodular Flow**: let n and m be the number of vertices and edges, C be the maximum edge cost in absolute value, and U be the maximum edge capacity in absolute value.
  We obtain a faster weakly polynomial running time of $O(n^2 \log nCU \cdot \mathrm{EO} + n^3 \log^{O(1)} nCU)$, improving upon the previous best of $O(mn^5 \log nU \cdot \mathrm{EO})$ and $O\left(n^4 h \min\{\log C, \log U\}\right)$ from 15 years ago by a factor of $\tilde{O}(n^4)$. We also achieve faster strongly polynomial time algorithms as a consequence of our result on submodular minimization.
- **Semidefinite Programming**: let n be the number of constraints, m be the number of dimensions and S be the total number of non-zeros in the constraint matrix.
  We obtain a running time of $\tilde{O}(n(n^2 + m^\omega + S))$, improving upon the previous best of $\tilde{O}(n(n^\omega + m^\omega + S))$ for the regime S is small.

Contents

Part
Overview

1 Introduction

The ellipsoid method and more generally, *cutting plane methods*,[1] that is optimization algorithms which iteratively call a separation oracle, have long been central to theoretical computer science. In combinatorial optimization, since Khachiyan's seminal result in 1980 [65] proving that the ellipsoid method solves linear programs in polynomial time, the ellipsoid method has been crucial to solving discrete problems in polynomial time [49]. In continuous optimization, cutting plane methods have long played a critical role in convex optimization, where they are fundamental to the theory of non-smooth optimization [45].

Despite the key role that cutting plane methods have played historically in both combinatorial and convex optimization, over the past two decades progress on improving both the theoretical running time of cutting plane methods as well as the complexity of using cutting plane methods for combinatorial optimization has stagnated.[2] The theoretical running time of cutting plane methods for convex optimization has not been improved since the breakthrough result by Vaidya in 1989 [103, 105]. Moreover, for many of the key combinatorial applications of ellipsoid method, such as submodular minimization, matroid intersection and submodular flow, the running time improvements over the past two decades have been primarily combinatorial; that is they have been achieved by discrete algorithms that do not use numerical machinery such as cutting plane methods.

In this paper we make progress on these classic optimization problems on two fronts. First we show how to improve on the running time of cutting plane methods for a broad range of parameters that arise frequently in both combinatorial applications and convex programming (Part I). Second, we provide several frameworks for applying the cutting plane method and illustrate the efficacy of these frameworks by obtaining faster running times for semidefinite programming, matroid intersection, and submodular flow (Part II). Finally, we show how to couple our approach with the problem specific structure and obtain faster weakly and strongly polynomial running times for submodular function minimization, a problem of tremendous importance in combinatorial optimization (Part III). In both cases our algorithms are faster than previous best by a factor of roughly $\Omega(n^2)$.

We remark that many of our running time improvements come both from our faster cutting method and from new careful analysis of how to apply these cutting plane methods. In fact, simply using our reductions to cutting plane methods and a seminal result of Vaidya [103, 105] on cutting plane methods we provide running times for solving many of these problems that improves upon the previous best stated. As such, we organized our presentation to hopefully make it easy to apply cutting plane methods to optimization problems and obtain provable guarantees in the future.

Our results demonstrate the power of cutting plane methods in theory and possibly pave the way for new cutting plane methods in practice. We show how cutting plane methods can continue to improve running times for classic optimization problems and we hope that these methods may find further use. As cutting plane methods such as analytic cutting plane method [43, 10, 44, 87, 111, 45] are frequently used in practice [48, 42], these techniques may have further implications.

[1] Throughout this paper our focus is on algorithms for polynomial time solvable convex optimization problems given access to a linear separation oracle. Our usage of the term *cutting plane methods*, should not be confused with work on integer programming, an NP-hard problem.

[2] There are exceptions to this trend. For example, [70] showed how to apply cutting plane methods to yield running time improvements for semidefinite programming, and recently [15] showed how to use cutting plane methods to obtain an optimal result for smooth optimization problems.

1.1 Paper Organization

After providing an overview of our results (Section 2) and preliminary information and notation used throughout the paper (Section 3), we split the remainder of the paper into three parts:

- In Part I we provide our new cutting plane method.
- In Part II we provide several general frameworks for using this cutting plane method and illustrate these frameworks with applications in combinatorial and convex optimization.
- In Part III we then consider the more specific problem of submodular function minimization and show how our methods can be used to improve the running time for both strongly and weakly polynomial time algorithms.

We aim to make each part relatively self contained. While each part builds upon the previous and the problems considered in each part are increasingly specific, we present the key results in each section in a modular way so that they may be read in any order. The dependencies between the different parts of our paper are characterized by the following:

- Part I presents our faster cutting plane method as Theorem 31.
- Part II depends only on Theorem 31 of Part I and presents a general running time guarantee for convex optimization problems as Theorem 42.
- The faster weakly polynomial time algorithm in Part III depends only on Theorem 42, Part II.
- The faster strongly polynomial time algorithm in Part III depends only on Theorem 31, Part I.

2 Overview of Our Results

Here we briefly summarize the contributions of our paper. For each of Part I, Part II, and Part III we describe the key technical contributions and present the running time improvements achieved.

2.1 Cutting Plane Methods

The central problem we consider in Part I is as follows. We are promised that a set K is contained a box of radius R and a separation oracle that given a point $\vec{x}$ in time SO either outputs that $\vec{x}$ is in K or outputs a separating hyperplane. We wish to either find a point in K or prove that K does not contain an ball of radius ϵ. The running times for this problem are given in Table 1.

Year	Algorithm	Complexity
1979	Ellipsoid Method [97, 112, 65]	$O(n^2\text{SO}\log\kappa + n^4\log\kappa)$
1988	Inscribed Ellipsoid [66, 88]	$O(n\text{SO}\log\kappa + (n\log\kappa)^{4.5})$
1989	Volumetric Center [103]	$O(n\text{SO}\log\kappa + n^{1+\omega}\log\kappa)$
1995	Analytic Center [10]	$O(n\text{SO}\log^2\kappa + n^{\omega+1}\log^2\kappa + (n\log\kappa)^{2+\omega/2})$
2004	Random Walk [13]	$\rightarrow O(n\text{SO}\log\kappa + n^7\log\kappa)$
2013	This paper	$O(n\text{SO}\log\kappa + n^3\log^{O(1)}\kappa)$

Table 1: Algorithms for the Feasibility Problem. κ indicates nR/ϵ. The arrow, $\rightarrow$, indicates that it solves a more general problem where only a membership oracle is given.

In Part I we show how to solve this problem in $O(n\text{SO}\log(nR/\epsilon) + n^3\log^{O(1)}(nR/\epsilon))$ time. This is an improvement over the previous best running time of $\widetilde{O}(n\text{SO}\log(nR/\epsilon) + n^{\omega+1}\log(nR/\epsilon))$ for the current best known bound of $\omega < 2.37$ [41] assuming that $R/\epsilon = O(\mathsf{poly}(n))$, a common assumption for many problems in combinatorial optimization and numerical analysis as we find in Part II and Part III. (See Table 1 for a summary of previous running times.)

Our key idea for achieving this running time improvement is a new straightforward technique for providing low variance unbiased estimates for changes in leverage scores that we hope will be of independent interest (See Section 7.1). We show how to use this technique along with ideas from [10, 104, 76] to decrease the $\widetilde{O}(n^{\omega+1}\log(D/\epsilon))$ overhead in the previous fastest algorithm [103].

2.2 Convex Optimization

In Part II we provide two techniques for applying our cutting plane method (and cutting plane methods in general) to optimization problems and provide several applications of these techniques.

The first technique concerns reducing the number of dimensions through duality. For many problems, their dual is significantly simpler than itself (primal). We use semidefinite programming as a concrete example to show how to improve upon the running time for finding both primal and dual solution by using the cutting planes maintained by our cutting plane method. (See Table 2.)

The second technique concerns how to minimize a linear function over the intersection of convex sets using optimization oracle. We analyze a simple potential function which allows us to bypass the typical reduction between separation and optimization to achieve faster running times. This reduction provides an improvement over the reductions used previously in [49]. Moroever, we show how this technique allows us to achieve improved running times for matroid intersection and minimum cost submodular flow. (See Tables 2, 3, 4, and 5 for running time summaries.)

Authors	Years	Running times
Nesterov, Nemirovsky[89]	1992	$\tilde{O}(\sqrt{m}(nm^\omega + n^{\omega-1}m^2))$
Anstreicher [7]	2000	$\tilde{O}((mn)^{1/4}(nm^\omega + n^{\omega-1}m^2))$
Krishnan, Mitchell [70]	2003	$\tilde{O}(n(n^\omega + m^\omega + S))$ (dual SDP)
This paper	2015	$\tilde{O}(n(n^2 + m^\omega + S))$

Table 2: Algorithms for solving a $m \times m$ SDP with n constraints and S non-zero entries

Authors	Years	Complexity
Edmonds [26]	1968	not stated
Aigner, Dowling [2]	1971	$O(nr^2\mathcal{T}_{\text{ind}})$
Tomizawa, Iri [102]	1974	not stated
Lawler [72]	1975	$O(nr^2\mathcal{T}_{\text{ind}})$
Edmonds [28]	1979	not stated
Cunningham [21]	1986	$O(nr^{1.5}\mathcal{T}_{\text{ind}})$
This paper	2015	$O(n^2\log n\mathcal{T}_{\text{ind}} + n^3\log^{O(1)} n)$ $O(nr\log^2 n\mathcal{T}_{\text{rank}} + n^3\log^{O(1)} n)$

Table 3: Algorithms for (unweighted) matroid intersection. n is the size of the ground set, r is the maximum rank of the two matroids, $\mathcal{T}_{\text{ind}}$ is the time to check if a set is independent (membership oracle), and $\mathcal{T}_{\text{rank}}$ is the time to compute the rank of a given set (rank oracle).

Authors	Years	Running times
Edmonds [26]	1968	not stated
Tomizawa, Iri [102]	1974	not stated
Lawler [72]	1975	$O(nr^2\mathcal{T}_{\text{ind}} + nr^3)$
Edmonds [28]	1979	not stated
Frank [33]	1981	$O(n^2r(\mathcal{T}_{\text{circuit}} + n))$
Orlin, Ahuja [91]	1983	not stated
Brezovec, Cornuéjols, Glover[14]	1986	$O(nr(\mathcal{T}_{\text{circuit}} + r + \log n))$
Fujishige, Zhang [39]	1995	$O(n^2r^{0.5}\log rM \cdot \mathcal{T}_{\text{ind}})$
Shigeno, Iwata [96]	1995	$O((n + \mathcal{T}_{\text{circuit}})nr^{0.5}\log rM)$
This paper	2015	$O(n^2\log nM\mathcal{T}_{\text{ind}} + n^3\log^{O(1)} nM)$ $O(nr\log n\log nM\mathcal{T}_{\text{rank}} + n^3\log^{O(1)} nM)$

Table 4: Algorithms for weighted matroid intersection. In addition to the notation in Table 3 $\mathcal{T}_{\text{circuit}}$ is the time needed to find a fundamental circuit and M is the bit complexity of the weights.

Authors	Years	Running times
Fujishige [35]	1978	not stated
Grotschel, Lovasz, Schrijver[49]	1981	weakly polynomial
Zimmermann [113]	1982	not stated
Barahona, Cunningham [12]	1984	not stated
Cunningham, Frank [22]	1985	$\rightarrow O(n^4h\log C)$
Fujishige [36]	1987	not stated
Frank, Tardos [34]	1987	strongly polynomial
Cui, Fujishige [108]	1988	not stated
Fujishige, Röck, Zimmermann[38]	1989	$\rightarrow O(n^6h\log n)$
Chung, Tcha [18]	1991	not stated
Zimmermann [114]	1992	not stated
McCormick, Ervolina [82]	1993	$O(n^7h^*\log nCU)$
Wallacher, Zimmermann [109]	1994	$O(n^8h\log nCU)$
Iwata [52]	1997	$O(n^7h\log U)$
Iwata, McCormick, Shigeno [57]	1998	$O\left(n^4h\min\left\{\log nC, n^2\log n\right\}\right)$
Iwata, McCormick, Shigeno [58]	1999	$O\left(n^6h\min\left\{\log nU, n^2\log n\right\}\right)$
Fleischer, Iwata, McCormick[32]	1999	$O\left(n^4h\min\left\{\log U, n^2\log n\right\}\right)$
Iwata, McCormick, Shigeno [59]	1999	$O\left(n^4h\min\left\{\log C, n^2\log n\right\}\right)$
Fleischer, Iwata [30]	2000	$O(mn^5\log nU \cdot \text{EO})$
This paper	2015	$O(n^2\log nCU \cdot \text{EO} + n^3\log^{O(1)} nCU)$

Table 5: Algorithms for minimum cost submodular flow with n vertices, maximum cost C and maximum capacity U. The factor h is the time for an exchange capacity oracle, h^* is the time for a "more complicated exchange capacity oracle," and EO is the time for evaluation oracle of the submodular function. The arrow, $\rightarrow$, indicates that it uses the current best submodular flow algorithm as subroutine which was non-existent at the time of the publication.

2.3 Submodular Function Minimization

In Part III we consider the problem of submodular minimization, a fundamental problem in combinatorial optimization with many diverse applications in theoretical computer science, operations research, machine learning and economics. We show that by considering the interplay between the guarantees of our cutting plane algorithm and the primal-dual structure of submodular minimization we can achieve improved running times in various settings.

First, we show that a direct application of our method yields an improved weakly polynomial time algorithm for submodular minimization. Then, we present a simple geometric argument that submodular function can be solved with $O(n^3 \log n \cdot \text{EO})$ oracle calls but with exponential running time. Finally, we show that by further studying the combinatorial structure of submodular minimization and a modification to our cutting plane algorithm we can obtained a fully improved strongly polynomial time algorithm for submodular minimization. We summarize the improvements in Table 6.

Authors	Years	Running times	Remarks
Grötschel, Lovász, Schrijver [49, 50]	1981,1988	$\widetilde{O}(n^5 \cdot \text{EO} + n^7)$ [81]	first weakly and strongly
Cunningham [20]	1985	$O(Mn^6 \log nM \cdot \text{EO})$	first combin. pseudopoly
Schrijver [93]	2000	$O(n^8 \cdot \text{EO} + n^9)$	first combin. strongly
Iwata, Fleischer, Fujishige[56]	2000	$O(n^5 \cdot \text{EO} \log M)$ $O(n^7 \log n \cdot \text{EO})$	first combin. strongly
Iwata, Fleischer [31]	2000	$O(n^7 \cdot \text{EO} + n^8)$	
Iwata [54]	2003	$O((n^4 \cdot \text{EO} + n^5) \log M)$ $O((n^6 \cdot \text{EO} + n^7) \log n)$	current best weakly
Vygen [107]	2003	$O(n^7 \cdot \text{EO} + n^8)$	
Orlin [90]	2007	$O(n^5 \cdot \text{EO} + n^6)$	current best strongly
Iwata, Orlin [60]	2009	$O((n^4 \cdot \text{EO} + n^5) \log nM)$ $O((n^5 \cdot \text{EO} + n^6) \log n)$	
Our algorithms	2015	$O(n^2 \log nM \cdot \text{EO} + n^3 \log^{O(1)} nM)$ $O(n^3 \log^2 n \cdot \text{EO} + n^4 \log^{O(1)} n)$	

Table 6: Algorithms for submodular function minimization.

3 Preliminaries

Here we introduce notation and concepts we use throughout the paper.

3.1 Notation

Basics: Throughout this paper, we use vector notation, e.g $\vec{x} = (x_1, \ldots, x_n)$, to denote a vector and bold, e.g. $\mathbf{A}$, to denote a matrix. We use $\text{nnz}(\vec{x})$ or $\text{nnz}(\mathbf{A})$ to denote the number of nonzero entries in a vector or a matrix respectively. Frequently, for $\vec{x} \in \mathbb{R}^d$ we let $\mathbf{X} \in \mathbb{R}^{d \times d}$ denote $\mathbf{diag}(\vec{x})$, the diagonal matrix such that $\mathbf{X}_{ii} = x_i$. For a symmetric matrix, $\mathbf{M}$, we let $\text{diag}(\mathbf{M})$ denote the vector corresponding to the diagonal entries of $\mathbf{M}$, and for a vector, $\vec{x}$, we let $\|\vec{x}\|_{\mathbf{M}} \stackrel{\text{def}}{=} \sqrt{\vec{x}^T \mathbf{M} \vec{x}}$.

Running Times: We typically use XO to denote the running time for invoking the oracle, where X depends on the type of oracle, e.g., SO typically denotes the running time of a separation oracle, EO denotes the running time of an evaluation oracle, etc. Furthermore, we use $\widetilde{O}(f) \stackrel{\text{def}}{=} O(f \log^{O(1)} f)$.

Spectral Approximations: For symmetric matrices $\mathbf{N}, \mathbf{M} \in \mathbb{R}^{n \times n}$, we write $\mathbf{N} \preceq \mathbf{M}$ to denote that $\vec{x}^T \mathbf{N} \vec{x} \leq \vec{x}^T \mathbf{M} \vec{x}$ for all $\vec{x} \in \mathbb{R}^n$ and we define $\mathbf{N} \succeq \mathbf{M}$, $\mathbf{N} \prec \mathbf{M}$ and $\mathbf{N} \succ \mathbf{M}$ analogously.

Standard Convex Sets: We let $B_p(r) \stackrel{\text{def}}{=} \{\vec{x} : \left\|\vec{x}\right\|_p \leq r\}$ denote a ball of radius r in the ℓ_p norm. For brevity we refer to $B_2(r)$ as a a *ball of radius* r and $B_\infty(r)$ as a *box of radius* r.

Misc: We let $\omega < 2.373$ [110] denote the matrix multiplication constant.

3.2 Separation Oracles

Throughout this paper we frequently make assumptions about the existence of separation oracles for sets and functions. Here we formally define these objects as we use them throughout the paper. Our definitions are possibly non-standard and chosen to handle the different settings that occur in this paper.

Definition 1 (Separation Oracle for a Set)**.** Given a set $K \subset \mathbb{R}^n$ and $\delta \geq 0$, a *δ-separation oracle* for K is a function on $\mathbb{R}^n$ such that for any input $\vec{x} \in \mathbb{R}^n$, it either outputs "successful" or a half space of the form $H = \{\vec{z} : \vec{c}^T \vec{z} \leq \vec{c}^T \vec{x} + b\} \supseteq K$ with $b \leq \delta \left\|\vec{c}\right\|_2$ and $\vec{c} \neq \vec{0}$. We let $\text{SO}_\delta(K)$ be the time complexity of this oracle.

For brevity we refer to a 0-separation oracle for a set as just a *separation oracle.* We refer to the hyperplanes defining the halfspaces returned by a δ-separation oracle as *oracle hyperplanes.*

Note that in Definition 1 we do not assume that K is convex. However, we remark that it is well known that there is a separation oracle for a set if and only if it is convex and that there is a δ separation oracle if and only if the set is close to convex in some sense.

Definition 2 (Separation Oracle for a Function)**.** For any convex function f, $\eta \geq 0$ and $\delta \geq 0$, a (η, δ)-separation oracle on a convex set Γ for f is a function on $\mathbb{R}^n$ such that for any input $\vec{x} \in \Gamma$, it either asserts $f(\vec{x}) \leq \min_{\vec{y} \in \Gamma} f(\vec{y}) + \eta$ or outputs a half space H such that

$$\{\vec{z} \in \Gamma : f(\vec{z}) \leq f(\vec{x})\} \subset H \stackrel{\text{def}}{=} \{\vec{z} : \vec{c}^T \vec{z} \leq \vec{c}^T \vec{x} + b\} \tag{3.1}$$

with $b \leq \delta \left\|\vec{c}\right\|$ and $\vec{c} \neq \vec{0}$. We let $\text{SO}_{\eta,\delta}(f)$ be the time complexity of this oracle.

Part I
A Faster Cutting Plane Method

4 Introduction

Throughout Part I we study the following *feasibility problem*:

Definition 3 (Feasibility Problem)**.** Given a separation oracle for a set $K \subseteq \mathbb{R}^n$ contained in a box of radius R either find a point $\vec{x} \in K$ or prove that K does not contain a ball of radius ϵ.

This feasibility problem is one of the most fundamental and classic problems in optimization. Since the celebrated result of Yudin and Nemirovski [112] in 1976 and Khachiyan [65] in 1979 essentially proving that it can be solved in time $O(\mathsf{poly}(n) \cdot \mathrm{SO} \cdot \log(R/\epsilon))$, this problem has served as one of the key primitives for solving numerous problems in both combinatorial and convex optimization.

Despite the prevalence of this feasibility problem, the best known running time for solving this problem has not been improved in over 25 years. In a seminal paper of Vaidya in 1989 [103], he showed how to solve the problem in $\widetilde{O}(n \cdot \mathrm{SO} \cdot \log(nR/\epsilon) + n^{\omega+1} \log(nR/\epsilon))$ time. Despite interesting generalizations and practical improvements [5, 92, 43, 10, 44, 87, 111, 45, 15], the best theoretical guarantees for solving this problem have not been improved since.

In Part I we show how to improve upon Vaidya's running time in certain regimes. We provide a cutting plane algorithm which achieves an expected running time of $O(n \cdot \mathrm{SO} \cdot \log(nR/\epsilon) + n^3 \log^{O(1)}(nR/\epsilon))$, improving upon the previous best known running time for the current known value of $\omega < 2.373$ [110, 41] when $R/\epsilon = O(\mathsf{poly}(n))$.

We achieve our results by the combination of multiple techniques. First we show how to use techniques from the work of Vaidya and Atkinson to modify Vaidya's scheme so that it is able to tolerate random noise in the computation in each iteration. We then show how to use known numerical machinery [104, 99, 76] in combination with some new techniques (Section 7.1 and Section 7.2) to implement each of these relaxed iterations efficiently. We hope that both these numerical techniques as well as our scheme for approximating complicated methods, such as Vaidya's, may find further applications.

While our paper focuses on theoretical aspects of cutting plane methods, we achieve our results via the careful application of practical techniques such as dimension reduction and sampling. As such we hope that ideas in this paper may lead to improved practical[3] algorithms for non-smooth optimization.

4.1 Previous Work

Throughout this paper, we restrict our attention to algorithms for the feasibility problem that have a polynomial dependence on SO, n, and $\log(R/\epsilon)$. Such "efficient" algorithms typically follow the following iterative framework. First, they compute some trivial region Ω that contains K. Then, they call the separation oracle at some point $\vec{x} \in \Omega$. If $\vec{x} \in K$ the algorithm terminates having successfully solved the problem. If $\vec{x} \notin K$ then the separation oracle must return a half-space

[3] Although cutting plane methods are often criticized for their empirical performance, recently, Bubeck, Lee and Singh [15] provided a variant of the ellipsoid method that achieves the same convergence rate as Nesterov's accelerated gradient descent. Moreover, they provided numerical evidence that this method can be superior to Nesterov's accelerated gradient descent, thereby suggesting that cutting plane methods can be as aggressive as first order methods if designed properly.

Year	Algorithm	Complexity
1979	Ellipsoid Method [97, 112, 65]	$O(n^2\text{SO}\log\kappa + n^4\log\kappa)$
1988	Inscribed Ellipsoid [66, 88]	$O(n\text{SO}\log\kappa + (n\log\kappa)^{4.5})$
1989	Volumetric Center [103]	$O(n\text{SO}\log\kappa + n^{1+\omega}\log\kappa)$
1995	Analytic Center [10]	$O\left(\begin{array}{c} n\text{SO}\log^2\kappa + n^{\omega+1}\log^2\kappa \\ +(n\log\kappa)^{2+\omega/2}\end{array}\right)$
2004	Random Walk [13]	$\rightarrow O(n\text{SO}\log\kappa + n^7\log\kappa)$
2013	This paper	$O(n\text{SO}\log\kappa + n^3\log^{O(1)}\kappa)$

Table 7: Algorithms for the Feasibility Problem. κ indicates nR/ϵ. The arrow, $\rightarrow$, indicates that it solves a more general problem where only a membership oracle is given.

containing K. The algorithm then uses this half-space to shrink the region Ω while maintaining the invariant that $K \subseteq \Omega$. The algorithm then repeats this process until it finds a point $\vec{x} \in K$ or the region Ω becomes too small to contain a ball with radius ϵ.

Previous works on efficient algorithms for the feasibility problem all follow this iterative framework. They vary in terms of what set Ω they maintain, how they compute the center to query the separation oracle, and how they update the set. In Table 7, we list the previous running times for solving the feasibility problem. As usual SO indicates the cost of the separation oracle. To simplify the running times we let $\kappa \stackrel{\text{def}}{=} nR/\epsilon$. The running times of some algorithms in the table depend on R/ϵ instead of nR/ϵ. However, for many situations, we have $\log(R/\epsilon) = \Theta(\log(nR/\epsilon))$ and hence we believe this is still a fair comparison.

The first efficient algorithm for the feasibility problem is the *ellipsoid* method, due to Shor [97], Nemirovksii and Yudin [112], and Khachiyan [65]. The ellipsoid method maintains an ellipsoid as Ω and uses the center of the ellipsoid as the next query point. It takes $\Theta(n^2\log\kappa)$ calls of oracle which is far from the lower bound $\Omega(n\log\kappa)$ calls [86].

To alleviate the problem, the algorithm could maintain all the information from the oracle, i.e., the polytope created from the intersection of all half-spaces obtained. The center of gravity method [77] achieves the optimal oracle complexity using this polytope and the center of gravity of this polytope as the next point. However, computing center of gravity is computationally expensive and hence we do not list its running time in Table 7. The Inscribed Ellipsoid Method [66] also achieved an optimal oracle complexity using this polytope as Ω but instead using the center of the maximal inscribed ellipsoid in the polytope to query the separation oracle. We listed it as occurring in year 1988 in Table 7 because it was [88] that yielded the first polynomial time algorithm to actually compute this maximal inscribed ellipsoid for polytope.

Vaidya [103] obtained a faster algorithm by maintaining an approximation of this polytope and using a different center, namely the volumetric center. Although the oracle complexity of this volumetric center method is very good, the algorithm is not extremely efficient as each iteration involves matrix inversion. Atkinson and Vaidya [10] showed how to avoid this computation in certain settings. However, they were unable to achieve the desired convergence rate from their method.

Bertsimas and Vempala [13] also gives an algorithm that avoids these expensive linear algebra operations while maintaining the optimal convergence rate by using techniques in sampling convex sets. Even better, this result works for a much weaker oracle, the membership oracle. However, the additional cost of this algorithm is relatively high in theory. We remark that while there are considerable improvemenst on the sampling techniques [79, 63, 76], the additional cost is still quite high compared to standard linear algebra.

4.2 Challenges in Improving Previous Work

Our algorithm builds upon the previous fastest algorithm of Vaidya [105]. Ignoring implementation details and analysis, Vaidya's algorithm is quite simple. This algorithm simply maintains a polytope $P^{(k)} = \{x \in \mathbb{R}^n : \mathbf{A}\vec{x} - \vec{b} \geq \vec{0}\}$ as the current Ω and uses the *volumetric center*, the minimizer of the following *volumetric barrier function*

$$\arg\min_{\vec{x}} \frac{1}{2} \log\det\left(\mathbf{A}^T \mathbf{S}_{\vec{x}}^{-2} \mathbf{A}\right) \quad \text{where} \quad \mathbf{S}_{\vec{x}} \stackrel{\text{def}}{=} \mathbf{diag}(\mathbf{A}\vec{x} - \vec{b}) \tag{4.1}$$

as the point at which to query the separation oracle. The polytope is then updated by adding shifts of the half-spaces returned by the separation oracle and dropping unimportant constraints. By choosing the appropriate shift, picking the right rule for dropping constraints, and using Newton's method to compute the volumetric center he achieved a running time of $O(n \cdot SO \cdot \log \kappa + n^{1+\omega} \log \kappa)$.

While Vaidya's algorithm's dependence on SO is essentially optimal, the additional per-iteration costs of his algorithm could possibly be improved. The computational bottleneck in each iteration of Vaidya's algorithm is computing the gradient of $\log\det$ which in turn involves computing the leverage scores $\vec{\sigma}(\vec{x}) \stackrel{\text{def}}{=} \text{diag}(\mathbf{S}_x^{-1}\mathbf{A}\left(\mathbf{A}^T\mathbf{S}_x^{-2}\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{S}_x^{-1})$, a commonly occurring quantity in numerical analysis and convex optimization [99, 19, 78, 76, 75]. As the best known algorithms for computing leverage scores exactly in this setting take time $O(n^\omega)$, directly improving the running time of Vaidya's algorithm seems challenging.

However, since an intriguing result of Spielman and Srivastava in 2008 [99], it has been well known that using Johnson-Lindenstrauss transform these leverage scores can be computed up to a multiplicative $(1 \pm \epsilon)$ error by solving $O(\epsilon^{-2} \log n)$ linear systems involving $\mathbf{A}^T\mathbf{S}_x^{-2}\mathbf{A}$. While in general this still takes time $O(\epsilon^{-2} n^\omega)$, there are known techniques for efficiently maintaining the inverse of a matrix so that solving linear systems take amortized $O(n^2)$ time [104, 75, 76]. Consequently if it could be shown that computing *approximate* leverage scores sufficed, this would potentially decrease the amortized cost per iteration of Vaidya's method.

Unfortunately, Vaidya's method does not seem to tolerate this type of multiplicative error. If leverage scores were computed this crudely then in using them to compute approximate gradients for (4.1), it seems that any point computed would be far from the true center. Moreover, without being fairly close to the true volumetric center, it is difficult to argue that such a cutting plane method would make sufficient progress.

To overcome this issue, it is tempting to directly use recent work on improving the running time of linear program [75]. In this work, the authors faced a similar issue where a volumetric, i.e. $\log\det$, potential function had the right analytic and geometric properties, however was computational expensive to minimize. To overcome this issue the authors instead computed a weighted analytic center:

$$\arg\min_{\vec{x}} -\sum_{i \in [m]} w_i \log s_i(\vec{x}) \quad \text{where} \quad \vec{s}(\vec{x}) \stackrel{\text{def}}{=} \mathbf{A}\vec{x} - \vec{b} \quad .$$

For carefully chosen weights this center provides the same convergence guarantees as the volumetric potential function, while each step can be computed by solving few linear systems (rather than forming the matrix inverse).

Unfortunately, it is unclear how to directly extend the work in [75] on solving an explicit linear program to the feasibility problem specified by a separation oracle. While it is possible to approximate the volumetric barrier by a weighted analytic center in many respects, proving that this approximation suffices for fast convergence remains open. In fact, the volumetric barrier

function as used in Vaidya's algorithm is well approximated simply by the standard analytic center

$$\arg\min_{\vec{x}} -\sum_{i\in[m]} \log s_i(\vec{x}) \quad \text{where} \quad \vec{s}(\vec{x}) \stackrel{\text{def}}{=} \mathbf{A}\vec{x} - \vec{b} \quad .$$

as all the unimportant constraints are dropped during the algorithm. However, despite decades of research, the best running times known for solving the feasibility problem using the analytic center are Vaidya and Atkinson algorithm from 1995 [10]. While the running time of this algorithm could possibly be improved using approximate leverage score computations and amortized efficient linear system solvers, unfortunately at best, without further insight this would yield an algorithm which requires a suboptimal $O(n \log^{O(1)} \kappa)$ queries to the separation oracle.

As pointed out in [10], the primary difficulty in using any sort of analytic center is quantifying the amount of progress made in each step. We still believe providing direct near-optimal analysis of weighted analytic center is a tantalizing open question warranting further investigation. However, rather than directly address the question of the performance of weighted analytic centers for the feasibility problem, we take a slightly different approach that side-steps this issue. We provide a partial answer that still sheds some light on the performance of the weighted analytic center while still providing our desired running time improvements.

4.3 Our Approach

To overcome the shortcoming of the volumetric and analytic centers we instead consider a hybrid barrier function

$$\arg\min_{\vec{x}} -\sum_{i\in[m]} w_i \log s_i(\vec{x}) + \log\det(\mathbf{A}^T\mathbf{S}_x^{-1}\mathbf{A}) \quad \text{where} \quad \vec{s}(\vec{x}) \stackrel{\text{def}}{=} \mathbf{A}\vec{x} - \vec{b} \quad .$$

for careful chosen weights. Our key observation is that for correct choice of weights, we can compute the gradient of this potential function. In particular if we let $\vec{w} = \vec{\tau} - \vec{\sigma}(\vec{x})$ then the gradient of this potential function is the same as the gradients of $\sum_{i\in[m]} \tau_i \log s_i(\vec{x})$, which we can compute efficiently. Moreover, since we are using $\log\det$, we can use analysis similar to Vaidya's algorithm [103] to analyze the convergence rate of this algorithm.

Unfortunately, this is a simple observation and does not immediately change the problem substantially. It simply pushes the problem of computing gradients of $\log\det$ to computing $\vec{w}$. Therefore, for this scheme to work, we would need to ensure that the weights do not change too much and that when they change, they do not significantly hurt the progress of our algorithm. In other words, for this scheme to work, we would still need very precise estimates of leverage scores.

However, we note that the leverage scores $\vec{\sigma}(\vec{x})$ do not change too much between iterations. Moreover, we provide what we believe is an interesting technical result that an unbiased estimates to the changes in leverage scores can be computed using linear system solvers such that the *total error* of the estimate is bounded by the total change of the leverage scores (See Section 7.1). Using this result our scheme simply follows Vaidya's basic scheme in [103], however instead of minimizing the hybrid barrier function directly we alternate between taking Newton steps we can compute, changing the weights so that we can still compute Newton steps, and computing accurate unbiased estimates of the changes in the leverage scores so that the weights do not change adversarially by too much.

To make this scheme work, there are two additional details that need to be dealt with. First, we cannot let the weights vary too much as this might ultimately hurt the rate of progress of our algorithm. Therefore, in every iteration we compute a single leverage score to high precision to

control the value of w_i and we show that by careful choice of the index we can ensure that no weight gets too large (See Section 7.2).

Second, we need to show that changing weights does not affect our progress by much more than the progress we make with respect to log det. To do this, we need to show the slacks are bounded above and below. We enforce this by adding regularization terms and instead consider the potential function

$$p_{\vec{c}}(\vec{x}) = -\sum_{i\in[m]} w_i \log s_i(\vec{x}) + \frac{1}{2}\log\det\left(\mathbf{A}^T\mathbf{S}_x^{-2}\mathbf{A} + \lambda\mathbf{I}\right) + \frac{\lambda}{2}\|x\|_2^2$$

This allows us to ensure that the entries of $\vec{s}(\vec{x})$ do not get too large or too small and therefore changing the weighting of the analytic center cannot affect the function value too much.

Third, we need to make sure our potential function is convex. If we simply take $\vec{w} = \vec{\tau} - \vec{\sigma}(\vec{x})$ with $\vec{\tau}$ as an estimator of $\vec{\sigma}(\vec{x})$, $\vec{w}$ can be negative and the potential function could be non-convex. To circumvent this issue, we use $\vec{w} = c_e + \vec{\tau} - \vec{\sigma}(\vec{x})$ and make sure $\left\|\vec{\tau} - \vec{\sigma}(\vec{x})\right\|_\infty < c_e$.

Combining these insights, using efficient algorithms for solving a sequence of slowly changing linear systems [104, 75, 76], and providing careful analysis ultimately allows us to achieve a running time of $O(n\mathrm{SO}\log\kappa + n^3\log^{O(1)}\kappa)$ for the feasibility problem. Furthermore, in the case that K does not contain a ball of radius ϵ, our algorithm provides a proof that the polytope does not contain a ball of radius ϵ. This proof ultimately allows us to achieve running time improvements for strongly polynomial submodular minimization in Part III.

4.4 Organization

The rest of Part I is organized as follows. In Section 5 we provide some preliminary information and notation we use throughout Part I. In Section 6 we then provide and analyze our cutting plane method. In Section 7 we provide key technical tools which may be of independent interest.

5 Preliminaries

Here we introduce some notation and concepts we use throughout Part I.

5.1 Leverage Scores

Our algorithms in this section make extensive use of *leverage scores*, a common measure of the importance of rows of a matrix. We denote the leverage scores of a matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$ by $\vec{\sigma} \in \mathbb{R}^n$ and say the *leverage score of row* $i \in [n]$ is $\sigma_i \stackrel{\text{def}}{=} [\mathbf{A}\left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T]_{ii}$. For $\mathbf{A} \in \mathbb{R}^{n\times d}$, $\vec{d} \in \mathbb{R}^n_{>0}$, and $\mathbf{D} \stackrel{\text{def}}{=} \mathbf{diag}(\vec{d})$ we use the shorthand $\vec{\sigma}_{\mathbf{A}}(\vec{d})$ to denote the leverage scores of the matrix $\mathbf{D}^{1/2}\mathbf{A}$. We frequently use well known facts regarding leverage scores, such as $\sigma_i \in [0,1]$ and $\left\|\vec{\sigma}\right\|_1 \le d$. (See [99, 80, 78, 19] for a more in-depth discussion of leverage scores, their properties, and their many applications.) In addition, we make use of the fact that given an efficient linear system solver of $\mathbf{A}^T\mathbf{A}$ we can efficiently compute multiplicative approximations to leverage scores (See Definition 4 and Lemma 5 below).

Definition 4 (Linear System Solver)**.** An algorithm $\mathtt{S}$ is a LO-time solver of a PD matrix $\mathbf{M} \in \mathbb{R}^{n\times n}$ if for all $\vec{b} \in \mathbb{R}^n$ and $\epsilon \in (0, 1/2]$, the algorithm outputs a vector $\mathtt{S}(\vec{b}, \epsilon) \in \mathbb{R}^n$ in time $O(\mathrm{LO}\cdot\log(\epsilon^{-1}))$ such that with high probability in n, $\left\|\mathtt{S}(\vec{b},\epsilon) - \mathbf{M}^{-1}\vec{b}\right\|_{\mathbf{M}}^2 \le \epsilon\left\|\mathbf{M}^{-1}\vec{b}\right\|_{\mathbf{M}}^2$.

Lemma 5 (Computing Leverage Scores [99])**.** *Let $\mathbf{A} \in \mathbb{R}^{n\times d}$, let $\vec{\sigma}$ denote the leverage scores of $\mathbf{A}$, and let $\epsilon > 0$. If we have a* LO*-time solver for $\mathbf{A}^T\mathbf{A}$ then in time $\widetilde{O}((\mathrm{nnz}(\mathbf{A}) + \mathrm{LO})\epsilon^{-2}\log(\epsilon^{-1}))$ we can compute $\vec{\tau} \in \mathbb{R}^n$ such that with high probability in d, $(1-\epsilon)\sigma_i \le \tau_i \le (1+\epsilon)\sigma_i$ for all $i \in [n]$.*

5.2 Hybrid Barrier Function

As explained in Section 4.3 our cutting plane method maintains a polytope $P = \{\vec{x} \in \mathbb{R}^n : \mathbf{A}\vec{x} \geq \vec{b}\}$ for $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\vec{b} \in \mathbb{R}^n$ that contains some target set K. We then maintain a minimizer of the following hybrid barrier function:

$$p_{\vec{e}}(\vec{x}) \stackrel{\text{def}}{=} -\sum_{i \in [m]} (c_e + e_i) \log s_i(\vec{x}) + \frac{1}{2} \log \det \left(\mathbf{A}^T \mathbf{S}_x^{-2} \mathbf{A} + \lambda \mathbf{I}\right) + \frac{\lambda}{2} \|x\|_2^2$$

where $\vec{e} \in \mathbb{R}^m$ is a variable we maintain, $c_e \geq 0$ and $\lambda \geq 0$ are constants we fix later, $\vec{s}(\vec{x}) \stackrel{\text{def}}{=} \mathbf{A}\vec{x} - \vec{b}$, and $\mathbf{S}_x = \mathbf{diag}(\vec{s}(\vec{x}))$. When the meaning is clear from context we often use the shorthand $\mathbf{A}_x \stackrel{\text{def}}{=} \mathbf{S}_x^{-1}\mathbf{A}$.

Rather than maintaining $\vec{e}$ explicitly, we instead maintain a vector $\vec{\tau} \in \mathbb{R}^m$ that approximates the leverage score

$$\vec{\psi}(\vec{x}) \stackrel{\text{def}}{=} \operatorname{diag}\left(\mathbf{A}_x \left(\mathbf{A}_x^T \mathbf{A}_x + \lambda \mathbf{I}\right)^{-1} \mathbf{A}_x^T\right) \quad .$$

Note that $\vec{\psi}(\vec{x})$ is simply the leverage scores of certain rows of the matrix

$$\begin{bmatrix} \mathbf{A}_x \\ \sqrt{\lambda}\mathbf{I} \end{bmatrix}.$$

and therefore the usual properties of leverage scores hold, i.e. $\psi_i(\vec{x}) \in (0, 1)$ and $\left\|\psi_i(\vec{x})\right\|_1 \leq n$. We write $\vec{\psi}(\vec{x})$ equivalently as $\vec{\psi}_{\mathbf{A}_x}$ or $\vec{\psi}_P$ when we want the matrix to be clear. Furthermore, we let $\mathbf{\Psi}_x \stackrel{\text{def}}{=} \mathbf{diag}(\vec{\psi}(\vec{x}))$ and $\mu(\vec{x}) \stackrel{\text{def}}{=} \min_i \psi_i(\vec{x})$. Finally, we typically pick $\vec{e}$ using the function $\vec{e}_P(\vec{\tau}, \vec{x}) \stackrel{\text{def}}{=} \vec{\tau} - \vec{\psi}(\vec{x})$. Again, we use the subscripts of $\mathbf{A}_x$ and P interchangeably and often drop them when the meaning is clear from context.

We remark that the last term $\frac{\lambda}{2}\|x\|_2^2$ ensures that our point is always within a certain region (Lemma 23) and hence the term $(c_e + e_i) \log s_i(\vec{x})_i$ never gets too large. However, this ℓ^2 term changes the Hessian of the potential function and hence we need to put a $\lambda\mathbf{I}$ term inside both the log det and the leverage score to reflect this. This is the reason why we use $\vec{\psi}$ instead of the standard leverage score.

6 Our Cutting Plane Method

In this section we develop and prove the correctness of our cutting plane method. We use the notation introduced in Section 3 and Section 5 as well as the technical tools we introduce in Section 7.

We break the presentation and proof of correctness of our cutting plane methods into multiple parts. First in Section 6.1 we describe how we maintain a center of the hybrid barrier function $p_{\vec{e}}$ and analyze this procedure. Then, in Section 6.2 we carefully analyze the effect of changing constraints on the hybrid barrier function and in Section 6.3 we prove properties of an approximate center of hybrid barrier function, which we call a hybrid center. In Section 6.4 we then provide our cutting plane method and in Section 6.5 we prove that the cutting plane method solves the feasibility problem as desired.

6.1 Centering

In this section we show how to compute approximate *centers* or minimizers of the hybrid barrier function for the current polytope $P = \{\vec{x} : \mathbf{A}\vec{x} \geq \vec{b}\}$. We split this proof up into multiple parts. First we simply bound the gradient and Hessian of the hybrid barrier function, $p_{\vec{e}}$, as follows.

Lemma 6. *For* $f(\vec{x}) \stackrel{\text{def}}{=} \frac{1}{2}\log\det\left(\mathbf{A}^T\mathbf{S}_x^{-2}\mathbf{A}+\lambda\mathbf{I}\right)$*, we have that*

$$\nabla f(\vec{x}) = -\mathbf{A}_x^T\vec{\psi}(\vec{x}) \quad \text{and} \quad \mathbf{A}_x^T\mathbf{\Psi}(\vec{x})\mathbf{A}_x \preceq \nabla^2 f(\vec{x}) \preceq 3\mathbf{A}_x^T\mathbf{\Psi}(\vec{x})\mathbf{A}_x \quad .$$

Proof. Our proof is similar to [4, Appendix] which proved the statement when $\lambda = 0$. This case does not change the derivation significantly, however for completeness we include the proof below.

We take derivatives on $\vec{s}$ first and then apply chain rule. Let $f(\vec{s}) = \frac{1}{2}\log\det\left(\mathbf{A}^T\mathbf{S}^{-2}\mathbf{A}+\lambda\mathbf{I}\right)$. We use the notation $Df(\vec{x})[\vec{h}]$ to denote the directional derivative of f along the direction $\vec{h}$ at the point $\vec{x}$. Using the standard formula for the derivative of $\log\det$, i.e. $\frac{d}{dt}\log\det\mathbf{B}_t = \mathrm{Tr}((\mathbf{B}_t)^{-1}(\frac{d\mathbf{B}_t}{dt}))$, we have

$$\begin{aligned} Df(\vec{s})[\vec{h}] &= \frac{1}{2}\mathrm{Tr}((\mathbf{A}^T\mathbf{S}^{-2}\mathbf{A}+\lambda\mathbf{I})^{-1}(\mathbf{A}^T(-2)\mathbf{S}^{-3}\mathbf{H}\mathbf{A})) \qquad (6.1)\\ &= -\sum_i \frac{h_i}{s_i}\vec{\mathbb{1}}_i^T\mathbf{S}^{-1}\mathbf{A}\left(\mathbf{A}^T\mathbf{S}^{-2}\mathbf{A}+\lambda\mathbf{I}\right)^{-1}\mathbf{A}\mathbf{S}^{-1}\vec{\mathbb{1}}_i = -\sum_i \frac{\psi_i h_i}{s_i} \quad . \end{aligned}$$

Applying chain rules, we have $\nabla f(\vec{x}) = -\mathbf{A}_x^T\vec{\psi}$. Now let $\mathbf{P} \stackrel{\text{def}}{=} \mathbf{S}^{-1}\mathbf{A}\left(\mathbf{A}^T\mathbf{S}^{-2}\mathbf{A}+\lambda\mathbf{I}\right)^{-1}\mathbf{A}^T\mathbf{S}^{-1}$. Taking the derivative of (6.1) again and using the cyclic property of trace, we have

$$\begin{aligned} D^2 f(\vec{s})[\vec{h}_1,\vec{h}_2] &= \mathrm{Tr}\left(\left(\mathbf{A}^T\mathbf{S}^{-2}\mathbf{A}+\lambda\mathbf{I}\right)^{-1}\left(\mathbf{A}^T(-2)\mathbf{S}^{-3}\mathbf{H}_2\mathbf{A}\right)\left(\mathbf{A}^T\mathbf{S}^{-2}\mathbf{A}+\lambda\mathbf{I}\right)^{-1}\left(\mathbf{A}^T\mathbf{S}^{-3}\mathbf{H}_1\mathbf{A}\right)\right)\\ &\quad -\mathrm{Tr}\left(\left(\mathbf{A}^T\mathbf{S}^{-2}\mathbf{A}+\lambda\mathbf{I}\right)^{-1}\left(\mathbf{A}^T(-3)\mathbf{S}^{-4}\mathbf{H}_2\mathbf{H}_1\mathbf{A}\right)\right)\\ &= 3\mathrm{Tr}\left(\mathbf{P}\mathbf{S}^{-2}\mathbf{H}_2\mathbf{H}_1\right) - 2\mathrm{Tr}\left(\mathbf{P}\mathbf{S}^{-1}\mathbf{H}_2\mathbf{P}\mathbf{S}^{-1}\mathbf{H}_1\right)\\ &= 3\sum_i P_{ii}\frac{\vec{h}_1(i)\vec{h}_2(i)}{s_i^2} - 2\sum_{ij}P_{ij}\frac{\vec{h}_2(j)}{s_j}P_{ji}\frac{\vec{h}_2(i)}{s_i}\\ &= 3\sum_i \psi_i\frac{\vec{h}_1(i)\vec{h}_2(i)}{s_i^2} - 2\sum_{ij}P_{ij}^2\frac{\vec{h}_2(j)}{s_j}\frac{\vec{h}_2(i)}{s_i} \quad . \end{aligned}$$

Consequently, $D^2 f(\vec{x})[\vec{\mathbb{1}}_i,\vec{\mathbb{1}}_j] = [\mathbf{S}^{-1}\left(3\mathbf{\Psi}-2\mathbf{P}^{(2)}\right)\mathbf{S}^{-1}]_{ij}$ where $\mathbf{P}^{(2)}$ is the Schur product of $\mathbf{P}$ with itself.

Now note that

$$\begin{aligned} \sum_i P_{ij}^2 &= \vec{\mathbb{1}}_j\mathbf{S}^{-1}\mathbf{A}\left(\mathbf{A}^T\mathbf{S}^{-2}\mathbf{A}+\lambda\mathbf{I}\right)^{-1}\mathbf{A}^T\mathbf{S}^{-2}\mathbf{A}\left(\mathbf{A}^T\mathbf{S}^{-2}\mathbf{A}+\lambda\mathbf{I}\right)^{-1}\mathbf{A}^T\mathbf{S}^{-1}\vec{\mathbb{1}}_j\\ &\le \vec{\mathbb{1}}_j\mathbf{S}^{-1}\mathbf{A}\left(\mathbf{A}^T\mathbf{S}^{-2}\mathbf{A}+\lambda\mathbf{I}\right)^{-1}\mathbf{A}^T\mathbf{S}^{-1}\vec{\mathbb{1}}_j = P_{jj} = \mathbf{\Psi}_{jj} \quad . \end{aligned}$$

Hence, the Gershgorin circle theorem shows that the eigenvalues of $\mathbf{\Psi}-\mathbf{P}^{(2)}$ are lies in union of the interval $[0,2\psi_j]$ over all j. Hence, $\mathbf{\Psi}-\mathbf{P}^{(2)} \succeq \mathbf{0}$. On the other hand, Schur product theorem shows that $\mathbf{P}^{(2)} \succeq \mathbf{0}$ as $\mathbf{P} \succeq \mathbf{0}$. Hence, the result follows by chain rule. □

Lemma 6 immediately shows that under our choice of $\vec{e} = \vec{e}_P(\vec{x},\vec{\tau})$ we can compute the gradient of the hybrid barrier function, $p_{\vec{e}}(\vec{x})$ efficiently. Formally, Lemma 6 immediately implies the following:

Lemma 7 (Gradient)**.** *For* $\vec{x} \in P = \{\vec{y} \in \mathbb{R}^n : \mathbf{A}\vec{y} \ge \vec{b}\}$ *and* $\vec{e} \in \mathbb{R}^m$ *we have*

$$\nabla p_{\vec{e}}(\vec{x}) = -\mathbf{A}_x^T(c_e\vec{\mathbb{1}}+\vec{e}+\vec{\psi}_P(\vec{x})) + \lambda\vec{x}$$

and therefore for all $\vec{\tau} \in \mathbb{R}^m$*, we have*

$$\nabla p_{\vec{e}(\vec{\tau},\vec{x})}(\vec{x}) = -\mathbf{A}_x^T\left(c_e\vec{\mathbb{1}}+\vec{\tau}\right)+\lambda\vec{x}.$$

Remark 8. To be clear, the vector $\nabla p_{\vec{e}(\vec{\tau},\vec{x})}(\vec{x})$ is defined as the vector such that

$$[\nabla p_{\vec{e}(\vec{\tau},\vec{x})}(\vec{x})]_i = \lim_{t\to 0} \frac{1}{t}\left(p_{\vec{e}(\vec{\tau},\vec{x})}(\vec{x}+t\vec{\mathbb{1}}_i) - p_{\vec{e}(\vec{\tau},\vec{x})}(\vec{x})\right) \quad .$$

In other words, we treat the parameter $\vec{e}(\vec{\tau},\vec{x})$ as fixed. This is the reason we denote it by subscript to emphasize that $p_{\vec{e}}(\vec{x})$ is a family of functions, $p_{\vec{e}(\vec{\tau},\vec{x})}$ is one particular function, and $\nabla p_{\vec{e}(\vec{\tau},\vec{x})}$ means taking gradient on that particular function.

Consequently, we can always compute $\nabla p_{\vec{e}(\vec{\tau},\vec{x})}(\vec{x})$ efficiently. Now, we measure *centrality* or how close we are to the hybrid center as follows.

Definition 9 (Centrality)**.** For $\vec{x} \in P = \left\{\vec{y} \in \mathbb{R}^n \,:\, \mathbf{A}\vec{y} \geq \vec{b}\right\}$ and $\vec{e} \in \mathbb{R}^m$, we define the *centrality* of $\vec{x}$ by

$$\delta_{\vec{e}}(\vec{x}) \stackrel{\text{def}}{=} \left\|\nabla p_{\vec{e}}(\vec{x})\right\|_{\mathbf{H}(\vec{x})^{-1}}$$

where $\mathbf{H}(\vec{x}) \stackrel{\text{def}}{=} \mathbf{A}_x^T \left(c_e\mathbf{I} + \mathbf{\Psi}(\vec{x})\right)\mathbf{A}_x + \lambda\mathbf{I}$. Often, we use *weights* $\vec{w} \in \mathbb{R}^m_{>0}$ to approximate this Hessian and consider $\mathbf{Q}(\vec{x},\vec{w}) \stackrel{\text{def}}{=} \mathbf{A}_x^T \left(c_e\mathbf{I} + \mathbf{W}\right)\mathbf{A}_x + \lambda\mathbf{I}$.

Next, we bound how much slacks can change in a region close to a nearly central point.

Lemma 10. *Let* $\vec{x} \in P = \left\{\vec{y} \in \mathbb{R}^n \,:\, \mathbf{A}\vec{y} \geq \vec{b}\right\}$ *and* $\vec{y} \in \mathbb{R}^n$ *such that* $\left\|\vec{x}-\vec{y}\right\|_{\mathbf{H}(\vec{x})} \leq \epsilon\sqrt{c_e + \mu(\vec{x})}$ *for* $\epsilon < 1$*. Then* $\vec{y} \in P$ *and* $(1-\epsilon)\mathbf{S}_x \preceq \mathbf{S}_y \preceq (1+\epsilon)\mathbf{S}_x$.

Proof. Direct calculation reveals the following:

$$\begin{aligned}\left\|\mathbf{S}_x^{-1}(\vec{s}_{\vec{y}} - \vec{s}_{\vec{x}})\right\|_\infty \leq \left\|\mathbf{A}_x(\vec{y}-\vec{x})\right\|_2 &\leq \frac{1}{\sqrt{c_e+\mu(\vec{x})}}\left\|\mathbf{A}_x(\vec{y}-\vec{x})\right\|_{c_e\mathbf{I}+\mathbf{\Psi}(\vec{x})} \\ &\leq \frac{1}{\sqrt{c_e+\mu(\vec{x})}}\left\|\vec{y}-\vec{x}\right\|_{\mathbf{H}(\vec{x})} \leq \epsilon \quad .\end{aligned}$$

Consequently, $(1-\epsilon)\mathbf{S}_x \preceq \mathbf{S}_y \preceq (1+\epsilon)\mathbf{S}_x$. Since $y \in P$ if and only if $\mathbf{S}_y \succeq \mathbf{0}$ the result follows. □

Combining the previous lemmas we obtain the following.

Lemma 11. *Let* $\vec{x} \in P = \{\vec{y} \in \mathbb{R}^n \,:\, \mathbf{A}\vec{y} \geq \vec{b}\}$ *and* $\vec{e},\vec{w} \in \mathbb{R}^m$ *such that* $\left\|\vec{e}\right\|_\infty \leq \frac{1}{2}c_e \leq 1$ *and* $\mathbf{\Psi}(\vec{x}) \preceq \mathbf{W} \preceq \frac{4}{3}\mathbf{\Psi}(\vec{x})$*. If* $\vec{y} \in \mathbb{R}^n$ *satisfies* $\left\|\vec{x}-\vec{y}\right\|_{\mathbf{Q}(\vec{x},\vec{w})} \leq \frac{1}{10}\sqrt{c_e+\mu(\vec{x})}$*, then*

$$\frac{1}{4}\mathbf{Q}(\vec{x},\vec{w}) \preceq \nabla^2 p_{\vec{e}}(\vec{y}) \preceq 8\mathbf{Q}(\vec{x},\vec{w}) \quad \text{and} \quad \frac{1}{2}\mathbf{H}(\vec{x}) \preceq \mathbf{H}(\vec{y}) \preceq 2\mathbf{H}(\vec{x}) \quad .$$

Proof. Lemma 6 shows that

$$\mathbf{A}_y^T\left(c_e\mathbf{I} + \mathbf{E} + \mathbf{\Psi}(\vec{y})\right)\mathbf{A}_y + \lambda\mathbf{I} \preceq \nabla^2 p_{\vec{e}}(\vec{y}) \preceq \mathbf{A}_y^T\left(c_e\mathbf{I} + \mathbf{E} + 3\mathbf{\Psi}(\vec{y})\right)\mathbf{A}_y + \lambda\mathbf{I} \quad . \tag{6.2}$$

Since $\mathbf{W} \succeq \mathbf{\Psi}$, we have that $\mathbf{Q}(\vec{x},\vec{w}) \succeq \mathbf{H}(\vec{x})$ and therefore $\left\|\vec{x}-\vec{y}\right\|_{\mathbf{H}(\vec{x})} \leq \epsilon\sqrt{c_e+\mu(\vec{x})}$ with $\epsilon = 0.1$. Consequently, by Lemma 10 we have $(1-\epsilon)\mathbf{S}_x \preceq \mathbf{S}_y \preceq (1+\epsilon)\mathbf{S}_x$ and therefore

$$\frac{(1-\epsilon)^2}{(1+\epsilon)^2}\mathbf{\Psi}(\vec{x}) \preceq \mathbf{\Psi}(\vec{y}) \preceq \frac{(1+\epsilon)^2}{(1-\epsilon)^2}\mathbf{\Psi}(\vec{x})$$

and

$$\frac{1}{2}\mathbf{H}(\vec{x}) \preceq \frac{(1-\epsilon)^2}{(1+\epsilon)^4}\mathbf{H}(\vec{x}) \preceq \mathbf{H}(\vec{y}) \preceq \frac{(1+\epsilon)^2}{(1-\epsilon)^4}\mathbf{H}(\vec{x}) \preceq 2\mathbf{H}(\vec{x})$$

Furthermore, (6.2) shows that

$$\begin{aligned}\nabla^2 p_{\vec{e}}(\vec{y}) &\preceq \mathbf{A}_y^T \left(c_e \mathbf{I} + \mathbf{E} + 3\mathbf{\Psi}(\vec{y})\right) \mathbf{A}_y + \lambda \mathbf{I} \\ &\preceq \frac{(1+\epsilon)^2}{(1-\epsilon)^4} \mathbf{A}_x^T \left(c_e \mathbf{I} + \mathbf{E} + 3\mathbf{\Psi}(\vec{x})\right) \mathbf{A}_x + \lambda \mathbf{I} \\ &\preceq \frac{(1+\epsilon)^2}{(1-\epsilon)^4} \mathbf{A}_x^T \left(\frac{3}{2} c_e \mathbf{I} + 3\mathbf{W}\right) \mathbf{A}_x + \lambda \mathbf{I} \\ &\preceq 3\frac{(1+\epsilon)^2}{(1-\epsilon)^4} \mathbf{Q}(\vec{x}, \vec{w}) \preceq 8\mathbf{Q}(\vec{x}, \vec{w})\end{aligned}$$

and

$$\begin{aligned}\nabla^2 p_{\vec{e}}(\vec{y}) &\succeq \mathbf{A}_y^T \left(c_e \mathbf{I} + \mathbf{E} + \mathbf{\Psi}(\vec{y})\right) \mathbf{A}_y + \lambda \mathbf{I} \\ &\succeq \frac{(1-\epsilon)^4}{(1+\epsilon)^2} \mathbf{A}_x^T \left(c_e \mathbf{I} + \mathbf{E} + \mathbf{\Psi}(\vec{x})\right) \mathbf{A}_x + \lambda \mathbf{I} \\ &\succeq \frac{(1-\epsilon)^4}{(1+\epsilon)^2} \mathbf{A}_x^T \left(\frac{1}{2} c_e \mathbf{I} + \frac{3}{4}\mathbf{W}\right) \mathbf{A}_x + \lambda \mathbf{I} \\ &\succeq \frac{1}{2}\frac{(1-\epsilon)^4}{(1+\epsilon)^2} \mathbf{Q}(\vec{x}, \vec{w}) \succeq \frac{1}{4}\mathbf{Q}(\vec{x}, \vec{w}).\end{aligned}$$

□

To analyze our centering scheme we use standard facts about gradient descent we prove in Lemma 12.

Lemma 12 (Gradient Descent)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be twice differentiable and $\mathbf{Q} \in \mathbb{R}^{n \times n}$ be positive definite. Let $\vec{x}_0 \in \mathbb{R}^n$ and $\vec{x}_1 \stackrel{\text{def}}{=} \vec{x}_0 - \frac{1}{L}\mathbf{Q}^{-1}\nabla f(\vec{x}_0)$. Furthermore, let $\vec{x}_\alpha = \vec{x}_0 + \alpha(\vec{x}_1 - \vec{x})$ and suppose that $\mu\mathbf{Q} \preceq \nabla^2 f(\vec{x}_\alpha) \preceq L\mathbf{Q}$ for all $\alpha \in [0, 1]$. Then,*

1. $\left\|\nabla f(\vec{x}_1)\right\|_{\mathbf{Q}^{-1}} \leq \left(1 - \frac{\mu}{L}\right) \left\|\nabla f(\vec{x}_0)\right\|_{\mathbf{Q}^{-1}}$
2. $f(\vec{x}_1) \geq f(\vec{x}_0) - \frac{1}{L}\left\|\nabla f(\vec{x}_0)\right\|_{\mathbf{Q}^{-1}}^2$

Proof. Integrating we have that

$$\nabla f(\vec{x}_1) = \nabla f(\vec{x}_0) + \int_0^1 \nabla^2 f(\vec{x}_\alpha)(\vec{x}_1 - \vec{x}_0) d\alpha = \int_0^1 \left(\mathbf{Q} - \frac{1}{L}\nabla^2 f(\vec{x}_\alpha)\right) \mathbf{Q}^{-1} \nabla f(\vec{x}_0) d\alpha$$

Consequently, by applying Jensen's inequality we have

$$\begin{aligned}\left\|\nabla f(\vec{x}_1)\right\|_{\mathbf{Q}^{-1}} &= \left\|\int_0^1 \left(\mathbf{Q} - \frac{1}{L}\nabla^2 f(\vec{x}_\alpha)\right) \mathbf{Q}^{-1} \nabla f(\vec{x}_0) d\alpha\right\|_{\mathbf{Q}^{-1}} \\ &\leq \int_0^1 \left\|\left(\mathbf{Q} - \frac{1}{L}\nabla^2 f(\vec{x}_\alpha)\right) \mathbf{Q}^{-1} \nabla f(\vec{x}_0)\right\|_{\mathbf{Q}^{-1}} d\alpha \\ &\leq \left\|\mathbf{Q}^{-1/2} \nabla f(\vec{x}_0)\right\|_{\left[\mathbf{Q}^{-1/2}\left(\mathbf{Q} - \frac{1}{L}\nabla^2 f(\vec{x}_\alpha)\right)\mathbf{Q}^{-1/2}\right]^2}\end{aligned}$$

Now we know that by assumption that

$$\mathbf{0} \preceq \mathbf{Q}^{-1/2}\left(\mathbf{Q} - \frac{1}{L}\nabla^2 f(\vec{x}_\alpha)\right)\mathbf{Q}^{-1/2} \preceq \left(1 - \frac{\mu}{L}\right)\mathbf{I}$$

and therefore combining these (1) holds.

Using the convexity of f, we have

$$\begin{aligned} f(\vec{x}_1) &\geq f(\vec{x}_0) + \langle \nabla f(\vec{x}_0), \vec{x}_1 - \vec{x}_0 \rangle \\ &\geq f(\vec{x}_0) - \left\|\nabla f(\vec{x}_0)\right\|_{\mathbf{Q}^{-1}} \left\|\vec{x}_1 - \vec{x}_0\right\|_{\mathbf{Q}} \end{aligned}$$

and since $\left\|\vec{x}_1 - \vec{x}_0\right\|_{\mathbf{Q}} = \frac{1}{L}\left\|\nabla f(\vec{x}_0)\right\|_{\mathbf{Q}^{-1}}$, (2) holds as well. □

Next we bound the effect of changing $\vec{e}$ on the hybrid barrier function $p_{\vec{e}}(\vec{x})$.

Lemma 13. *For $\vec{x} \in P = \{\vec{y} \in \mathbb{R}^n : \mathbf{A}\vec{y} \geq \vec{b}\}$, $\vec{e}, \vec{f} \in \mathbb{R}^m$, and $\vec{w} \in \mathbb{R}^m_{>0}$ such that $\mathbf{W} \succeq \mathbf{\Psi}_x$*

$$\left\|\nabla p_{\vec{f}}(\vec{x})\right\|_{\mathbf{Q}(\vec{x},\vec{w})^{-1}} \leq \left\|\nabla p_{\vec{e}}(\vec{x})\right\|_{\mathbf{Q}(\vec{x},\vec{w})^{-1}} + \frac{1}{\sqrt{c_e + \mu(\vec{x})}}\left\|\vec{f} - \vec{e}\right\|_2$$

Proof. Direct calculation shows the following

$$\begin{aligned} \left\|\nabla p_{\vec{f}}(\vec{x})\right\|_{\mathbf{Q}(\vec{x},\vec{w})^{-1}} &= \left\| - \mathbf{A}_x^T(c_e\vec{\mathbb{1}} + \vec{f} + \vec{\psi}_P(\vec{x})) + \lambda\vec{x}\right\|_{\mathbf{Q}(\vec{x},\vec{w})^{-1}} && \text{(Formula for } \nabla p_{\vec{f}}(\vec{x})) \\ &\leq \left\|\nabla p_{\vec{e}}(\vec{x})\right\|_{\mathbf{Q}(\vec{x},\vec{w})^{-1}} + \left\|\mathbf{A}_x^T(\vec{f} - \vec{e})\right\|_{\mathbf{Q}(\vec{x},\vec{w})^{-1}} && \text{(Triangle inequality)} \\ &\leq \left\|\nabla p_{\vec{e}}(\vec{x})\right\|_{\mathbf{Q}(\vec{x},\vec{w})^{-1}} + \frac{1}{\sqrt{c_e + \mu(\vec{x})}}\left\|\mathbf{A}_x^T(\vec{f} - \vec{e})\right\|_{(\mathbf{A}_x^T\mathbf{A}_x)^{-1}} && \text{(Bound on } \mathbf{Q}(\vec{x},\vec{w})) \\ &\leq \left\|\nabla p_{\vec{e}}(\vec{x})\right\|_{\mathbf{Q}(\vec{x},\vec{w})^{-1}} + \frac{1}{\sqrt{c_e + \mu(\vec{x})}}\left\|\vec{f} - \vec{e}\right\|_2 && \text{(Property of projection matrix)} \end{aligned}$$

where in the second to third line we used $\mathbf{Q}(\vec{x},\vec{w}) \succeq \mathbf{H}(\vec{x}) \succeq (c_e + \mu(\vec{x}))\mathbf{A}_x^T\mathbf{A}_x$. □

We now have everything we need to analyze our centering algorithm.

Algorithm 1: $(\vec{x}^{(r)}, \vec{\tau}^{(r)}) = \texttt{Centering}(\vec{x}^{(0)}, \vec{\tau}^{(0)}, r, c_\Delta)$

Input: Initial point $\vec{x}^{(0)} \in P = \{\vec{y} \in \mathbb{R}^n : \mathbf{A}\vec{y} \geq \vec{b}\}$, Estimator of leverage scores $\vec{\tau}^{(0)} \in \mathbb{R}^n$
Input: Number of iterations $r > 0$, Accuracy of the estimator $0 \leq c_\Delta \leq 0.01c_e$.
Given: $\left\|\vec{e}^{(0)}\right\|_\infty \leq \frac{1}{3}c_e \leq \frac{1}{3}$ where $\vec{e}^{(0)} = \vec{e}(\vec{\tau}^{(0)}, \vec{x}^{(0)})$.
Given: $\delta_{\vec{e}^{(0)}}(\vec{x}^{(0)}) = \left\|\nabla p_{\vec{e}^{(0)}}(\vec{x}^{(0)})\right\|_{\mathbf{H}(\vec{x}^{(0)})^{-1}} \leq \frac{1}{100}\sqrt{c_e + \mu(\vec{x}^{(0)})}$.
Compute $\vec{w}$ such that $\mathbf{\Psi}(\vec{x}^{(0)}) \preceq \mathbf{W} \preceq \frac{4}{3}\mathbf{\Psi}(\vec{x}^{(0)})$ (See Lemma 5)
Let $\mathbf{Q} \stackrel{\text{def}}{=} \mathbf{Q}(\vec{x}^{(0)}, \vec{w})$.
for $k = 1$ *to* r **do**
  $\vec{x}^{(k)} := \vec{x}^{(k-1)} - \frac{1}{8}\mathbf{Q}^{-1}\nabla p_{\vec{e}^{(k-1)}}(\vec{x}^{(k-1)})$.
  Sample $\vec{\Delta}^{(k)} \in \mathbb{R}^n$ s.t.
    $\mathbb{E}[\vec{\Delta}^{(k)}] = \vec{\psi}(\vec{x}^{(k)}) - \vec{\psi}(\vec{x}^{(k-1)})$ and
    with high probability in n,
  $\left\|\vec{\Delta}^{(k)} - (\vec{\psi}(\vec{x}^{(k)}) - \vec{\psi}(\vec{x}^{(k-1)}))\right\|_2 \leq c_\Delta\left\|\mathbf{S}^{-1}_{\vec{x}^{(k-1)}}(\vec{s}_{\vec{x}^{(k)}} - \vec{s}_{\vec{x}^{(k-1)}})\right\|_2$ (See Section 7.1)
  $\vec{\tau}^{(k)} := \vec{\tau}^{(k-1)} + \vec{\Delta}^{(k)}$.
  $\vec{e}^{(k)} := \vec{e}(\vec{\tau}^{(k)}, \vec{x}^{(k)})$.
end
Output: $(\vec{x}^{(r)}, \vec{\tau}^{(r)})$

Lemma 14. *Let $\vec{x}^{(0)} \in P = \{\vec{y} \in \mathbb{R}^n : \mathbf{A}\vec{y} \geq \vec{b}\}$ and let $\vec{\tau}^{(0)} \in \mathbb{R}^m$ such that $\left\|\vec{e}(\vec{\tau}^{(0)}, \vec{x}^{(0)})\right\|_\infty \leq \frac{1}{3}c_e \leq \frac{1}{3}$. Assume that r is a positive integer, $0 \leq c_\Delta \leq 0.01 c_e$ and $\delta_{\vec{e}^{(0)}}(\vec{x}^{(0)}) \leq \frac{1}{100}\sqrt{c_e + \mu(\vec{x}^{(0)})}$. With high probability in n, the algorithm* `Centering`$(\vec{x}^{(0)}, \vec{\tau}^{(0)}, r, c_\Delta)$ *outputs $(\vec{x}^{(r)}, \vec{\tau}^{(r)})$ such that*

1. $\delta_{\vec{e}^{(r)}}(\vec{x}^{(r)}) \leq 2\left(1 - \frac{1}{64}\right)^r \delta_{\vec{e}^{(0)}}(\vec{x}^{(0)})$.
2. $\mathbb{E}[p_{\vec{e}^{(k)}}(\vec{x}^{(r)})] \geq p_{\vec{e}^{(0)}}(\vec{x}^{(0)}) - 8\left(\delta_{\vec{e}^{(0)}}(\vec{x}^{(0)})\right)^2$.
3. $\mathbb{E}\vec{e}^{(r)} = \vec{e}^{(0)}$ *and* $\left\|\vec{e}^{(r)} - \vec{e}^{(0)}\right\|_2 \leq \frac{1}{10}c_\Delta$.
4. $\left\|\mathbf{S}^{-1}_{\vec{x}^{(0)}}(\vec{s}(\vec{x}^{(r)}) - \vec{s}(\vec{x}^{(0)}))\right\|_2 \leq \frac{1}{10}$.

where $\vec{e}^{(r)} = \vec{e}(\vec{\tau}^{(r)}, \vec{x}^{(r)})$.

Proof. Let $\eta = \left\|\nabla p_{\vec{e}^{(0)}}(\vec{x}^{(0)})\right\|_{\mathbf{Q}^{-1}}$. First, we use induction to prove that $\left\|\vec{x}^{(r)} - \vec{x}^{(0)}\right\|_{\mathbf{Q}} \leq 8\eta$, $\left\|\nabla p_{\vec{e}^{(r)}}(\vec{x}^{(r)})\right\|_{\mathbf{Q}^{-1}} \leq \left(1 - \frac{1}{64}\right)^r \eta$ and $\left\|\vec{e}^{(r)} - \vec{e}^{(0)}\right\|_2 \leq \frac{1}{10}c_\Delta$ for all r.

Clearly the claims hold for $r = 0$. We now suppose they hold for all $r \leq t$ and show that they hold for $r = t + 1$. Now, since $\left\|\vec{x}^{(t)} - \vec{x}^{(0)}\right\|_{\mathbf{Q}} \leq 8\eta$, $\vec{x}^{(t+1)} = \vec{x}^{(t)} - \frac{1}{8}\mathbf{Q}^{-1}\nabla p_{\vec{e}^{(t)}}(\vec{x}^{(t)})$, and $\left\|\nabla p_{\vec{e}^{(t)}}(\vec{x}^{(t)})\right\|_{\mathbf{Q}^{-1}} \leq \left(1 - \frac{1}{64}\right)^t \eta \leq \eta$, we have

$$\left\|\vec{x}^{(t+1)} - \vec{x}^{(0)}\right\|_{\mathbf{Q}} \leq \left\|\vec{x}^{(t)} - \vec{x}^{(0)}\right\|_{\mathbf{Q}} + \frac{1}{8}\left\|\nabla p_{\vec{e}^{(t)}}(\vec{x}^{(t)})\right\|_{\mathbf{Q}^{-1}} \leq 9\eta.$$

We will improve this estimate later in the proof to finish the induction on $\left\|\vec{x}^{(t+1)} - \vec{x}^{(0)}\right\|_{\mathbf{Q}}$, but using this, $\eta \leq 0.01\sqrt{c_e + \mu(\vec{x}^{(0)})}$, and $\left\|\vec{e}^{(t)}\right\|_\infty \leq \left\|\vec{e}^{(t)} - \vec{e}^{(0)}\right\|_\infty + \left\|\vec{e}^{(0)}\right\|_\infty \leq \frac{c_e}{2}$, we can invoke Lemma 11 and Lemma 12 and therefore

$$\left\|\nabla p_{\vec{e}^{(t)}}(\vec{x}^{(t+1)})\right\|_{\mathbf{Q}^{-1}} \leq \left(1 - \frac{1}{32}\right)\left\|\nabla p_{\vec{e}^{(t)}}(\vec{x}^{(t)})\right\|_{\mathbf{Q}^{-1}}.$$

By Lemma 13 we have

$$\left\|\nabla p_{\vec{e}^{(t+1)}}(\vec{x}^{(t+1)})\right\|_{\mathbf{Q}^{-1}} \leq \left(1 - \frac{1}{32}\right)\left\|\nabla p_{\vec{e}^{(t)}}(\vec{x}^{(t)})\right\|_{\mathbf{Q}^{-1}} + \frac{1}{\sqrt{c_e + \mu(\vec{x}^{(0)})}}\left\|\vec{e}^{(t+1)} - \vec{e}^{(t)}\right\|_2. \tag{6.3}$$

To bound $\left\|\vec{e}^{(t+1)} - \vec{e}^{(t)}\right\|_2$, we note that Lemma 10 and the induction hypothesis $\left\|\vec{x}^{(t)} - \vec{x}^{(0)}\right\|_{\mathbf{H}(\vec{x}^{(0)})} \leq \left\|\vec{x}^{(t)} - \vec{x}^{(0)}\right\|_{\mathbf{Q}} \leq 8\eta$ shows that $(1 - 0.1)\mathbf{S}_{x^{(0)}} \preceq \mathbf{S}_{x^{(t)}} \preceq (1 + 0.1)\mathbf{S}_{x^{(0)}}$ and therefore

$$\begin{aligned}
\left\|\mathbf{S}^{-1}_{x^{(t)}}(\vec{s}_{x^{(t)}} - \vec{s}_{x^{(t+1)}})\right\|_2 &\leq \frac{1}{1 - 0.1}\left\|\mathbf{S}^{-1}_{x^{(0)}}\mathbf{A}\left(\vec{x}^{(t)} - \vec{x}^{(t+1)}\right)\right\|_2 \\
&= \frac{1}{1 - 0.1}\left\|\frac{1}{8}\mathbf{Q}^{-1}\nabla p_{\vec{e}^{(t)}}(\vec{x}^{(t)})\right\|_{\mathbf{A}^T\mathbf{S}^{-2}_{x^{(0)}}\mathbf{A}} \\
&\leq \frac{1}{8\,(1 - 0.1)\sqrt{c_e + \mu(\vec{x}^{(0)})}}\left\|\nabla p_{\vec{e}^{(t)}}(\vec{x}^{(t)})\right\|_{\mathbf{Q}^{-1}}
\end{aligned} \tag{6.4}$$

Now since

$$\begin{aligned}
\vec{e}^{(t+1)} - \vec{e}^{(t)} &= \left(\vec{\tau}^{(t+1)} - \vec{\psi}(\vec{x}^{(t+1)})\right) - \left(\vec{\tau}^{(t)} - \vec{\psi}(\vec{x}^{(t)})\right) \\
&= \vec{\Delta}^{(t+1)} - \left(\vec{\psi}(\vec{x}^{(t+1)}) - \vec{\psi}(\vec{x}^{(t)})\right)
\end{aligned}$$

Consequently, with high probability in n,

$$\begin{aligned}\left\|\vec{e}^{(t+1)}-\vec{e}^{(t)}\right\|_2 &= \left\|\vec{\Delta}^{(t+1)}-\left(\vec{\psi}(\vec{x}^{(t+1)})-\vec{\psi}(\vec{x}^{(t)})\right)\right\|_2 \\ &\leq c_\Delta\left\|\mathbf{S}^{-1}_{x^{(t)}}(\vec{s}_{x^{(t+1)}}-\vec{s}_{x^{(t)}})\right\|_2 \\ &\leq \frac{c_\Delta}{8\left(1-0.1\right)\sqrt{c_e+\mu(\vec{x}^{(0)})}}\left\|\nabla p_{\vec{e}^{(t)}}(\vec{x}^{(t)})\right\|_{\mathbf{Q}^{-1}} \quad .\end{aligned}$$

where in the last line we used $\min_{i\in[m]} w_i \geq \mu(\vec{x}^{(0)})$. Since $c_\Delta < 0.01c_e$, by (6.3), we have

$$\begin{aligned}\left\|\nabla p_{\vec{e}^{(t+1)}}(\vec{x}^{(t+1)})\right\|_{\mathbf{Q}^{-1}} &\leq \left(1-\frac{1}{32}\right)\left\|\nabla p_{\vec{e}^{(t)}}(\vec{x}^{(t)})\right\|_{\mathbf{Q}^{-1}}+\frac{0.01c_e}{8\left(1-0.1\right)\left(c_e+\mu(\vec{x}^{(0)})\right)}\left\|\nabla p_{\vec{e}^{(t)}}(\vec{x}^{(t)})\right\|_{\mathbf{Q}^{-1}} \\ &\leq \left(1-\frac{1}{64}\right)\left\|\nabla p_{\vec{e}^{(t)}}(\vec{x}^{(t)})\right\|_{\mathbf{Q}^{-1}} \quad .\end{aligned}$$

Furthermore, this implies that

$$\left\|\vec{x}^{(t+1)}-\vec{x}^{(0)}\right\|_{\mathbf{Q}} \leq \left\|\sum_{k=0}^{t}\frac{1}{8}\mathbf{Q}^{-1}\nabla p_{\vec{e}^{(k)}}(\vec{x}^{(k)})\right\|_{\mathbf{Q}^{-1}} \leq \frac{1}{8}\sum_{i=0}^{\infty}\left(1-\frac{1}{64}\right)^k\eta \leq \frac{64}{8}\eta = 8\eta \quad .$$

Similarly, we have that

$$\begin{aligned}\left\|\vec{e}^{(t+1)}-\vec{e}^{(0)}\right\|_2 &\leq \sum_{k=0}^{t}\frac{c_\Delta}{8\left(1-0.1\right)\sqrt{c_e+\mu(\vec{x}^{(0)})}}\left(1-\frac{1}{64}\right)^k\left\|\nabla p_{\vec{e}^{(0)}}(\vec{x}^{(0)})\right\|_{\mathbf{Q}^{-1}} \\ &\leq \frac{8c_\Delta\eta}{(1-0.1)\sqrt{c_e+\mu(\vec{x}^{(0)})}} \leq \frac{8c_\Delta}{(1-0.1)\sqrt{c_e+\mu(\vec{x}^{(0)})}}\delta_{\vec{e}^{(0)}}(\vec{x}^{(0)}) \leq \frac{1}{10}c_\Delta\end{aligned}$$

where we used $\eta = \left\|\nabla p_{\vec{e}^{(0)}}(\vec{x}^{(0)})\right\|_{\mathbf{Q}^{-1}} \leq \left\|\nabla p_{\vec{e}^{(0)}}(\vec{x}^{(0)})\right\|_{\mathbf{H}^{-1}} = \delta_{\vec{e}^{(0)}}(\vec{x}^{(0)})$ and this finishes the induction on $\left\|\nabla p_{\vec{e}^{(t)}}(\vec{x}^{(t)})\right\|_{\mathbf{Q}^{-1}}$, $\left\|\vec{x}^{(t)}-\vec{x}^{(0)}\right\|_{\mathbf{Q}}$ and $\left\|\vec{e}^{(t)}-\vec{e}^{(0)}\right\|_2$.

Hence, for all r, Lemma 11 shows that

$$\begin{aligned}\delta_{\vec{e}^{(r)}}(\vec{x}^{(r)}) &= \left\|\nabla p_{\vec{e}^{(r)}}(\vec{x}^{(r)})\right\|_{\mathbf{H}(\vec{x}^{(r)})^{-1}} \leq \sqrt{2}\left\|\nabla p_{\vec{e}^{(r)}}(\vec{x}^{(r)})\right\|_{\mathbf{H}(\vec{x}^{(0)})^{-1}} \\ &\leq \sqrt{\frac{8}{3}}\left\|\nabla p_{\vec{e}^{(r)}}(\vec{x}^{(r)})\right\|_{\mathbf{Q}^{-1}} \leq \sqrt{\frac{8}{3}}\left(1-\frac{1}{64}\right)^r\left\|\nabla p_{\vec{e}^{(0)}}(\vec{x}^{(0)})\right\|_{\mathbf{Q}^{-1}} \\ &\leq 2\left(1-\frac{1}{64}\right)^r\delta_{\vec{e}^{(0)}}(\vec{x}^{(0)}).\end{aligned}$$

Using that $\mathbb{E}\vec{e}^{(t+1)} = \vec{e}^{(t)}$, we see that the expected change in function value is only due to the change while taking centering steps and therefore Lemma 12 shows that

$$\mathbb{E}[p_{\vec{e}^{(r)}}(\vec{x}^{(r)})] \geq p_{\vec{e}^{(0)}}(\vec{x}^{(0)})-\frac{1}{8}\sum_{k=0}^{\infty}\left(1-\frac{1}{64}\right)^{2k}\left\|\nabla p_{\vec{e}^{(0)}}(\vec{x}^{(0)})\right\|^2_{\mathbf{Q}^{-1}} \geq p_{\vec{e}^{(0)}}(\vec{x}^{(0)})-8\left(\delta_{\vec{e}^{(0)}}(\vec{x}^{(0)})\right)^2.$$

Finally, for (4), we note that

$$\left\|\frac{s(\vec{x}^{(r)})-s(\vec{x}^{(0)})}{s(\vec{x}^{(0)})}\right\|_2 = \left\|\vec{x}^{(r)}-\vec{x}^{(0)}\right\|_{\mathbf{A}^T\mathbf{S}^{-2}_{x^{(0)}}\mathbf{A}} \leq \frac{1}{\sqrt{\mu(\vec{x}^{(0)})+c_e}}\left\|\vec{x}^{(r)}-\vec{x}^{(0)}\right\|_{\mathbf{Q}^{-1}} \leq \frac{1}{10}.$$

□

6.2 Changing Constraints

Here we bound the effect that adding or a removing a constraint has on the hybrid barrier function. Much of the analysis in this section follows from the following lemma which follows easily from the Sherman Morrison Formula.

Lemma 15 (Sherman Morrison Formula Implications)**.** *Let $\mathbf{B} \in \mathbb{R}^{n \times n}$ be an invertible symmetric matrix and let $\vec{a} \in \mathbb{R}^n$ be arbitrary vector satisfying $\vec{a}^T \mathbf{B}^{-1} \vec{a} < 1$. The following hold:*

1. $\left(\mathbf{B} \pm \vec{a}\vec{a}^T\right)^{-1} = \mathbf{B}^{-1} \mp \frac{\mathbf{B}^{-1}\vec{a}\vec{a}^T\mathbf{B}^{-1}}{1 \pm \vec{a}^T\mathbf{B}^{-1}\vec{a}}$.
2. $\mathbf{0} \preceq \frac{\mathbf{B}^{-1}\vec{a}\vec{a}^T\mathbf{B}^{-1}}{1 \pm \vec{a}^T\mathbf{B}^{-1}\vec{a}} \preceq \frac{\vec{a}^T\mathbf{B}^{-1}\vec{a}}{1 \pm \vec{a}^T\mathbf{B}^{-1}\vec{a}}\mathbf{B}^{-1}$.
3. $\log\det\left(\mathbf{B} \pm \vec{a}\vec{a}^T\right) = \ln\det\mathbf{B} + \ln\left(1 \pm \vec{a}^T\mathbf{B}^{-1}\vec{a}\right)$.

Proof. (1) follows immediately from Sherman Morrison [95]. (2) follows since $\vec{a}\vec{a}^T$ is PSD,

$$\frac{\mathbf{B}^{-1}\vec{a}\vec{a}^T\mathbf{B}^{-1}}{1 \pm \vec{a}^T\mathbf{B}^{-1}\vec{a}} = \mathbf{B}^{-1/2}\left[\frac{\mathbf{B}^{-1/2}\vec{a}\vec{a}^T\mathbf{B}^{-1/2}}{1 \pm \vec{a}^T\mathbf{B}^{-1}\vec{a}}\right]\mathbf{B}^{-1/2} \quad ,$$

and $\vec{y}\vec{y}^T \preceq \left\|\vec{y}\right\|_2^2\mathbf{I}$ for any vector $\vec{y}$. (3) follows immediately from the Matrix Determinant Lemma. □

We also make use of the following technical helper lemma.

Lemma 16. *For $\mathbf{A} \in \mathbb{R}^{n \times m}$ and all $\vec{a} \in \mathbb{R}^n$ we have*

$$\sum_{i \in [m]} \frac{1}{\psi_{\mathbf{A}}[i]}\left(\mathbf{A}\left(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I}\right)^{-1}\vec{a}\right)_i^4 \leq \left(\vec{a}^T\left(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I}\right)^{-1}\vec{a}\right)^2$$

Proof. We have by Cauchy Schwarz that

$$\left(\vec{\mathbb{1}}_i^T\mathbf{A}\left(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I}\right)^{-1}\vec{a}\right)^2 \leq \psi_{\mathbf{A}}[i] \cdot \vec{a}^T\left(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I}\right)^{-1}\vec{a}$$

and consequently

$$\sum_{i \in [m]} \frac{\left(\vec{\mathbb{1}}_i^T\mathbf{A}\left(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I}\right)^{-1}\vec{a}\right)^4}{\psi_{\mathbf{A}}[i]} \leq \left(\vec{a}^T\left(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I}\right)^{-1}\vec{a}\right)\sum_{i \in [m]}\left(\vec{\mathbb{1}}_i\mathbf{A}\left(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I}\right)^{-1}\vec{a}\right)^2 .$$

Since

$$\begin{aligned}\sum_{i \in [m]}\left(\vec{\mathbb{1}}_i^T\mathbf{A}\left(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I}\right)^{-1}\vec{a}\right)^2 &= \vec{a}^T\left(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I}\right)^{-1}\mathbf{A}^T\mathbf{A}\left(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I}\right)^{-1}\vec{a} \\ &\leq \vec{a}^T\left(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I}\right)^{-1}\vec{a},\end{aligned}$$

we have the desired result. □

We now bound the effect of adding a constraint.

Lemma 17. *Let* $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\vec{b} \in \mathbb{R}^m$, $\vec{\tau} \in \mathbb{R}^m$, *and* $\vec{x} \in P \stackrel{\text{def}}{=} \left\{\vec{y} \in \mathbb{R}^n : \mathbf{A}\vec{y} \geq \vec{b}\right\}$. *Let* $\overline{\mathbf{A}} \in \mathbb{R}^{(m+1)\times n}$ *be* $\mathbf{A}$ *with a row* $\vec{a}_{m+1}$ *added, let* $\overline{b} \in \mathbb{R}^{m+1}$ *be the vector* $\vec{b}$ *with an entry* b_{m+1} *added, and let* $\overline{P} \stackrel{\text{def}}{=} \left\{\vec{y} \in \mathbb{R}^n : \overline{\mathbf{A}}\vec{y} \geq \overline{b}\right\}$. *Let* $s_{m+1} = \vec{a}_{m+1}^T\vec{x} - b_{m+1} > 0$, $\psi_a = \frac{\vec{a}_{m+1}^T(\mathbf{A}_x^T\mathbf{A}_x+\lambda\mathbf{I})^{-1}\vec{a}_{m+1}}{s_{m+1}^2}$.

Now, let $\vec{\upsilon} \in \mathbb{R}^{m+1}$ *be defined so that* $\upsilon_{m+1} = \frac{\psi_a}{1+\psi_a}$ *and for all* $i \in [m]$

$$\upsilon_i = \tau_i - \frac{1}{1+\psi_a}\left[\mathbf{A}_x\left(\mathbf{A}_x^T\mathbf{A}_x + \lambda\mathbf{I}\right)^{-1}\frac{\vec{a}_{m+1}}{s_{m+1}}\right]_i^2.$$

Then, the following hold

- [Leverage Score Estimation] $e_{\overline{P}}(\vec{\upsilon},\vec{x})_{m+1} = 0$ and $e_{\overline{P}}(\vec{\upsilon},\vec{x})_i = e_P(\vec{\tau},\vec{x})_i$ for all $i \in [m]$.
- [Function Value Increase] $p_{\vec{e}_{\overline{P}}(\vec{\upsilon},\vec{x})}(\vec{x}) = p_{\vec{e}_P(\vec{\tau},\vec{x})}(\vec{x}) - c_e \ln s(\vec{x})_{m+1} + \ln(1+\psi_a)$.
- [Centrality Increase] $\delta_{\vec{e}_{\overline{P}}(\vec{\upsilon},\vec{x})}(\vec{x}) \leq \delta_{\vec{e}_P(\vec{\upsilon},\vec{x})}(\vec{x}) + (c_e+\psi_a)\sqrt{\frac{\psi_a}{\mu(\vec{x})}} + \psi_a$.

Proof. By (1) in Lemma 15, we have that for all $i \in [m]$

$$\psi_{\overline{P}}(\vec{x})_i = \psi_P(\vec{x})_i - \frac{1}{1+\psi_a}\left[\mathbf{A}_x\left(\mathbf{A}_x^T\mathbf{A}_x + \lambda\mathbf{I}\right)^{-1}\frac{\vec{a}_{m+1}}{s_{m+1}}\right]_i^2$$

and that

$$\psi_{\overline{P}}(\vec{x})_{m+1} = \psi_a - \frac{\psi_a^2}{1+\psi_a} = \frac{\psi_a}{1+\psi_a}.$$

Consequently [Leverage Score Estimation] holds. Furthermore, by (3) in Lemma 15 this then implies that [Function Value Change] holds.

To bound the change in centrality note that by (2) in Lemma 15 we have that $\overline{\mathbf{H}}^{-1} \preceq \mathbf{H}^{-1}$. Therefore if let $\vec{v}' \in \mathbb{R}^m$ be defined so that $\vec{v}'_i = \vec{v}_i$ for all $i \in [m]$ then by triangle inequality we have

$$\begin{aligned}
\delta_{\vec{e}_{\overline{P}}(\vec{\upsilon},\vec{x})}(\vec{x}) &= \left\|\overline{\mathbf{A}}_x^T(c_e\vec{\mathbb{1}} + \vec{v})\right\|_{\overline{\mathbf{H}}^{-1}} \leq \left\|\overline{\mathbf{A}}_x^T(c_e\vec{\mathbb{1}} + \vec{v})\right\|_{\mathbf{H}^{-1}} \\
&\leq \left\|\mathbf{A}_x^T(c_e\vec{\mathbb{1}} + \vec{\tau})\right\|_{\mathbf{H}^{-1}} + \left\|\frac{\vec{a}_{m+1}}{s_{m+1}}(c_e + \upsilon_{m+1})\right\|_{\mathbf{H}^{-1}} + \left\|\mathbf{A}_x^T(\vec{v}' - \vec{\tau})\right\|_{\mathbf{H}^{-1}} \\
&= \delta_{\vec{e}_P(\vec{\tau},\vec{x})}(\vec{x}) + \left(c_e + \frac{\psi_a}{1+\psi_a}\right)\left\|\frac{\vec{a}_{m+1}}{s_{m+1}}\right\|_{\mathbf{H}^{-1}} + \left\|\mathbf{A}_x^T(\vec{v}' - \vec{\tau})\right\|_{\mathbf{H}^{-1}}
\end{aligned}$$

Now, since $\mathbf{H}^{-1} \preceq \frac{1}{\mu(\vec{x})}\left(\mathbf{A}_x^T\mathbf{A}_x + \lambda\mathbf{I}\right)^{-1}$, we have that

$$\left\|\frac{\vec{a}_{m+1}}{s_{m+1}}\right\|_{\mathbf{H}^{-1}} \leq \frac{1}{\sqrt{\mu(\vec{x})}}\left\|\frac{\vec{a}_{m+1}}{s_{m+1}}\right\|_{(\mathbf{A}_x^T\mathbf{A}_x+\lambda\mathbf{I})^{-1}} = \sqrt{\frac{\psi_a}{\mu(\vec{x})}}.$$

Since $\mathbf{\Psi}^{1/2}\mathbf{A}_x\left(\mathbf{A}_x^T\mathbf{\Psi}\mathbf{A}_x\right)^{-1}\mathbf{A}_x^T\mathbf{\Psi}^{1/2}$ is a projection matrix, we have $\mathbf{\Psi}^{-1} \succeq \mathbf{A}_x\left(\mathbf{A}_x^T\mathbf{\Psi}\mathbf{A}_x\right)^{-1}\mathbf{A}_x^T \succeq$

$\mathbf{A}_x\mathbf{H}^{-1}\mathbf{A}_x^T$. By Lemma 16, we have

$$\begin{aligned}\left\|\mathbf{A}_x^T\left(\vec{\tau}'-\vec{v}\right)\right\|_{\mathbf{H}^{-1}}^2 &\leq \left\|\vec{\tau}'-\vec{v}\right\|_{\mathbf{\Psi}^{-1}}^2 \\ &= \sum_{i\in[m]}\frac{1}{\psi(\vec{x})_i}\left(\frac{1}{1+\psi_a}\left(\vec{\mathbb{1}}_i\mathbf{A}_x\left(\mathbf{A}_x^T\mathbf{A}_x+\lambda\mathbf{I}\right)^{-1}\frac{\vec{a}_{m+1}}{s_{m+1}}\right)^2\right)^2 \\ &\leq \left(\frac{1}{1+\psi_a}\right)^2\left(\frac{\vec{a}_{m+1}^T\left(\mathbf{A}_x^T\mathbf{A}_x+\lambda\mathbf{I}\right)^{-1}\vec{a}_{m+1}}{s_{m+1}^2}\right)^2=\left(\frac{\psi_a}{1+\psi_a}\right)^2\end{aligned}$$

Combining, we have that

$$\begin{aligned}\delta_{\vec{e}_{\overline{P}}(\vec{v},\vec{x})}(\vec{x}) &\leq \delta_{\vec{e}_P(\vec{\tau},\vec{x})}(\vec{x})+\left(c_e+\frac{\psi_a}{1+\psi_a}\right)\sqrt{\frac{\psi_a}{\mu(\vec{x})}}+\frac{\psi_a}{1+\psi_a} \\ &\leq \delta_{\vec{e}_P(\vec{\tau},\vec{x})}(\vec{x})+(c_e+\psi_a)\sqrt{\frac{\psi_a}{\mu(\vec{x})}}+\psi_a.\end{aligned}$$

□

We now bound the effect of removing a constraint.

Lemma 18 (Removing a Constraint)**.** *Let* $\mathbf{A}\in\mathbb{R}^{m\times n}$, $\vec{b}\in\mathbb{R}^m$, $\vec{\tau}\in\mathbb{R}^m$, *and* $\vec{x}\in P\stackrel{\text{def}}{=}\{\vec{y}\in\mathbb{R}^n:\mathbf{A}\vec{y}\geq\vec{b}\}$. *Let* $\overline{\mathbf{A}}\in\mathbb{R}^{(m-1)\times n}$ *be* $\mathbf{A}$ *with row* m *removed, let* $\overline{b}\in\mathbb{R}^{m-1}$ *denote the first* $m-1$ *coordinates of* $\vec{b}$, *and let* $\overline{P}\stackrel{\text{def}}{=}\{\vec{y}\in\mathbb{R}^n:\overline{\mathbf{A}}\vec{y}\geq\overline{b}\}$. *Let* $\psi_d=\psi_P(\vec{x})_m$.

Now, let $\vec{v}\in\mathbb{R}^{m-1}$ *be defined so that for all* $i\in[m-1]$

$$\upsilon_i=\tau_i+\frac{1}{1-\psi_d}\left(\mathbf{A}_x\left(\mathbf{A}_x^T\mathbf{A}_x+\lambda\mathbf{I}\right)^{-1}\mathbf{A}_x^T\vec{\mathbb{1}}_m\right)_i^2.$$

Assume $\psi_d\leq 1.1\mu(\vec{x})\leq\frac{1}{10}$ *and* $\left\|\vec{e}_P(\vec{\tau},\vec{x})\right\|_\infty\leq c_e\leq\frac{1}{2}$, *we have the following:*

- [Leverage Score Estimation] $e_{\overline{P}}(\vec{v},\vec{x})_i=e_P(\vec{\tau},\vec{x})_i$ for all $i\in[m-1]$.
- [Function Value Decrease] $p_{\vec{e}_{\overline{P}}(\vec{v},\vec{x})}(\vec{x})=p_{\vec{e}_P(\vec{\tau},\vec{x})}(\vec{x})+[c_e+e_P(\vec{\tau},\vec{x})_m]\ln s(\vec{x})_m+\ln(1-\psi_d)$
- [Centrality Increase] $\delta_{\vec{e}_{\overline{P}}(\vec{v},\vec{x})}(\vec{x})\leq\frac{1}{\sqrt{1-2\mu(\vec{x})}}\delta_{\vec{e}_P(\vec{\tau},\vec{x})}(\vec{x})+3(c_e+\mu(\vec{x}))$.

Proof. By (1) in Lemma 15, we have that for all $i\in[m-1]$

$$\psi_{\overline{P}}(\vec{x})_i=\psi_P(\vec{x})_i+\frac{1}{1-\psi_d}\left(\vec{\mathbb{1}}_i^T\mathbf{A}_x\left(\mathbf{A}_x^T\mathbf{A}_x+\lambda\mathbf{I}\right)^{-1}\mathbf{A}_x^T\vec{\mathbb{1}}_m\right)^2.$$

Consequently, [Leverage Score Estimation] holds. Furthermore, by (3) in Lemma 15, this then implies that [Function Value Change] holds.

To bound the change in centrality we first note that by (1) and (2) in Lemma 15, we have that the approximate Hessian for $\overline{P}$, denoted $\overline{\mathbf{H}}(\vec{x})$, is bounded by

$$\begin{aligned}\overline{\mathbf{H}}(\vec{x})^{-1}&=\left(\mathbf{H}(\vec{x})-\mathbf{A}_x^T\left(c_e\mathbf{I}+\mathbf{\Psi}_x\right)^{1/2}\vec{\mathbb{1}}_m\vec{\mathbb{1}}_m^T\left(c_e\mathbf{I}+\mathbf{\Psi}_x\right)^{1/2}\mathbf{A}_x\right)^{-1} \\ &\preceq\left(1+\frac{\alpha}{1-\alpha}\right)\mathbf{H}(\vec{x})^{-1}=\left(\frac{1}{1-\alpha}\right)\mathbf{H}(\vec{x})^{-1}\end{aligned}$$

where $\alpha \stackrel{\text{def}}{=} \vec{\mathbb{1}}_m^T \left(c_e \mathbf{I} + \mathbf{\Psi}_x\right)^{1/2} \mathbf{A}_x \mathbf{H}(\vec{x})^{-1} \mathbf{A}_x^T \left(c_e \mathbf{I} + \mathbf{\Psi}_x\right)^{1/2} \vec{\mathbb{1}}_m$. Using $c_e + \mu(\vec{x}) \leq \frac{1}{2} + \frac{1}{10} \leq 1$, we have

$$\mathbf{H}(\vec{x})^{-1} \preceq \left(\mathbf{A}_x^T (c_e + \mu(\vec{x})) \mathbf{A}_x + \lambda \mathbf{I}\right)^{-1} \preceq (c_e + \mu(\vec{x}))^{-1} \left(\mathbf{A}_x^T \mathbf{A}_x + \lambda \mathbf{I}\right)^{-1}. \tag{6.5}$$

Using this, we have

$$\alpha \leq \left(\frac{c_e + \psi_d}{c_e + \mu(\vec{x})}\right) \vec{\mathbb{1}}_m^T \mathbf{A}_x \left(\mathbf{A}_x^T \mathbf{A}_x + \lambda \mathbf{I}\right)^{-1} \mathbf{A}_x^T \vec{\mathbb{1}}_m = \left(\frac{c_e + \psi_d}{c_e + \mu(\vec{x})}\right) \psi_d \,. \tag{6.6}$$

Now let $\vec{\tau}' \in \mathbb{R}^{m-1}$ be defined so that $\tau'_i = \tau_i$ for all $i \in [m-1]$. We have by above that

$$\delta_{\vec{e}_{\overline{P}}(\vec{v},\vec{x})}(\vec{x}) = \left\|\overline{\mathbf{A}}_x^T (c_e \vec{\mathbb{1}} + \vec{v})\right\|_{\overline{\mathbf{H}}^{-1}} \leq \frac{1}{\sqrt{1-\alpha}} \left\|\overline{\mathbf{A}}_x^T (c_e \vec{\mathbb{1}} + \vec{v})\right\|_{\mathbf{H}^{-1}}$$

and therefore, by triangle inequality

$$\begin{aligned}
\left\|\overline{\mathbf{A}}_x^T (c_e \vec{\mathbb{1}} + \vec{v})\right\|_{\mathbf{H}^{-1}} &\leq \left\|\mathbf{A}_x^T (c_e \vec{\mathbb{1}} + \vec{\tau})\right\|_{\mathbf{H}^{-1}} + \left\|\mathbf{A}_x^T \vec{\mathbb{1}}_m (c_e + \tau_m)\right\|_{\mathbf{H}^{-1}} + \left\|\mathbf{A}_x^T (\vec{\tau}' - \vec{v})\right\|_{\mathbf{H}^{-1}} \\
&= \delta_{\vec{e}_P(\vec{\tau},\vec{x})}(\vec{x}) + (c_e + \tau_m) \left\|\mathbf{A}_x^T \vec{\mathbb{1}}_m\right\|_{\mathbf{H}^{-1}} + \left\|\mathbf{A}_x^T (\vec{\tau}' - \vec{v})\right\|_{\mathbf{H}^{-1}}.
\end{aligned}$$

Now, (6.5) shows that

$$\left\|\mathbf{A}_x^T \vec{\mathbb{1}}_m\right\|_{\mathbf{H}^{-1}} \leq \frac{1}{\sqrt{c_e + \mu(\vec{x})}} \left\|\mathbf{A}_x^T \vec{\mathbb{1}}_m\right\|_{(\mathbf{A}_x^T \mathbf{A}_x + \lambda \mathbf{I})^{-1}} \leq \sqrt{\frac{\psi_d}{c_e + \mu(\vec{x})}}$$

Furthermore, since $\mathbf{\Psi}^{-1} \succeq \mathbf{A}_x \left(\mathbf{A}_x^T \mathbf{\Psi} \mathbf{A}_x\right)^{-1} \mathbf{A}_x^T \succeq \mathbf{A}_x \mathbf{H}^{-1} \mathbf{A}_x^T$, by Lemma 16 we have

$$\begin{aligned}
\left\|\mathbf{A}_x^T \left(\vec{\tau}' - \vec{v}\right)\right\|_{\mathbf{H}^{-1}}^2 &\leq \left\|\vec{\tau}' - \vec{v}\right\|_{\mathbf{\Psi}^{-1}}^2 \\
&= \sum_{i \in [m]} \frac{1}{\psi(\vec{x})_i} \left(\frac{1}{1 - \psi_d} \left(\vec{\mathbb{1}}_i^T \mathbf{A}_x \left(\mathbf{A}_x^T \mathbf{A}_x + \lambda \mathbf{I}\right)^{-1} \mathbf{A}_x^T \vec{\mathbb{1}}_m\right)^2\right)^2 \\
&\leq \left(\frac{1}{1 - \psi_d}\right)^2 \left(\vec{\mathbb{1}}_m^T \mathbf{A}_x \left(\mathbf{A}_x^T \mathbf{A}_x + \lambda \mathbf{I}\right)^{-1} \mathbf{A}_x^T \vec{\mathbb{1}}_m\right)^2 = \left(\frac{\psi_d}{1 - \psi_d}\right)^2
\end{aligned}$$

Combining, we have that

$$\delta_{\vec{e}_{\overline{P}}(\vec{v},\vec{x})}(\vec{x}) \leq \frac{1}{\sqrt{1-\alpha}} \left[\delta_{\vec{e}_P(\vec{\tau},\vec{x})}(\vec{x}) + (c_e + \tau_m) \sqrt{\frac{\psi_d}{c_e + \mu(\vec{x})}} + \frac{\psi_d}{1 - \psi_d}\right].$$

Using the assumption $\psi_d \leq 1.1\mu(\vec{x}) \leq \frac{1}{10}$, $\left\|\vec{e}_P(\vec{\tau},\vec{x})\right\|_\infty \leq c_e$ and (6.6), we have $\alpha \leq 1.1\psi_d \leq 1.21\mu(\vec{x})$ and $\tau_m \leq \psi_d + c_e$, and

$$\begin{aligned}
\delta_{\vec{e}_{\overline{P}}(\vec{v},\vec{x})}(\vec{x}) &\leq \frac{1}{\sqrt{1 - 1.3\mu(\vec{x})}} \left[\delta_{\vec{e}_P(\vec{\tau},\vec{x})}(\vec{x}) + (c_e + \tau_m)\sqrt{1.1} + 1.2\psi_d\right] \\
&\leq \frac{1}{\sqrt{1 - 2\mu(\vec{x})}} \delta_{\vec{e}_P(\vec{\tau},\vec{x})}(\vec{x}) + \frac{1}{\sqrt{1 - \frac{1.3}{10}}} \left(\sqrt{1.1} \cdot 2c_e + (\sqrt{1.1} + 1.2 \cdot 1.1)\mu(\vec{x})\right)
\end{aligned}$$

□

6.3 Hybrid Center Properties

Here we prove properties of points near the hybrid center. First we bound the distance between points in the $\mathbf{H}(\vec{x})$ norm in terms of the ℓ_2 norm of the points.

Lemma 19. *For $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $\vec{b} \in \mathbb{R}^m$ suppose that $\vec{x} \in P = \{\vec{y} : \mathbf{A}\vec{y} \geq \vec{b}\}$ and $\vec{e} \in \mathbb{R}^m$ such that $\left\|\vec{e}\right\|_\infty \leq \frac{1}{2}c_e < \frac{1}{20}$ and $\delta_{\vec{e}} \leq 0.1\sqrt{c_e + \mu(\vec{x})}$. Then for all $\vec{y} \in P$ we have*

$$\left\|\vec{x} - \vec{y}\right\|_{\mathbf{H}(\vec{x})} \leq \frac{12mc_e + 6n + 2\lambda\left\|\vec{y}\right\|_2^2}{\sqrt{c_e + \mu(\vec{x})}} \tag{6.7}$$

and

$$\left\|\vec{x}\right\|_2^2 \leq 4\lambda^{-1}(mc_e + n) + 2\left\|\vec{y}\right\|_2^2 \quad .$$

Proof. For notational simplicity let $\vec{t} \stackrel{\text{def}}{=} c_e\vec{\mathbb{1}} + \vec{e} + \vec{\psi}_x$, $\mathbf{T} \stackrel{\text{def}}{=} \mathbf{diag}(\vec{t})$, and $\mathbf{Q} \stackrel{\text{def}}{=} \mathbf{A}_x^T(c_e\mathbf{I} + \mathbf{\Psi}_x)\mathbf{A}_x$. We have

$$\left\|\vec{x} - \vec{y}\right\|_{\mathbf{A}_x^T\mathbf{T}\mathbf{A}_x}^2 = \sum_{i\in[m]} t_i\frac{[\vec{s}_x - \vec{s}_y]_i^2}{[\vec{s}_x]_i^2} = \sum_{i\in[m]} t_i\left(1 - 2\frac{[\vec{s}_y]_i}{[\vec{s}_x]_i} + \frac{[\vec{s}_y]_i^2}{[\vec{s}_x]_i^2}\right) \tag{6.8}$$

and

$$\sum_{i\in[m]} t_i\frac{[\vec{s}_y]_i^2}{[\vec{s}_x]_i^2} \leq \left(\sum_{i\in[m]} t_i\frac{[\vec{s}_y]_i}{[\vec{s}_x]_i}\right)\max_{i\in[m]}\frac{[\vec{s}_y]_i}{[\vec{s}_x]_i} \leq \left(\sum_{i\in[m]} t_i\frac{[\vec{s}_y]_i}{[\vec{s}_x]_i}\right)\left(1 + \left\|\mathbf{S}_x^{-1}(\vec{s}_y - \vec{s}_x)\right\|_\infty\right) \tag{6.9}$$

and

$$\begin{aligned}\left\|\mathbf{S}_x^{-1}\left(\vec{s}_x - \vec{s}_y\right)\right\|_\infty &\leq \max_{i\in[m]}\left|\vec{\mathbb{1}}_i\mathbf{S}_x^{-1}\mathbf{A}\left(\vec{x} - \vec{y}\right)\right| \leq \left\|\vec{x} - \vec{y}\right\|_{\mathbf{H}(\vec{x})}\sqrt{\max_{i\in[m]}\left[\mathbf{S}_x^{-1}\mathbf{A}\mathbf{H}(\vec{x})^{-1}\mathbf{A}^T\mathbf{S}_x^{-1}\right]_{ii}} \\ &\leq \left(c_e + \mu(\vec{x})\right)^{-1/2}\left\|\vec{x} - \vec{y}\right\|_{\mathbf{H}(\vec{x})} \quad .\end{aligned} \tag{6.10}$$

Now, clearly $\sum_{i\in[m]} t_i[\vec{s}_y]_i/[\vec{s}_x]_i$ is positive and since $\left\|\vec{e}\right\|_\infty \leq \frac{1}{2}c_e$ we know that $\frac{1}{2}\mathbf{Q} \preceq \mathbf{A}_x^T\mathbf{T}\mathbf{A}_x$. Therefore, by combining, (6.8), (6.9), and (6.10) we have

$$\begin{aligned}\frac{1}{2}\left\|\vec{x} - \vec{y}\right\|_{\mathbf{Q}}^2 &\leq \left\|\vec{t}\right\|_1 - \sum_{i\in[m]} t_i\frac{[\vec{s}_y]_i}{[\vec{s}_x]_i} + \left(\sum_{i\in[m]} t_i\frac{[\vec{s}_y]_i}{[\vec{s}_x]_i}\right)\frac{\left\|\vec{x} - \vec{y}\right\|_{\mathbf{H}(\vec{x})}}{\sqrt{c_e + \mu(\vec{x})}} \\ &\leq \left\|\vec{t}\right\|_1 + \left(\sum_{i\in[m]} t_i\frac{[\vec{s}_y]_i}{[\vec{s}_x]_i}\right)\frac{\left\|\vec{x} - \vec{y}\right\|_{\mathbf{H}(\vec{x})}}{\sqrt{c_e + \mu(\vec{x})}}\end{aligned} \tag{6.11}$$

Now since $\nabla p_{\vec{e}}(\vec{x}) = -\mathbf{A}^T\mathbf{S}_x^{-1}\mathbf{T}\vec{\mathbb{1}} + \lambda\vec{x}$ we have

$$\left\langle\vec{x} - \vec{y}, \nabla p_{\vec{e}}(\vec{x})\right\rangle = -\sum_{i\in[m]} t_i\frac{[\vec{s}_x - \vec{s}_y]_i}{[\vec{s}_x]_i} + \lambda\vec{x}^T(\vec{x} - \vec{y})$$

and therefore by Cauchy Schwarz and $\vec{x}^T\vec{y} \leq \left\|\vec{x}\right\|_2^2 + \frac{1}{4}\left\|\vec{y}\right\|_2^2$,

$$\sum_{i\in[m]} t_i\frac{[\vec{s}_y]_i}{[\vec{s}_x]_i} = \left\|\vec{t}\right\|_1 - \lambda\left\|\vec{x}\right\|_2^2 + \lambda\vec{x}^T\vec{y} + \left\langle\vec{x} - \vec{y}, \nabla p_{\vec{e}}(\vec{x})\right\rangle \tag{6.12}$$

$$\leq \left\|t\right\|_1 + \frac{\lambda}{4}\left\|\vec{y}\right\|_2^2 + \left\|\vec{x} - \vec{y}\right\|_{\mathbf{H}(\vec{x})}\delta_{\vec{e}}(\vec{x}) \quad . \tag{6.13}$$

Now, using (6.11), (6.13) and the definition of $\mathbf{H}(\vec{x})$, we have

$$\begin{aligned}\frac{1}{2}\|\vec{x}-\vec{y}\|^2_{\mathbf{H}(\vec{x})} &= \frac{1}{2}\|\vec{x}-\vec{y}\|^2_{\mathbf{Q}} + \frac{\lambda}{2}\|\vec{x}-\vec{y}\|_2^2 \\ &\le \|\vec{t}\|_1 + \left(\sum_{i\in[m]} t_i \frac{[\vec{s}_y]_i}{[\vec{s}_x]_i}\right)\frac{\|\vec{x}-\vec{y}\|_{\mathbf{H}(\vec{x})}}{\sqrt{c_e+\mu(\vec{x})}} + \frac{\lambda}{2}\|\vec{x}-\vec{y}\|_2^2 \\ &\le \|\vec{t}\|_1 + \frac{\|t\|_1 + \frac{\lambda}{4}\|\vec{y}\|_2^2}{\sqrt{c_e+\mu(\vec{x})}}\|\vec{x}-\vec{y}\|_{\mathbf{H}(\vec{x})} + \delta_e(\vec{x})\frac{\|\vec{x}-\vec{y}\|^2_{\mathbf{H}(\vec{x})}}{\sqrt{c_e+\mu(\vec{x})}} + \frac{\lambda}{2}\|\vec{x}-\vec{y}\|_2^2.\end{aligned}$$

Using the fact that $\delta_e(\vec{x}) \le 0.1\sqrt{c_e+\mu(\vec{x})}$, we have

$$\frac{1}{4}\|\vec{x}-\vec{y}\|^2_{\mathbf{H}(\vec{x})} \le \|\vec{t}\|_1 + \frac{\lambda}{2}\|\vec{x}-\vec{y}\|_2^2 + \frac{\|t\|_1 + \frac{\lambda}{4}\|\vec{y}\|_2^2}{\sqrt{c_e+\mu(\vec{x})}}\|\vec{x}-\vec{y}\|_{\mathbf{H}(\vec{x})}. \tag{6.14}$$

Furthermore, since $\sum_{i\in[m]} t_i[\vec{s}_y]_i/[\vec{s}_x]_i$ is positive, (6.12) shows that

$$\lambda\vec{x}^T(\vec{x}-\vec{y}) = \lambda\|\vec{x}\|_2^2 - \lambda\vec{x}^T\vec{y} \le \|\vec{t}\|_1 + \langle \vec{x}-\vec{y}, \nabla p_{\vec{e}}(\vec{x})\rangle \le \|\vec{t}\|_1 + \|\vec{x}-\vec{y}\|_{\mathbf{H}(\vec{x})}\delta_e(\vec{x})$$

and hence

$$\begin{aligned}\frac{\lambda}{2}\|\vec{x}-\vec{y}\|_2^2 \le \frac{\lambda}{2}\|\vec{x}-\vec{y}\|_2^2 + \frac{\lambda}{2}\|\vec{x}\|_2^2 &= \lambda\vec{x}^T(\vec{x}-\vec{y}) + \frac{\lambda}{2}\|\vec{y}\|_2^2 \\ &\le \|\vec{t}\|_1 + \frac{\lambda}{2}\|\vec{y}\|_2^2 + \|\vec{x}-\vec{y}\|_{\mathbf{H}(\vec{x})}\delta_e(\vec{x}) \quad .\end{aligned} \tag{6.15}$$

Putting (6.15) into (6.14) and using the fact that $\delta_e(\vec{x}) \le 0.1\sqrt{c_e+\mu(\vec{x})}$, we have

$$\frac{1}{4}\|\vec{x}-\vec{y}\|^2_{\mathbf{H}(\vec{x})} \le 2\|\vec{t}\|_1 + \frac{\lambda}{2}\|\vec{y}\|_2^2 + \left(0.1 + \frac{\|\vec{t}\|_1 + \frac{\lambda}{4}\|\vec{y}\|_2^2}{\sqrt{c_e+\mu(\vec{x})}}\right)\|\vec{x}-\vec{y}\|_{\mathbf{H}(\vec{x})}.$$

Now, using $\|\vec{t}\|_1 \le 2mc_e + n$, we have

$$\frac{1}{4}\|\vec{x}-\vec{y}\|^2_{\mathbf{H}(\vec{x})} \le 2\alpha + (0.1+\alpha)\|\vec{x}-\vec{y}\|_{\mathbf{H}(\vec{x})} \quad \text{for} \quad \alpha = \frac{2mc_e + n + \frac{\lambda}{4}\|\vec{y}\|_2^2}{\sqrt{c_e+\mu(\vec{x})}} \quad .$$

Since $\sqrt{c_e+\mu(\vec{x})} \le 1.05$, we have $\alpha \ge 0.9$ and hence

$$\|\vec{x}-\vec{y}\|_{\mathbf{H}(\vec{x})} \le \frac{0.1+\alpha+\sqrt{(\alpha+0.1)^2+2\alpha}}{2\cdot\frac{1}{4}} \le 6\alpha$$

yielding (6.7).

We also have by (6.12) and the fact that $\delta_e(\vec{x}) \le 0.1\sqrt{c_e+\mu(\vec{x})}$,

$$\begin{aligned}\lambda\|\vec{x}\|_2^2 &= \|t\|_1 + \lambda\vec{x}^T\vec{y} + \langle \vec{x}-\vec{y}, \nabla p_{\vec{e}}(\vec{x})\rangle - \sum_{i\in[m]} t_i\frac{[\vec{s}_y]_i}{[\vec{s}_x]_i} \\ &\le \|t\|_1 + \frac{\lambda}{2}\|\vec{x}\|_2^2 + \frac{\lambda}{2}\|\vec{y}\|_2^2 + \|\vec{x}-\vec{y}\|_{\mathbf{H}(\vec{x})}\delta_e(\vec{x}) \\ &\le \|t\|_1 + \frac{\lambda}{2}\|\vec{x}\|_2^2 + \frac{\lambda}{2}\|\vec{y}\|_2^2 + 0.1\sqrt{c_e+\mu(\vec{x})}\|\vec{x}-\vec{y}\|_{\mathbf{H}(\vec{x})}\end{aligned}$$

Hence, using $\left\|\vec{t}\right\|_1 \leq 2mc_e + n$ and $\left\|\vec{x} - \vec{y}\right\|_{\mathbf{H}(\vec{x})} \leq 6\alpha$, we have

$$\begin{aligned}\frac{\lambda}{2}\|\vec{x}\|_2^2 &\leq \|t\|_1 + \frac{\lambda}{2}\|\vec{y}\|_2^2 + 0.6\left(2mc_e + n + \frac{\lambda}{4}\|\vec{y}\|_2^2\right) \\ &\leq \lambda\left\|\vec{y}\right\|_2^2 + 2(mc_e + n).\end{aligned}$$

□

In the following lemma we show how we can write one hyperplane in terms of the others provided that we are nearly centered and show there is a constraint that the central point is close to.

Lemma 20. *Let $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $\vec{b} \in \mathbb{R}^m$ such that $\left\|a_i\right\|_2 = 1$ for all i. Suppose that $\vec{x} \in P = \{\vec{y} : \mathbf{A}\vec{y} \geq \vec{b}\}$ and $\vec{e} \in \mathbb{R}^m$ such that $\left\|\vec{e}\right\|_\infty \leq \frac{1}{2}c_e \leq \frac{1}{2}$. Furthermore, let $\epsilon = \min_{j\in[m]} s_j(\vec{x})$ and suppose that $i = \arg\min_{j\in[m]} s_j(\vec{x})$ then*

$$\left\|\vec{a}_i + \sum_{j\neq i}\left(\frac{s(x)_i}{s(x)_j}\right)\left(\frac{c_e + e_j + \psi_j(\vec{x})}{c_e + e_i + \psi_i(\vec{x})}\right)\vec{a}_j\right\|_2 \leq \frac{2\epsilon}{(c_e + \mu(\vec{x}))}\left[\lambda\left\|\vec{x}\right\|_2 + \delta_e(\vec{x})\sqrt{\frac{mc_e + n}{\epsilon^2} + \lambda}\right].$$

Proof. We know that

$$\begin{aligned}\nabla p_e(\vec{x}) &= -\mathbf{A}^T\mathbf{S}_x^{-1}(c_e\vec{\mathbb{1}} + \vec{e} + \vec{\psi}_x) + \lambda\vec{x} \\ &= \lambda\vec{x} - \sum_{i\in[m]}\frac{(c_e + e_i + \psi_i)}{s(\vec{x})_i}\vec{a}_i\end{aligned}$$

Consequently, by $\left\|\vec{e}\right\|_\infty \leq \frac{1}{2}c_e$, and $\psi_i(\vec{x}) \geq \mu(\vec{x})$

$$\begin{aligned}\left\|\vec{a}_i + \sum_{j\neq i}\left(\frac{s(x)_i}{s(x)_j}\right)\left(\frac{c_e + e_j + \psi_j(\vec{x})}{c_e + e_i + \psi_i(\vec{x})}\right)\vec{a}_j\right\|_2 &= \frac{s_i(\vec{x})}{(c_e + e_i + \psi_i(\vec{x}))}\left\|\mathbf{A}^T\mathbf{S}_x^{-1}(c_e\vec{\mathbb{1}} + \vec{e} + \vec{\psi}_x)\right\|_2 \\ &\leq \frac{2\epsilon}{(c_e + \mu(\vec{x}))}\left[\lambda\left\|\vec{x}\right\|_2 + \left\|\nabla p_e(\vec{x})\right\|_2\right].\end{aligned}$$

Using $\left\|\vec{a}_i\right\| = 1$, $\sum_i \psi_i \leq n$, and $s_i(\vec{x}) \geq \epsilon$, we have

$$\begin{aligned}\mathrm{Tr}(\mathbf{A}_x^T(c_e\mathbf{I} + \mathbf{\Psi}_x)\mathbf{A}_x) &= \mathrm{Tr}(\mathbf{A}_x\mathbf{A}_x^T(c_e\mathbf{I} + \mathbf{\Psi}_x)) \\ &= \sum_i (c_e + \psi_i)\frac{\left\|a_i\right\|_2^2}{s_i^2(\vec{x})} \leq \frac{mc_e + n}{\epsilon^2}. \qquad (6.16)\end{aligned}$$

Hence, we have $\mathbf{H}(\vec{x}) \preceq \left(\frac{mc_e+n}{\epsilon^2} + \lambda\right)\mathbf{I}$ and $\left\|\nabla p_e(\vec{x})\right\|_2 \leq \delta_e(\vec{x})\sqrt{\frac{mc_e+n}{\epsilon^2} + \lambda}$ yielding the result. □

6.4 The Algorithm

Here, we put all the results in the previous sections to get our ellipsoid algorithm. Below is a sketch of the pseudocode; we use c_a, c_d, c_e, c_Δ to denote parameters we decide later.

In the algorithm, there are two main invariants we maintain. First, we maintain that the centrality $\delta_{P,\vec{e}}(\vec{x})$, which indicates how close $\vec{x}$ is to the minimum point of $p_{\vec{e}}$, is small. Second, we maintain that $\left\|\vec{e}(\vec{\tau},\vec{x})\right\|_\infty$, which indicates how accurate the leverage score estimate $\vec{\tau}$ is, is small. In the following lemma we show that we maintain both invariants throughout the algorithm.

Algorithm 2: Our Cutting Plane Method

Input: $\mathbf{A}^{(0)} \in \mathbb{R}^{m \times n}$, $\vec{b}^{(0)} \in \mathbb{R}^m$, $\epsilon > 0$, and radius $R > 0$.
Input: A separation oracle for a non-empty set $K \subset B_\infty(R)$.
Check: Throughout the algorithm, if $s_i(\vec{x}^{(k)}) < \epsilon$ output $P^{(k)}$.
Check: Throughout the algorithm, if $\vec{x}^{(k)} \in K$, output $\vec{x}^{(k)}$.
Set $P^{(0)} = B_\infty(R)$.
Set $\vec{x}^{(0)} := \vec{0}$ and compute $\tau_i^{(0)} = \psi_{P^{(0)}}(\vec{x}^{(0)})_i$ for all $i \in [m]$ exactly.
for $k = 0$ *to* ∞ **do**
  Let $m^{(k)}$ be the number of constraints in $P^{(k)}$.
  Compute $\vec{w}^{(k)}$ such that $\mathbf{\Psi}_{P^{(k)}}(\vec{x}^{(k)}) \preceq \mathbf{W}^{(k)} \preceq (1 + c_\Delta)\mathbf{\Psi}_{P^{(k)}}(\vec{x}^{(k)})$.
  Let $i^{(k)} \in \arg\max_{i \in [m^{(k)}]} \left| w_i^{(k)} - \tau_i^{(k)} \right|$.
  Set $\tau_{i^{(k)}}^{(k+\frac{1}{3})} = \psi_{P^{(k)}}(\vec{x}^{(k)})_{i^{(k)}}$ and $\tau_j^{(k+\frac{1}{3})} = \tau_j^{(k)}$ for all $j \neq i^{(k)}$.
  if $\min_{i \in [m^{(k)}]} w_i^{(k)} \leq c_d$ **then**
    Remove constraint with minimum $w_i^{(k)}$ yielding polytope $P^{(k+1)}$.
    Update $\vec{\tau}$ according to Lemma 18 to get $\tau_j^{(k+\frac{2}{3})}$.
  else
    Use separation oracle at $\vec{x}^{(k)}$ to get a constraint $\{\vec{x} : \vec{a}^T\vec{x} \geq \vec{a}^T\vec{x}^{(k)}\}$ with $\left\|\vec{a}\right\|_2 = 1$.
    Add constraint $\{\vec{x} : \vec{a}^T\vec{x} \geq \vec{a}^T\vec{x}^{(k)} - c_a^{-1/2}\sqrt{\vec{a}^T(\mathbf{A}^T\mathbf{S}_{\vec{x}^{(k)}}^{-2}\mathbf{A} + \lambda\mathbf{I})^{-1}\vec{a}}\}$ yielding polytope $P^{(k+1)}$.
    Update $\vec{\tau}$ according to Lemma 17 to get $\tau_j^{(k+\frac{2}{3})}$.
  $(\vec{x}^{(k+1)}, \vec{\tau}^{(k+1)}) = \texttt{Centering}(\vec{x}^{(k)}, \vec{\tau}^{(k+\frac{2}{3})}, 200, c_\Delta)$.
end

Lemma 21. *Assume that $c_e \leq c_d \leq \frac{1}{10^6}$, $c_a\sqrt{c_a} \leq \frac{c_d}{10^3}$, $c_d \leq c_a$, and $c_\Delta \leq Cc_e/\log(n)$ for some small enough universal constant C. During our cutting plane method, for all k, with high probability in n, we have*

1. $\left\|\vec{e}(\vec{\tau}^{(k+\frac{1}{3})}, \vec{x}^{(k)})\right\|_\infty \leq \frac{1}{1000}c_e$, $\left\|\vec{e}(\vec{\tau}^{(k+\frac{2}{3})}, \vec{x}^{(k)})\right\|_\infty \leq \frac{1}{1000}c_e$, $\left\|\vec{e}(\vec{\tau}^{(k+1)}, \vec{x}^{(k+1)})\right\|_\infty \leq \frac{1}{400}c_e$.
2. $\delta_{P^{(k)}, \vec{e}(\vec{\tau}^{(k+\frac{2}{3})}, \vec{x}^{(k)})}(\vec{x}^{(k)}) \leq \frac{1}{100}\sqrt{c_e + \min\left(\mu(\vec{x}^{(k)}), c_d\right)}$.
3. $\delta_{P^{(k+1)}, \vec{e}(\vec{\tau}^{(k+1)}, \vec{x}^{(k+1)})}(\vec{x}^{(k+1)}) \leq \frac{1}{400}\sqrt{c_e + \min\left(\mu(\vec{x}^{(k+1)}), c_d\right)}$.

Proof. Some statements of the proof hold only with high probability in n; we omit mentioning this for simplicity.

We prove by induction on k. Note that the claims are written in order consistent with the algorithm and proving the statement for k involves bounding centrality at the point $\vec{x}^{(k+1)}$. Trivially we define, $\vec{\tau}^{(-1)} = \vec{\tau}^{(-\frac{2}{3})} = \vec{\tau}^{(-\frac{1}{3})} = \vec{\tau}^{(0)}$ and note that the claims then hold for $k = -1$ as we compute the initial leverage scores, $\vec{\tau}^{(0)}$, exactly and since the polytope is symmetric we have $\delta_{\vec{e}(\vec{\tau}^{(0)}, \vec{x}^{(0)})}(\vec{0}) = 0$. We now suppose they hold for all $r < t$ and show that they hold for $r = t$.

We first bound δ. For notational simplicity, let $\eta_t \stackrel{\text{def}}{=} \sqrt{c_e + \min\{\mu(\vec{x}^{(t)}), c_d\}}$. By the induction hypothesis we know that $\delta_{P^{(t)}, \vec{e}(\vec{\tau}^{(t)}, \vec{x}^{(t)})}(\vec{x}^{(t)}) \leq \frac{1}{400}\eta_t$. Now, when we update $\tau^{(t)}$ to $\tau^{(t+\frac{1}{3})}$, we set

$\vec{e}_{i^{(t)}}$ to 0. Consequently, Lemma 13 and the induction hypothesis $\left\|\vec{e}(\vec{\tau}^{(t)}, \vec{x}^{(t)})\right\|_\infty \le \frac{1}{400} c_e$ show that

$$\begin{aligned}\delta_{P^{(t)}, \vec{e}(\vec{\tau}^{(t+\frac{1}{3})}, \vec{x}^{(t)})}(\vec{x}^{(t)}) &\le \delta_{P^{(t)}, \vec{e}(\vec{\tau}^{(t)}, \vec{x}^{(t)})}(\vec{x}^{(t)}) + \frac{1}{\sqrt{c_e + \mu(\vec{x}^{(t)})}} e_{i^{(t)}}(\vec{\tau}^{(t)}, \vec{x}^{(t)}) \\ &\le \frac{1}{400}\eta_t + \frac{\sqrt{c_e}}{400} \le \frac{\eta_t}{200} \end{aligned} \tag{6.17}$$

Next, we estimate the δ changes when we remove or add a constraint.

For the case of removal, we note that it happens only if $\mu(\vec{x}^{(t)}) \le \min_i w_i \le c_d \le \frac{1}{10^6}$. Also, the row we remove has leverage score at most $1.1\mu(\vec{x}^{(t)})$ because we pick the row with minimum w. Hence, Lemma 18 and $c_e \le \frac{1}{10^6}$ show that

$$\begin{aligned}\delta_{P^{(t+1)}, \vec{e}(\vec{\tau}^{(t+\frac{2}{3})}, \vec{x}^{(t)})}(\vec{x}^{(t)}) &\le \frac{1}{\sqrt{1 - 2\mu(\vec{x}^{(t)})}} \delta_{P^{(t)}, \vec{e}(\vec{\tau}^{(t+\frac{1}{3})}, \vec{x}^{(t)})}(\vec{x}^{(t)}) + 2.7(c_e + \mu(\vec{x}^{(t)})) \\ &\le \frac{1}{\sqrt{1 - 2 \cdot 10^{-6}}} \left(\frac{\eta_t}{200}\right) + 3(c_e + \mu(\vec{x}^{(t)})) \le \frac{\eta_t}{100}\end{aligned}$$

where we used the fact $\mu(\vec{x}^{(t)}) \le c_d$ and hence $c_e + \mu(\vec{x}^{(t)}) \le \sqrt{c_e + c_d}\eta_t \le \frac{\sqrt{2}}{1000}\eta_t$.

For the case of addition, we note that it happens only if $2\mu(\vec{x}^{(t)}) \ge \min_i w_i \ge c_d$. Furthermore, in this case the hyperplane we add is chosen precisely so that $\psi_a = c_a$. Furthermore, since $c_e \le c_d \le c_a$ by Lemma 17 we have that

$$\delta_{P^{(t+1)}, \vec{e}(\vec{\tau}^{(t+\frac{2}{3})}, \vec{x}^{(t)})}(\vec{x}^{(t)}) \le \delta_{P^{(t)}, \vec{e}(\vec{\tau}^{(t+\frac{1}{3})}, \vec{x}^{(t)})} + (c_e + \psi_a)\sqrt{\frac{\psi_a}{\mu(\vec{x}^{(t)})}} + \psi_a \le \frac{\eta_t}{200} + 4c_a\sqrt{\frac{c_a}{c_d}} \quad .$$

Furthermore, since $c_a\sqrt{c_a} \le \frac{c_d}{1000}$, $\mu(\vec{x}^{(t)}) \ge c_d/2$, and $c_d \le 10^{-6}$ we know that $4c_a\sqrt{c_a/c_d} \le \frac{1}{200}\eta_t$ and consequently in both cases we have $\delta_{P^{(t+1)}, \vec{e}(\vec{\tau}^{(t+\frac{2}{3})}, \vec{x}^{(t)})}(\vec{x}^{(t)}) \le \frac{1}{100}\eta_t$.

Now, note that Lemmas 17 and 18 show that $\vec{e}$ does not change during the addition or removal of an constraint. Hence, we have $\left\|\vec{e}(\vec{\tau}^{(t+\frac{2}{3})}, \vec{x}^{(t)})\right\|_\infty \le \left\|\vec{e}(\vec{\tau}^{(t+\frac{1}{3})}, \vec{x}^{(t)})\right\|_\infty$. Furthermore, we know the step "$\vec{\tau}^{(k+\frac{1}{3})}_{i^{(k)}} = \psi_{P^{(k)}}(\vec{x}^{(k)})_{i^{(k)}}$" only decreases $\left\|\vec{e}\right\|_\infty$ and hence we have $\left\|\vec{e}(\vec{\tau}^{(t+\frac{2}{3})}, \vec{x}^{(t)})\right\|_\infty \le \left\|\vec{e}(\vec{\tau}^{(t)}, \vec{x}^{(t)})\right\|_\infty \le \frac{c_e}{400}$. Thus, we have all the conditions needed for Lemma 14 and consequently

$$\delta_{P^{(t+1)}, \vec{e}(\vec{\tau}^{(t+1)}, \vec{x}^{(t+1)})}(\vec{x}^{(t+1)}) \le 2\left(1 - \frac{1}{64}\right)^{200} \delta_{P^{(t+1)}, \vec{e}(\vec{\tau}^{(t+\frac{2}{3})}, \vec{x}^{(t)})}(\vec{x}^{(t)}) \le \frac{1}{1000}\eta_t \quad .$$

Lemma 14 also shows that that $\left\|\frac{s(\vec{x}^{(t+1)}) - s(\vec{x}^{(t)})}{s(\vec{x}^{(t)})}\right\|_2 \le \frac{1}{10}$ and hence $\psi_i(\vec{x}^{(t)}) \le 2\psi_i(\vec{x}^{(t+1)})$ for all i. Therefore, $\eta_t \le 2\eta_{t+1}$ and thus

$$\delta_{P^{(t+1)}, \vec{e}(\vec{\tau}^{(t+1)}, \vec{x}^{(t+1)})}(\vec{x}^{(t+1)}) \le \frac{\sqrt{c_e + \min\left(c_d, \mu(\vec{x}^{(t+1)})\right)}}{400}.$$

completing the induction case for δ.

Now, we bound $\left\|\vec{e}\right\|_\infty$. Lemma 17 and 18 show that $\vec{e}$ does not change during the addition or removal of an constraint. Hence, $\vec{e}$ is affected by only the update step "$\vec{\tau}^{(k+\frac{1}{2})}_{i^{(k)}} = \psi_{P^{(k)}}(\vec{x}^{(k)})_{i^{(k)}}$" and the centering step. Using the induction hypothesis $\delta_{P^{(r)}, \vec{e}(\vec{\tau}^{(r+\frac{2}{3})}, \vec{x}^{(r)})}(\vec{x}^{(r)}) \le \frac{1}{100}\eta_r$ and Lemma 14 shows that $\mathbb{E}\vec{e}(\vec{\tau}^{(r+1)}, \vec{x}^{(r+1)}) = \vec{e}(\vec{\tau}^{(r+\frac{2}{3})}, \vec{x}^{(r)})$ and $\left\|\vec{e}(\vec{\tau}^{(r+1)}, \vec{x}^{(r+1)}) - \vec{e}(\vec{\tau}^{(r+\frac{2}{3})}, \vec{x}^{(r)})\right\|_2 \le \frac{1}{10} c_\Delta$ for

all $r \leq t$. The goal for the update step is to decrease $\vec{e}$ by updating $\vec{\tau}$. In Section 7.2, we give a self-contained analysis of the effect of this step as a game. In each round, the vector $\vec{e}$ is corrupted by some mean 0 and bounded variance noise and the problem is how to update $\vec{e}$ such that $\left\|\vec{e}\right\|_\infty$ is bounded. Theorem 34 shows that we can do this by setting the $\vec{e}_i = 0$ for the almost maximum coordinate in each iteration. This is exactly what the update step is doing. Hence, Theorem 34 shows that this strategy guarantees that after the update step, we have

$$\left\|\vec{e}(\tau^{(r+\frac{1}{3})}, \vec{x}^{(r)})\right\|_\infty = O\left(c_\Delta \log(n)\right)$$

for all $r \leq t$. Now, by our choice of c_Δ, we have $\left\|\vec{e}(\vec{\tau}^{(t+\frac{1}{3})}, \vec{x}^{(t)})\right\|_\infty \leq \frac{1}{1000} c_e$. Lemma 17 and 18 show that $\vec{e}$ does not change during the addition or removal of an constraint. Hence, we have $\left\|\vec{e}(\vec{\tau}^{(t+\frac{2}{3})}, \vec{x}^{(t)})\right\|_\infty \leq \frac{1}{1000} c_e$. Now, we note that again Lemma 14 shows $\left\|\vec{e}(\vec{\tau}^{(t+1)}, \vec{x}^{(t+1)}) - \vec{e}(\vec{\tau}^{(t+\frac{2}{3})}, \vec{x}^{(t)})\right\|_2 \leq \frac{1}{10} c_\Delta \leq \frac{1}{1000} c_e$, and we have $\left\|\vec{e}(\vec{\tau}^{(t+1)}, \vec{x}^{(t+1)})\right\|_\infty \leq \frac{c_e}{400}$. This finishes the induction case for $\left\|\vec{e}\right\|_\infty$ and proves this lemma. □

Next, we show the number of constraints is always linear to n.

Lemma 22. *Throughout our cutting plane method, there are at most* $1 + \frac{2n}{c_d}$ *constraints.*

Proof. We only add a constraint if $\min_i w_i \geq c_d$. Since $2\psi_i \geq w_i$, we have $\psi_i \geq \frac{c_d}{2}$ for all i. Letting m denote the number of constraints after we add that row, we have $n \geq \sum_i \psi_i \geq (m-1)(c_d/2)$. □

Using $K \neq \emptyset$ and $K \subset B_\infty(R)$, here we show that the points are bounded.

Lemma 23. *During our Cutting Plane Method, for all* k*, we have* $\left\|\vec{x}^{(k)}\right\|_2 \leq 6\sqrt{n/\lambda} + 2\sqrt{n}R$.

Proof. By Lemma 21 and Lemma 19 we know that $\left\|\vec{x}^{(k)}\right\|_2^2 \leq 4\lambda^{-1}(mc_e + n) + 2\left\|\vec{y}\right\|_2^2$ for any $\vec{y} \in P^{(k)}$. Since our method never cuts out any point in K and since K is nonempty, there is some $\vec{y} \in K \subset P^{(k)}$. Since $K \subset B_\infty(R)$, we have $\left\|\vec{y}\right\|_2^2 \leq nR$. Furthermore, by Lemma 22 we have that $mc_e \leq c_e + 2n \leq 3n$ yielding the result. □

Lemma 24. $s_i\left(\vec{x}^{(k)}\right) \leq 12\sqrt{n/\lambda} + 4\sqrt{n}R + \sqrt{\frac{1}{c_a \lambda}}$ *for all* i *and* k *in the our cutting plane method.*

Proof. Let $\vec{x}^{(j)}$ be the current point at the time that the constraint corresponding to s_i, denoted $\{\vec{x} : \vec{a}_i^T \vec{x} \geq a_i^T \vec{x}^{(j)} - s_i(\vec{x}^{(j)})\}$, was added. Clearly

$$s_i(\vec{x}^{(k)}) = \vec{a}_i^T \vec{x}^{(k)} - a_i^T \vec{x}^{(j)} + s_i(\vec{x}^{(j)}) \leq \left\|\vec{a}_i\right\| \cdot \left\|\vec{x}^{(k)}\right\| + \left|\vec{a}_i^T \vec{x}^{(j)} - s_i(\vec{x}^{(j)})\right| \quad .$$

On the one hand, if the constraint for s_i comes from the initial symmetric polytope $P^{(0)} = B_\infty(R)$, we know $\left|\vec{a}^T \vec{x}^{(j)} - \vec{s}_i(\vec{x}^{(j)})\right| \leq R$. On the other hand, if the constraint was added later then we know that

$$s_i(\vec{x}^{(j)}) = c_a^{-1/2} \sqrt{\vec{a}^T (\mathbf{A}^T \mathbf{S}_{\vec{x}^{(j)}}^{-2} \mathbf{A} + \lambda \mathbf{I})^{-1} \vec{a}} \leq (c_a \lambda)^{-1/2}$$

and $\left|\vec{a}^T \vec{x}^{(j)} - s_i(\vec{x}^{(j)})\right| \leq \left\|\vec{a}_i\right\| \cdot \left\|\vec{x}^{(j)}\right\| + \left|s_i(\vec{x}^{(j)})\right|$. Since $\left\|\vec{a}_i\right\|_2 = 1$ by design and $\left\|\vec{x}^{(j)}\right\|_2$ and $\left\|\vec{x}^{(k)}\right\|_2$ are upper bounded by $6\sqrt{n/\lambda} + 2\sqrt{n}R$ by Lemma 23, in either case the result follows. □

Now, we have everything we need to prove that the potential function is increasing in expectation.

Lemma 25. *Under the assumptions of Lemma 21 if* $\lambda = \frac{1}{c_a R^2}$, $c_e = \frac{c_d}{6\ln(17nR/\epsilon)}$, *and* $24c_d \le c_a \le \frac{1}{3}$ *then for all* k *we have*

$$\mathbb{E}p_{\vec{e}(\vec{\tau}^{(k+1)},\vec{x}^{(k+1)})}(\vec{x}^{(k+1)}) \;\ge\; p_{\vec{e}(\vec{\tau}^{(k)},\vec{x}^{(k)})}(\vec{x}^{(k)}) - c_d + \ln(1+\beta)$$

where $\beta = c_a$ *for the case of adding a constraint* $\beta = -c_d$ *for the case of removal.*

Proof. Note that there are three places which affect the function value, namely the update step for $\tau^{(k+\frac{1}{3})}$, the addition/removal of constraints, and the centering step. We bound the effect of each separately.

First, for the update step, we have

$$p_{\vec{e}(\vec{\tau}^{(k+\frac{1}{3})},\vec{x}^{(k)})}(\vec{x}^{(k)}) = -e_{i^{(k)}}\log(s_{i^{(k)}}(\vec{x}^{(k)})) + p_{\vec{e}(\vec{\tau}^{(k)},\vec{x}^{(k)})}(\vec{x}^{(k)}).$$

Lemma 24, the termination condition and $\lambda = \frac{1}{c_a R^2}$ ensure that

$$\epsilon \le s_{i^{(k)}}(\vec{x}^{(k)}) \le 12\sqrt{n/\lambda} + 4\sqrt{n}R + \sqrt{\frac{1}{c_a\lambda}} \le 17\sqrt{n}R \tag{6.18}$$

and Lemma 21 shows that $|e_{i^{(k)}}| \le c_e$. Hence, we have

$$p_{\vec{e}(\vec{\tau}^{(k+\frac{1}{3})},\vec{x}^{(k)})}(\vec{x}^{(k)}) \ge p_{\vec{e}(\vec{\tau}^{(k)},\vec{x}^{(k)})}(\vec{x}^{(k)}) - c_e\log(17nR/\epsilon).$$

For the addition step, Lemma 17 shows that

$$\begin{aligned} p_{\vec{e}(\vec{\tau}^{(k+\frac{2}{3})},\vec{x}^{(k)})}(\vec{x}^{(k)}) &= p_{\vec{e}(\vec{\tau}^{(k+\frac{1}{3})},\vec{x}^{(k)})}(\vec{x}^{(k)}) - c_e\ln s(\vec{x})_{m+1} + \ln(1+c_a) \\ &\ge p_{\vec{e}(\vec{\tau}^{(k+\frac{1}{3})},\vec{x}^{(k)})}(\vec{x}^{(k)}) - c_e\log(17nR/\epsilon) + \ln(1+c_a) \end{aligned}$$

and for the removal step, Lemma 18 and $|e_i| \le c_e$ shows that

$$\begin{aligned} p_{\vec{e}(\vec{\tau}^{(k+\frac{2}{3})},\vec{x}^{(k)})}(\vec{x}^{(k)}) &\ge p_{\vec{e}(\vec{\tau}^{(k+\frac{1}{3})},\vec{x}^{(k)})}(\vec{x}^{(k)}) - [c_e + e_P(\vec{\tau},\vec{x})_m]\ln s(\vec{x})_m + \ln(1-c_d) \\ &\ge p_{\vec{e}(\vec{\tau}^{(k+\frac{1}{3})},\vec{x}^{(k)})}(\vec{x}^{(k)}) - 2c_e\log(17nR/\epsilon) + \ln(1-c_d) \end{aligned}$$

After the addition or removal of a constraint, Lemma 21 shows that

$$\delta_{P^{(k)},\vec{e}(\vec{\tau}^{(k+\frac{2}{3})},\vec{x}^{(k)})}(\vec{x}^{(k)}) \le \frac{1}{100}\sqrt{c_e + \min\left(\mu(\vec{x}^{(k)}), c_d\right)}$$

and therefore Lemma 14 and $c_e \le c_d$ show that

$$\begin{aligned} \mathbb{E}p_{\vec{e}(\vec{\tau}^{(k+1)},\vec{x}^{(k+1)})}(\vec{x}^{(k+1)}) &\ge p_{\vec{e}(\vec{\tau}^{(k+\frac{2}{3})},\vec{x}^{(k)})}(\vec{x}^{(k)}) - 8\left(\frac{\sqrt{c_e + \min\left(\mu(\vec{x}^{(k)}), c_d\right)}}{100}\right)^2 \\ &\ge p_{\vec{e}(\vec{\tau}^{(k+\frac{2}{3})},\vec{x}^{(k)})}(\vec{x}^{(k)}) - \frac{c_d}{625}. \end{aligned}$$

Combining them with $c_e = \frac{c_d}{6\ln(17nR/\epsilon)}$, we have

$$\begin{aligned} \mathbb{E}p_{\vec{e}(\vec{\tau}^{(k+1)},\vec{x}^{(k+1)})}(\vec{x}^{(k+1)}) &\ge p_{\vec{e}(\vec{\tau}^{(k)},\vec{x}^{(k)})}(\vec{x}^{(k)}) - 3c_e\log(17nR/\epsilon) - \frac{c_d}{625} + \ln(1+\beta) \\ &\ge p_{\vec{e}(\vec{\tau}^{(k)},\vec{x}^{(k)})}(\vec{x}^{(k)}) - c_d + \ln(1+\beta) \end{aligned}$$

where $\beta = c_a$ for the case of addition and $\beta = -c_d$ for the case of removal. □

Theorem 26. *For* $c_a = \frac{1}{10^{10}}$, $c_d = \frac{1}{10^{12}}$, $c_e = \frac{c_d}{6\ln(17nR/\epsilon)}$, $c_\Delta = \frac{Cc_e}{\log(n)}$ *and* $\lambda = \frac{1}{c_a R^2}$ *for some small enough universal constant* C*, then we have*

$$\mathbb{E}p_{\vec{e}(\vec{\tau}^{(k+1)},\vec{x}^{(k+1)})}(\vec{x}^{(k+1)}) \quad \geq \quad p_{\vec{e}(\vec{\tau}^{(k)},\vec{x}^{(k)})}(\vec{x}^{(k)}) - \frac{1}{10^{11}} + \frac{9\beta}{10^{11}}$$

where $\beta = 1$ *for the case of addition and* $\beta = 0$ *for the case of removal.*

Proof. It is easy to see that these parameters satisfy the requirements of Lemma 25. □

6.5 Guarantees of the Algorithm

In this section we put everything together to prove Theorem 31, the main result of this section, providing the guarantees of our cutting plane method.

For the remainder of this section we assume that $c_a = \frac{1}{10^{10}}$, $c_d = \frac{1}{10^{12}}$, $c_e = \frac{c_d}{6\ln(17nR/\epsilon)}$, $c_\Delta = \frac{Cc_e}{\log(n)}$ and $\lambda = \frac{1}{c_a R^2}$. Consequently, throughout the algorithm we have

$$\left\|\vec{x}\right\|_2 \leq 6\sqrt{n/\lambda} + 2\sqrt{n}R = 6\sqrt{c_a n R^2} + 2\sqrt{n}R \leq 3\sqrt{n}R. \tag{6.19}$$

Lemma 27. *If* $s_i(\vec{x}^{(k)}) < \epsilon$ *for some* i *and* k *during our Cutting Plane Method then*

$$\max_{\vec{y}\in P^{(k)}\cap B_\infty(R)} \langle \vec{a}_i, \vec{y}\rangle - \min_{\vec{y}\in P^{(k)}\cap B_\infty(R)} \langle \vec{a}_i, \vec{y}\rangle \leq \frac{8n\epsilon}{c_a c_e}.$$

Proof. Let $\vec{y} \in P^{(k)} \cap B_\infty(R)$ be arbitrary. Since $\vec{y} \in B_\infty(R)$ clearly $\left\|\vec{y}\right\|_2^2 \leq nR^2$. Furthermore, by Lemma 22 and the choice of parameters $mc_e + n \leq 3n$. Consequently, by Lemma 19 and the fact that $\lambda = \frac{1}{c_a R^2}$ and $c_a < 1$ we have

$$\left\|\vec{x} - \vec{y}\right\|_{\mathbf{H}(\vec{x})} \leq \frac{12mc_e + 6n + 2\lambda\left\|\vec{y}\right\|_2^2}{\sqrt{c_e + \mu(\vec{x})}} \leq \frac{30n + 2\frac{n}{c_a}}{\sqrt{c_e + \mu(\vec{x})}} \leq \frac{4n}{c_a\sqrt{c_e}}$$

and therefore

$$\left\|\mathbf{S}_{x^{(k)}}^{-1}(s(\vec{x}^{(k)}) - s(\vec{y}))\right\|_\infty \leq \frac{1}{\sqrt{c_e}}\left\|\mathbf{S}_{x^{(k)}}^{-1}(s(\vec{x}^{(k)}) - s(\vec{y}))\right\|_{c_e\mathbf{I}+\mathbf{\Psi}} \leq \frac{4n}{c_a c_e} \quad .$$

Consequently, we have $(1 - \frac{4n}{c_a c_e})s_i(\vec{x}^{(k)}) \leq s_i(\vec{y}) \leq (1 + \frac{4n}{c_a c_e})s_i(\vec{x}^{(k)})$ for all $\vec{y} \in P^{(k)} \cap B_\infty(R)$. □

Now let us show how to compute a proof (or certificate) that the feasible region has small width on the direction $\vec{a}_i$.

Lemma 28. *Suppose that during some iteration* k *for* $i = \arg\min_j s_j(\vec{x}^{(k)})$ *we have* $s_i(\vec{x}^{(k)}) \leq \epsilon$*. Let* $(\vec{x}_*, \vec{\tau}_*) = \texttt{Centering}(\vec{x}^{(k)}, \vec{\tau}^{(k)}, 64\log(2R/\epsilon), c_\Delta)$ *where* $\vec{\tau}^{(k)}$ *is the* τ *at that point in the algorithm and let*

$$\vec{a}^* = \sum_{j\neq i} t_j \vec{a}_j \text{ where } t_j = \left(\frac{s(\vec{x}_*)_i}{s(\vec{x}_*)_j}\right)\left(\frac{c_e + e_j(\vec{x}_*, \vec{\tau}_*) + \psi_j(\vec{x}_*)}{c_e + e_i(\vec{x}_*, \vec{\tau}_*) + \psi_i(\vec{x}_*)}\right).$$

Then, we have that $\left\|\vec{a}_i + \vec{a}^*\right\|_2 \leq \frac{8\sqrt{n}\epsilon}{c_a c_e R}$ *and* $t_j \geq 0$ *for all* j*. Furthermore, we have*

$$\left(\sum_{j\neq i}^{O(n)} t_j a_j\right)^T \vec{x}_* - \sum_{j\neq i}^{O(n)} t_j b_j \leq \frac{3n}{c_e} s(\vec{x}_*)_i \quad .$$

Proof. By Lemma 14 and Lemma 21 we know that $\vec{e}(\vec{x}_*, \vec{\tau}_*) \leq \frac{1}{2}c_e$ and $\delta_{\vec{e}(\vec{x}_*,\vec{\tau}_*)} \leq \frac{\epsilon}{R}\sqrt{c_e + \mu(\vec{x}_*)}$. Since $\vec{e}(\vec{x}_*, \vec{\tau}_*) \leq \frac{1}{2}c_e$, we have $t_j \geq 0$ for all j. Furthermore, by Lemma 20 and (6.19), we then have that with high probability in n

$$\begin{aligned}\left\|\vec{a}_i + \vec{a}^*\right\|_2 &\leq \frac{2\epsilon}{(c_e + \mu(\vec{x}_*))}\left[\lambda\left\|\vec{x}_*\right\|_2 + \delta_e(\vec{x}_*)\sqrt{\frac{mc_e + n}{\epsilon^2} + \lambda}\right] \\ &\leq \frac{2\epsilon}{c_e}\left[\frac{1}{c_a R^2}(3\sqrt{n}R) + \frac{\epsilon}{R}\sqrt{\frac{3n}{\epsilon^2} + \frac{n}{c_a R^2}}\right] \\ &\leq \frac{2\epsilon}{c_e}\left[\frac{3\sqrt{n}}{c_a R} + \frac{\sqrt{3n}}{4} + \frac{2\sqrt{n}}{\sqrt{c_a}R}\right].\end{aligned}$$

Hence, we have

$$\left\|\vec{a}_i + \vec{a}^*\right\|_2 \leq \frac{8\sqrt{n}\epsilon}{c_a c_e R}.$$

By Lemma 21 we know that $\vec{e}(\vec{x}_*, \vec{\tau}_*) \leq \frac{1}{2}c_e$ and hence

$$\begin{aligned}\left(\sum_{j\neq i}^{O(n)} t_j a_j\right)^T \vec{x}_* - \sum_{j\neq i}^{O(n)} t_j b_j &= \sum_{j\neq i}^{O(n)} t_j s(\vec{x}_*)_j = s_i(\vec{x}_*)\sum_{j\neq i}^{O(n)}\left(\frac{c_e + e_j(\vec{x}_*, \vec{\tau}_*) + \psi_j(\vec{x}_*)}{c_e + e_i(\vec{x}_*, \vec{\tau}_*) + \psi_i(\vec{x}_*)}\right) \\ &\leq s_i(\vec{x}_*)\sum_{j\neq i}^{O(n)}\left(\frac{\frac{3}{2}c_e + \psi_j(\vec{x}_*)}{\frac{1}{2}c_e + \psi_i(\vec{x}_*)}\right) \leq s_i(\vec{x}_*)\sum_{j\neq i}^{O(n)}\left(\frac{3mc_e + 2n}{c_e}\right) \leq \frac{3n}{c_e}s_i(\vec{x}_*)\end{aligned}$$

□

Lemma 29. *During our Cutting Plane Method, if $p_{\vec{e}}(\vec{x}^{(k)}) \geq n\log(\frac{n}{c_a\epsilon}) + \frac{6n}{c_a}$, then we have $s_i(\vec{x}^{(k)}) \leq \epsilon$ for some i.*

Proof. Recall that

$$p_{\vec{e}}(\vec{x}^{(k)}) = -\sum_{i\in[m]}(c_e + e_i)\log s_i(\vec{x}^{(k)})_i + \frac{1}{2}\log\det\left(\mathbf{A}^T\mathbf{S}_{x^{(k)}}^{-2}\mathbf{A} + \lambda\mathbf{I}\right) + \frac{\lambda}{2}\left\|\vec{x}^{(k)}\right\|_2^2.$$

Using $\left\|\vec{x}^{(k)}\right\| \leq 3\sqrt{n}R$ (6.19) and $\lambda = \frac{1}{c_a R^2}$, we have

$$p_{\vec{e}}(\vec{x}^{(k)}) \leq -\sum_{i\in[m]}(c_e + e_i)\log(s(\vec{x}^{(k)})_i) + \frac{1}{2}\log\det\left(\mathbf{A}^T\mathbf{S}_{x^{(k)}}^{-2}\mathbf{A} + \lambda\mathbf{I}\right) + \frac{5n}{c_a}.$$

Next, we note that $\left\|e_i\right\|_\infty \leq c_e \leq \frac{1}{12\ln(17nR/\epsilon)}$ and $s_i\left(\vec{x}^{(k)}\right) \leq 12\sqrt{n/\lambda} + 4\sqrt{n}R + \sqrt{\frac{1}{c_a\lambda}} \leq 6\sqrt{n}R$ (Lemma 24). Hence, we have

$$p_{\vec{e}}(\vec{x}^{(k)}) \leq \frac{1}{2}\log\det\left(\mathbf{A}^T\mathbf{S}_{x^{(k)}}^{-2}\mathbf{A} + \lambda\mathbf{I}\right) + \frac{6n}{c_a}.$$

Since $p_{\vec{e}}(\vec{x}^{(k)}) \geq n\log(\frac{n}{c_a\epsilon}) + \frac{6n}{c_a}$, we have $\frac{1}{2}\log\det\left(\mathbf{A}^T\mathbf{S}_{x^{(k)}}^{-2}\mathbf{A} + \lambda\mathbf{I}\right) \geq n\log(\frac{n}{c_a\epsilon})$. Using $\epsilon < R$, we have that $\frac{n^2}{c_a^2\epsilon^2} \geq \frac{n^2}{\epsilon^2} + \lambda$ and hence

$$\sum_i \log\lambda_i\left(\mathbf{A}^T\mathbf{S}_x^{-2}\mathbf{A} + \lambda\mathbf{I}\right) \geq n\log\left(\frac{n}{\epsilon^2} + \lambda\right).$$

Therefore, we have $\log \lambda_{\max}\left(\mathbf{A}^T\mathbf{S}_x^{-2}\mathbf{A}+\lambda\mathbf{I}\right) \geq \log\left(\frac{n}{\epsilon^2}+\lambda\right)$. Hence, we have some unit vector $\vec{v}$ such that $\vec{v}\mathbf{A}^T\mathbf{S}_x^{-2}\mathbf{A}\vec{v}+\lambda\vec{v}^T\vec{v} \geq \frac{n}{\epsilon^2}+\lambda$. Thus,

$$\sum_i \frac{(\mathbf{A}\vec{v})_i^2}{s(\vec{x}^{(k)})_i^2} \geq \frac{n}{\epsilon^2}.$$

Therefore there is some i such that $\frac{(\mathbf{A}\vec{v})_i^2}{s(\vec{x}^{(k)})_i^2} \geq \frac{1}{\epsilon^2}$. Since $\vec{a}_i$ and $\vec{v}$ are unit vectors, we have $1 \geq \langle \vec{a}_i, \vec{v}\rangle^2 \geq s(\vec{x}^{(k)})_i^2/\epsilon^2$ and hence $s(\vec{x}^{(k)})_i \leq \epsilon$. □

Lemma 30. *With constant probability, the algorithm ends in $10^{24}n\log(\frac{nR}{\epsilon})$ iterations.* [4]

Proof. Theorem 26 shows that for all k

$$\mathbb{E}p_{\vec{e}(\vec{\tau}^{(k+1)},\vec{x}^{(k+1)})}(\vec{x}^{(k+1)}) \quad \geq \quad p_{\vec{e}(\vec{\tau}^{(k)},\vec{x}^{(k)})}(\vec{x}^{(k)}) - \frac{1}{10^{11}} + \frac{9\beta}{10^{11}} \tag{6.20}$$

where $\beta = 1$ for the case of adding a constraint and $\beta = 0$ for the case of removing a constraint. Now, for all t consider the random variable

$$\mathbf{X}_t = p_{\vec{e}(\vec{\tau}^{(t)},\vec{x}^{(t)})}(\vec{x}^{(t)}) - \frac{4.5m^{(t)}}{10^{11}} - \frac{3.5t}{10^{11}}$$

where $m^{(t)}$ is the number of constraints in iteration t of the algorithm. Then, since $m^{(t+1)} = m^{(t)} - 1 + 2\beta$, (6.20) shows that

$$\begin{aligned}\mathbb{E}\mathbf{X}_{t+1} &\geq p_{\vec{e}(\vec{\tau}^{(t)},\vec{x}^{(t)})}(\vec{x}^{(t)}) - \frac{1}{10^{11}} + \frac{9\beta}{10^{11}} - \frac{4.5m^{(t+1)}}{10^{11}} - \frac{3.5(t+1)}{10^{11}} \\ &= \mathbf{X}_t - \frac{1}{10^{11}} + \frac{9\beta}{10^{11}} - \frac{4.5(-1+2\beta)}{10^{11}} - \frac{3.5}{10^{11}} = \mathbf{X}_t.\end{aligned}$$

Hence, it is a sub-martingale. Let τ be the iteration the algorithm throws out error or outputs $P^{(k)}$. Optional stopping theorem shows that

$$\mathbb{E}\mathbf{X}_{\min(\tau,t)} \geq \mathbb{E}\mathbf{X}_0. \tag{6.21}$$

Using the diameter of $P^{(0)}$ is $\sqrt{n}R$, we have

$$\begin{aligned}p_{\vec{0}}(\vec{0}) &= -\sum_{i\in[m^{(0)}]} c_e \log s_i(\vec{0}) + \frac{1}{2}\log\det\left(\mathbf{A}^T\mathbf{S}_0^{-2}\mathbf{A}+\lambda\mathbf{I}\right) + \frac{\lambda}{2}\left\|\vec{0}\right\|_2^2 \\ &\geq -c_e m^{(0)}\log(\sqrt{n}R) + \frac{n}{2}\log\left(\frac{1}{c_a R^2}\right) \\ &\geq -\left(n + c_e m^{(0)}\right)\log(\sqrt{n}R).\end{aligned}$$

Using $c_e = \frac{c_d}{6\ln(17nR/\epsilon)}$, $c_d = \frac{1}{10^{12}}$ and $m^{(0)} = 2n$, we have

$$\begin{aligned}\mathbf{X}_0 &\geq -\left(n + c_e m^{(0)}\right)\log(\sqrt{n}R) - \frac{4.5m^{(0)}}{10^{11}} \\ &\geq -n\log(\sqrt{n}R) - 100n.\end{aligned}$$

[4] We have made no effort on improving this constant and we believe it can be improved to less than 300 using techniques in [5, 6].

Therefore, (6.21) shows that for all t we have

$$\begin{aligned} -n(\log(nR)+100) &\leq \mathbb{E}\mathbf{X}_{\min(\tau,t)} \\ &= p\mathbb{E}\left[\mathbf{X}_{\min(\tau,t)}|\tau<t\right]+(1-p)\mathbb{E}\left[\mathbf{X}_{\min(\tau,t)}|\tau\geq t\right] \end{aligned} \tag{6.22}$$

where $p \stackrel{\text{def}}{=} \mathbb{P}(\tau < t)$.

Note that

$$\begin{aligned} \mathbb{E}\left[\mathbf{X}_{\min(\tau,t)}|\tau\geq t\right] &\leq \mathbb{E}\left[p_{\vec{e}(\vec{\tau}^{(t)},\vec{x}^{(t)})}(\vec{x}^{(t)})|\tau\geq t\right]-\frac{4.5m^{(t)}}{10^{11}}-\frac{3.5t}{10^{11}}. \\ &\leq \mathbb{E}\left[p_{\vec{e}(\vec{\tau}^{(t)},\vec{x}^{(t)})}(\vec{x}^{(t)})|\tau\geq t\right]-\frac{3.5t}{10^{11}}. \end{aligned}$$

Furthermore, by Lemma 29 we know that when $p_{\vec{e}(\vec{\tau}^{(t)},\vec{x}^{(t)})}(\vec{x}^{(t)}) \geq n\log(\frac{n}{c_a\epsilon})+\frac{6n}{c_a}$, there is a slack that is too small and the algorithm terminates. Hence, we have

$$\mathbb{E}\left[\mathbf{X}_{\min(\tau,t)}|\tau\geq t\right] \leq n\log(\frac{n}{c_a\epsilon})+\frac{6n}{c_a}-\frac{3.5t}{10^{11}}.$$

The proof of Lemma 21 shows that the function value does not change by more than 1 in one iteration by changing $\vec{x}$ and can change by at most $mc_e\log(\frac{3nR}{\epsilon})$ by changing τ. Since by Lemma 22 we know that $m \leq 1+\frac{2n}{c_a}$ and $c_e = \frac{c_d}{6\ln(17nR/\epsilon)}$, we have that $p_{\vec{e}}(\vec{x}) \leq n\log(\frac{n}{c_a\epsilon})+\frac{7n}{c_a}$ throughout the execution of the algorithm. Therefore, we have

$$\mathbb{E}\left[\mathbf{X}_{\min(\tau,t)}|\tau\leq t\right] \leq \mathbb{E}_{\tau<t}p_{\vec{e}(\vec{\tau}^{(t)},\vec{x}^{(t)})}(\vec{x}^{(t)}) \leq n\log(\frac{n}{c_a\epsilon})+\frac{7n}{c_a}.$$

Therefore, (6.22) shows that

$$-n(\log(nR)+100) \leq n\log\left(\frac{n}{c_a\epsilon}\right)+\frac{7n}{c_a}-(1-p)\frac{3.5t}{10^{11}}.$$

Hence, we have

$$\begin{aligned} (1-p)\frac{3.5t}{10^{11}} &\leq n\log\left(\frac{n}{c_a\epsilon}\right)+\frac{7n}{c_a}+n(\log(nR)+100) \\ &\leq n\log\left(\frac{Rn^2}{c_a\epsilon}\right)+\frac{7n}{c_a}+100n \\ &= n\log\left(\frac{Rn^2}{c_a\epsilon}\right)+8\cdot 10^{10}n. \end{aligned}$$

Thus, we have

$$\mathbb{P}(\tau<t)=p\geq 1-\frac{1}{t}\left(10^{11}n\log\left(\frac{n^2R}{\epsilon}\right)+10^{22}n\right).$$

□

Now, we gather all the result as follows:

Theorem 31 (Our Cutting Plane Method)**.** *Let $K \subseteq \mathbb{R}^n$ be a non-empty set contained in a box of radius R, i.e. $K \subseteq B_\infty(R)$. For any $\epsilon \in (0,R)$ in expected time $O(n\mathrm{SO}_{\Omega(\epsilon/\sqrt{n})}(K)\log(nR/\epsilon)+n^3\log^{O(1)}(nR/\epsilon))$ our cutting plane method either outputs $\vec{x} \in K$ or finds a polytope $P=\{\vec{x} : \mathbf{A}\vec{x}\geq\vec{b}\} \supseteq K$ such that*

1. *P has $O(n)$ many constraints (i.e. $\mathbf{A} \in \mathbb{R}^{O(n)\times n}$ and $\vec{b} \in \mathbb{R}^{O(n)}$).*

2. *Each constraint of P is either an initial constraint from $B_\infty(R)$ or of the form $\langle \vec{a}, \vec{x}\rangle \geq b - \delta$ where $\langle \vec{a}, \vec{x}\rangle \geq b$ is a normalized hyperplane (i.e. $\left\|\vec{a}\right\|_2 = 1$) returned by the separation oracle and $\delta = \Omega\left(\frac{\epsilon}{\sqrt{n}}\right)$.*

3. *The polytope P has small width with respect to some direction $\vec{a}_1$ given by one of the constraints, i.e.*
$$\max_{\vec{y}\in P\cap B_\infty(R)} \langle \vec{a}_1, \vec{y}\rangle - \min_{\vec{y}\in P\cap B_\infty(R)} \langle \vec{a}_1, \vec{y}\rangle \leq O\left(n\epsilon \ln(R/\epsilon)\right)$$

4. *Furthermore, the algorithm produces a proof of the fact above involving convex combination of the constraints, namely, non-negatives $t_2, ..., t_{O(n)}$ and $\vec{x} \in P$ such that*

   (a) $\left\|\vec{x}\right\|_2 \leq 3\sqrt{n}R$,

   (b) $\left\|\vec{a}_1 + \sum_{i=2}^{O(n)} t_i \vec{a}_i\right\|_2 = O\left(\frac{\epsilon}{R}\sqrt{n}\log(R/\epsilon)\right)$,

   (c) $\vec{a}_1^T\vec{x} - \vec{b}_1 \leq \epsilon$,

   (d) $\left(\sum_{i=2}^{O(n)} t_i a_i\right)^T \vec{x} - \sum_{i=2}^{O(n)} t_i b_i \leq O(n\epsilon\log(R/\epsilon))$.

Proof. Our algorithm either finds $\vec{x} \in K$ or we have $s_i(\vec{x}^{(k)}) < \epsilon$. When $s_i(\vec{x}^{(k)}) < \epsilon$, we apply Lemma 28 to construct the polytope P and the linear combination $\sum_{i=2}^{O(n)} t_i\vec{a}_i$.

Notice that each iteration of our algorithm needs to solve constant number of linear systems and implements the sampling step to find $\vec{\Delta}^{(k)} \in \mathbb{R}^n$ s.t. $\mathbb{E}[\vec{\Delta}^{(k)}] = \vec{\psi}(\vec{x}^{(k)}) - \vec{\psi}(\vec{x}^{(k-1)})$. Theorem 33 shows how to do the sampling in $\tilde{O}(1)$ many linear systems. Hence, in total, each iterations needs to solve $\tilde{O}(1)$ many linear systems plus nearly linear work. To output the proof for (4), we use Lemma 28.

Note that the linear systems the whole algorithm need to solve is of the form
$$(\mathbf{A}^T\mathbf{S}_x^{-2}\mathbf{A} + \lambda\mathbf{I})^{-1}\vec{x} = \vec{y}.$$
where the matrix $\mathbf{A}^T\mathbf{S}_x^{-2}\mathbf{A} + \lambda\mathbf{I}$ can be written as $\overline{\mathbf{A}}^T\mathbf{D}\overline{\mathbf{A}}$ for the matrix $\overline{\mathbf{A}} = [\mathbf{A}\ \mathbf{I}]$ and diagonal matrix
$$\mathbf{D} = \begin{bmatrix} \mathbf{S}^{-2} & \mathbf{0} \\ \mathbf{0} & \lambda\mathbf{I} \end{bmatrix}.$$
Note that Lemma 14 shows that $\left\| \left(\mathbf{S}^{(k)}\right)^{-1} (\vec{s}^{(k+1)} - \vec{s}^{(k)})\right\|_2 \leq \frac{1}{10}$ for the k^{th} and $(k+1)^{th}$ linear systems we solved in the algorithm. Hence, we have $\left\| \left(\mathbf{D}^{(k)}\right)^{-1} (\vec{d}^{(k+1)} - \vec{d}^{(k)})\right\|_2 \leq \frac{1}{10}$. In [76], they showed how to solve such sequence of systems in $\tilde{O}(n^2)$ amortized cost. Moreover, since our algorithm always changes the constraints by δ amount where $\delta = \Omega(\frac{\epsilon}{\sqrt{n}})$ an inexact separation oracle $\mathrm{SO}_{\Omega(\epsilon/\sqrt{n})}$ suffices. (see Def 1). Consequently, the total work $O(n\mathrm{SO}_{\Omega(\epsilon/\sqrt{n})}(K)\log(nR/\epsilon) + n^3\log^{O(1)}(nR/\epsilon))$. Note that as the running time holds with only constant probability, we can restart the algorithm whenever the running time is too large.

To prove (2), we note that from the algorithm description, we know the constraints are either from $B_\infty(R)$ or of the form $\vec{a}^T\vec{x} \geq \vec{a}^T\vec{x}^{(k)} - \delta$ where
$$\delta = \sqrt{\frac{\vec{a}^T(\mathbf{A}^T\mathbf{S}_{\vec{x}^{(k)}}^{-2}\mathbf{A} + \lambda\mathbf{I})^{-1}\vec{a}}{c_a}}.$$

From the proof of Lemma 29, we know that if $\lambda_{\max}(\mathbf{A}^T\mathbf{S}_x^{-2}\mathbf{A}+\lambda\mathbf{I}) \geq \frac{n}{\epsilon^2}$, then there is $s_i < \epsilon$. Hence, we have $\lambda_{\min}((\mathbf{A}^T\mathbf{S}_x^{-2}\mathbf{A}+\lambda\mathbf{I})^{-1}) \leq \frac{\epsilon^2}{n}$. Since $\vec{a}$ is a unit vector, we have

$$\sqrt{\frac{\vec{a}^T(\mathbf{A}^T\mathbf{S}_{\vec{x}^{(k)}}^{-2}\mathbf{A}+\lambda\mathbf{I})^{-1}\vec{a}}{c_a}} \geq \sqrt{\frac{\epsilon^2}{nc_a}}.$$

□

7 Technical Tools

In this section we provide stand-alone technical tools we use in our cutting plane method in Section 6. In Section 7.1 we show how to efficiently compute accurate estimates of changes in leverage scores using access to a linear system solver. In Section 7.2 we study what we call the "Stochastic Chasing $\vec{0}$ Game" and show how to maintain that a vector is small in ℓ_∞ norm by making small coordinate updates while the vector changes randomly in ℓ_2.

7.1 Estimating Changes in Leverage Scores

In previous sections, we needed to compute leverage scores accurately and efficiently for use in our cutting plane method. Note that the leverage score definition we used was

$$\psi(\vec{w})_i = \vec{\mathbb{1}}_i^T\sqrt{\mathbf{W}}\mathbf{A}\left(\mathbf{A}^T\mathbf{W}\mathbf{A}+\lambda\mathbf{I}\right)^{-1}\mathbf{A}^T\sqrt{\mathbf{W}}\vec{\mathbb{1}}_i$$

for some $\lambda > 0$ which is different from the standard definition

$$\sigma(\vec{w})_i = \vec{\mathbb{1}}_i^T\sqrt{\mathbf{W}}\mathbf{A}\left(\mathbf{A}^T\mathbf{W}\mathbf{A}\right)^{-1}\mathbf{A}^T\sqrt{\mathbf{W}}\vec{\mathbb{1}}_i.$$

However, note that the matrix $\mathbf{A}^T\mathbf{W}\mathbf{A}+\lambda\mathbf{I}$ can be written as $\overline{\mathbf{A}}^T\mathbf{D}\overline{\mathbf{A}}$ for the matrix $\overline{\mathbf{A}} = [\mathbf{A}\ \mathbf{I}]$ and diagonal matrix

$$\mathbf{D} = \begin{bmatrix} \mathbf{W} & \mathbf{0} \\ \mathbf{0} & \lambda\mathbf{I} \end{bmatrix}.$$

and therefore computing ψ is essentially strictly easier than computing typical leverage scores. Consequently, we use the standard definition σ to simplify notation.

In [99], Spielman and Srivastava observed that leverage scores can be written as the norm of certain vectors

$$\sigma(\vec{w})_i = \left\|\sqrt{\mathbf{W}}\mathbf{A}\left(\mathbf{A}^T\mathbf{W}\mathbf{A}\right)^{-1}\mathbf{A}^T\sqrt{\mathbf{W}}\vec{\mathbb{1}}_i\right\|_2^2$$

and therefore leverage scores can be approximated efficiently using dimension reduction. Unfortunately, the error incurred by this approximation is too large to use inside the cutting point method. In this section, we show how to efficiently approximate the *change* of leverage score more accurately.

In particular, we show how to approximate $\sigma(\vec{w}) - \sigma(\vec{v})$ for any given $\vec{w}, \vec{v}$ with $\left\|\log(\vec{w}) - \log(\vec{v})\right\|_2 \ll 1$. Our algorithm breaks $\sigma(\vec{w})_i - \sigma(\vec{v})_i$ into the sum of the norm of small vectors and then uses the Johnson-Lindenstrauss dimension reduction to approximate the norm of each vector separately. Our algorithm makes use of the following version of Johnson-Lindenstrauss.

Lemma 32 ([1]). *Let $0 \leq \epsilon \leq \frac{1}{2}$ and let $\vec{x}_1, ..., \vec{x}_m \in \mathbb{R}^n$ be arbitrary m points. For $k = O(\epsilon^{-2}\log(m))$ let $\mathbf{Q}$ be a $k \times n$ random matrix with each entry sampled from $\{-\frac{1}{\sqrt{k}}, \frac{1}{\sqrt{k}}\}$ uniformly and independently. Then, $\mathbb{E}\left\|\mathbf{Q}\vec{x}_i\right\|^2 = \left\|\vec{x}_i\right\|^2$ for all $i \in [m]$ and with high probability in m we have that for all $i \in [m]$*

$$(1-\epsilon)\left\|\vec{x}_i\right\|^2 \leq \left\|\mathbf{Q}\vec{x}_i\right\|^2 \leq (1+\epsilon)\left\|\vec{x}_i\right\|^2 \quad .$$

Algorithm 3: $\widehat{h} = \texttt{LeverageChange}(\mathbf{A}, \vec{v}, \vec{w}, \epsilon, \alpha)$

Input: $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\vec{v}, \vec{w} \in \mathbb{R}^m_{>0}$, $\epsilon \in (0, 0.5)$.

Given: $\left\|\mathbf{V}^{-1}(\vec{v} - \vec{w})\right\|_2 \leq \frac{1}{10}$ and $\mathbf{A}^T\mathbf{V}\mathbf{A}$ and $\mathbf{A}^T\mathbf{W}\mathbf{A}$ are invertible.

Sample $\mathbf{Q}_d \in \mathbb{R}^{O(\epsilon^{-2}\log(m)) \times n}$ as in Lemma 32.

Let $\hat{d}_i = \left\|\mathbf{Q}_d\sqrt{\mathbf{W}}\mathbf{A}\left(\mathbf{A}^T\mathbf{W}\mathbf{A}\right)^{-1}\mathbf{A}^T\vec{\mathbb{1}}_i\right\|_2^2$ for all $i \in [n]$.

Let $t = O\left(\log(\epsilon^{-1})\right)$.

Sample $\mathbf{Q}_f \in \mathbb{R}^{O(\epsilon^{-2}\log(mt)) \times n}$ as in Lemma 32.

Pick positive integer u randomly such that $\Pr[u = i] = (\frac{1}{2})^i$.

for $j \in \{1, 2, \cdots, t\} \cup \{t + u\}$ **do**

  if j *is even* **then**

    Let $\hat{f}_i^{(j)} = \left\|\mathbf{Q}_f\sqrt{\mathbf{V}}\mathbf{A}\left(\mathbf{A}^T\mathbf{V}\mathbf{A}\right)^{-1}\left(\mathbf{A}^T\left(\mathbf{V} - \mathbf{W}\right)\mathbf{A}\left(\mathbf{A}^T\mathbf{V}\mathbf{A}\right)^{-1}\right)^{\frac{j}{2}}\mathbf{A}^T\vec{\mathbb{1}}_i\right\|_2^2$.

  else

    Let $\Delta^+ \stackrel{\text{def}}{=} (\mathbf{V} - \mathbf{W})^+$, i.e. the matrix $\mathbf{V} - \mathbf{W}$ with negative entries set to 0.

    Let $\Delta^- \stackrel{\text{def}}{=} (\mathbf{W} - \mathbf{V})^+$, i.e. the matrix $\mathbf{W} - \mathbf{V}$ with negative entries set to 0.

    Let $\hat{\alpha}_i^{(j)} = \left\|\mathbf{Q}_f\sqrt{\Delta^+}\mathbf{A}(\mathbf{A}^T\mathbf{V}\mathbf{A})^{-1}(\mathbf{A}^T\left(\mathbf{V} - \mathbf{W}\right)\mathbf{A}\left(\mathbf{A}^T\mathbf{V}\mathbf{A}\right)^{-1})^{\frac{j-1}{2}}\mathbf{A}^T\vec{\mathbb{1}}_i\right\|_2^2$.

    Let $\hat{\beta}_i^{(j)} = \left\|\mathbf{Q}_f\sqrt{\Delta^-}\mathbf{A}(\mathbf{A}^T\mathbf{V}\mathbf{A})^{-1}(\mathbf{A}^T(\mathbf{V} - \mathbf{W})\mathbf{A}(\mathbf{A}^T\mathbf{V}\mathbf{A})^{-1})^{\frac{j-1}{2}}\mathbf{A}^T\vec{\mathbb{1}}_i\right\|_2^2$.

    Let $\hat{f}_i^{(j)} = \hat{\alpha}_i^{(j)} - \hat{\beta}_i^{(j)}$.

  end

end

Let $\hat{f}_i = 2^u\hat{f}_i^{(t+u)} + \sum_{j=1}^t \hat{f}_i^{(j)}$.

Output: $\hat{h}_i = (w_i - v_i)\hat{d}_i + v_i\hat{f}_i$. for all $i \in [m]$

Theorem 33. *Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\vec{v}, \vec{w} \in \mathbb{R}^m_{>0}$ be such that $\alpha \stackrel{\text{def}}{=} \left\|\mathbf{V}^{-1}(\vec{v} - \vec{w})\right\|_2 \leq \frac{1}{10}$ and both $\mathbf{A}^T\mathbf{V}\mathbf{A}$ and $\mathbf{A}^T\mathbf{W}\mathbf{A}$ are invertible. For any $\epsilon \in (0, 0.5)$, Algorithm 3 generates a random variable $\widehat{h}$ such that $\mathbb{E}\widehat{h} = \sigma(\vec{w}) - \sigma(\vec{v})$ and with high probability in m, we have $\|\widehat{h} - (\sigma(\vec{w}) - \sigma(\vec{v}))\|_2 \leq O(\alpha\epsilon)$. Furthermore, the expected running time is $\widetilde{O}((\text{nnz}(\mathbf{A}) + \text{LO})/\epsilon^2)$ where LO is the amount of time needed to apply $\left(\mathbf{A}^T\mathbf{V}\mathbf{A}\right)^{-1}$ and $\left(\mathbf{A}^T\mathbf{W}\mathbf{A}\right)^{-1}$ to a vector.*

Proof. First we bound the running time. To compute $\hat{d}_i, \hat{f}_i^{(j)}, \hat{\alpha}_i^{(j)}, \hat{\beta}_i^{(j)}$, we simply perform matrix multiplications from the left and then consider the dot products with each of the rows of $\mathbf{A}$. Naively this would take time $\widetilde{O}((t+u)^2\log(mt)(\text{nnz}(\mathbf{A}) + \text{LO}))$. However, we can reuse the computation in computing high powers of j to only take time $\widetilde{O}((t+u)\log(mt)(\text{nnz}(\mathbf{A}) + \text{LO}))$. Now since $\mathbb{E}[u]$ is constant we see that the total running time is as desired. It only remains to prove the desired properties of $\hat{h}$.

First we note that we can re-write leverage score differences using

$$\sigma(\vec{w})_i - \sigma(\vec{v})_i = (w_i - v_i)\left[\mathbf{A}\left(\mathbf{A}^T\mathbf{W}\mathbf{A}\right)^{-1}\mathbf{A}^T\right]_{ii} + v_i\left[\mathbf{A}\left(\left(\mathbf{A}^T\mathbf{W}\mathbf{A}\right)^{-1} - \left(\mathbf{A}^T\mathbf{V}\mathbf{A}\right)^{-1}\right)\mathbf{A}^T\right]_{ii} \quad .$$

Consequently, for all $i \in [m]$, if we let

$$\begin{aligned} d_i &\stackrel{\text{def}}{=} \vec{\mathbb{1}}_i^T\mathbf{A}\left(\mathbf{A}^T\mathbf{W}\mathbf{A}\right)^{-1}\mathbf{A}^T\vec{\mathbb{1}}_i, \\ f_i &\stackrel{\text{def}}{=} \vec{\mathbb{1}}_i^T\mathbf{A}\left[\left(\mathbf{A}^T\mathbf{W}\mathbf{A}\right)^{-1} - \left(\mathbf{A}^T\mathbf{V}\mathbf{A}\right)^{-1}\right]\mathbf{A}^T\vec{\mathbb{1}}_i. \end{aligned}$$

then

$$\sigma(\vec{w})_i - \sigma(\vec{v})_i = (w_i - v_i)d_i + (v_i)f_i \quad . \tag{7.1}$$

We show that $\hat{d}_i$ approximates d and $\hat{f}_i$ approximate $\hat{f}$ well enough to satisfy the statements in the Theorem.

First we bound the quality of $\hat{d}_i$. Note that $d_i = \left\|\sqrt{\mathbf{W}}\mathbf{A}\left(\mathbf{A}^T\mathbf{W}\mathbf{A}\right)^{-1}\mathbf{A}^T\vec{\mathbb{1}}_i\right\|_2^2$. Consequently, Lemma 32 shows that $\mathbb{E}[\hat{d}_i] = d_i$ and that with high probability in m we have $(1-\epsilon)d_i \leq \hat{d}_i \leq (1+\epsilon)d_i$ for all $i \in [m]$. Therefore, with high probability in m, we have

$$\begin{aligned}\left\|\left(\vec{w}-\vec{v}\right)\widehat{d}-\left(\vec{w}-\vec{v}\right)\vec{d}\right\|_2^2 &= \sum_{i\in[m]}\left(w_i-v_i\right)^2\left(\widehat{d}_i-d_i\right)^2 \leq \epsilon^2\sum_{i\in[m]}\left(w_i-v_i\right)^2 d_i^2 \\ &= \epsilon^2\sum_{i\in[m]}\left(w_i-v_i\right)^2\left(\frac{\sigma(\vec{w})_i}{\vec{w}_i}\right)^2 \leq 2\epsilon^2\sum_{i\in[m]}\left(\frac{w_i-v_i}{v_i}\right)^2 \quad .\end{aligned}$$

Next we show how to estimate f. Let $\mathbf{X} \stackrel{\text{def}}{=} \left(\mathbf{A}^T\mathbf{V}\mathbf{A}\right)^{-1/2}\mathbf{A}^T\left(\mathbf{V}-\mathbf{W}\right)\mathbf{A}\left(\mathbf{A}^T\mathbf{V}\mathbf{A}\right)^{-1/2}$. By the assumption on α we know $-\frac{1}{2}\mathbf{V} \prec \mathbf{V}-\mathbf{W} \prec \frac{1}{2}\mathbf{V}$ and therefore $-\frac{1}{2}\mathbf{I} \prec \mathbf{X} \prec \frac{1}{2}\mathbf{I}$. Consequently we have that

$$\begin{aligned}\left(\mathbf{A}^T\mathbf{W}\mathbf{A}\right)^{-1} &= \left(\mathbf{A}^T\mathbf{V}\mathbf{A}\right)^{-1/2}\left(\mathbf{I}-\mathbf{X}\right)^{-1}\left(\mathbf{A}^T\mathbf{V}\mathbf{A}\right)^{-1/2} \\ &= \sum_{j=0}^{\infty}\left(\mathbf{A}^T\mathbf{V}\mathbf{A}\right)^{-1/2}\mathbf{X}^j\left(\mathbf{A}^T\mathbf{V}\mathbf{A}\right)^{-1/2}.\end{aligned}$$

and therefore

$$\begin{aligned}f_i &= \vec{\mathbb{1}}_i^T\mathbf{A}\left(\sum_{j=0}^{\infty}\left(\mathbf{A}^T\mathbf{V}\mathbf{A}\right)^{-1/2}\mathbf{X}^j\left(\mathbf{A}^T\mathbf{V}\mathbf{A}\right)^{-1/2}-\left(\mathbf{A}^T\mathbf{V}\mathbf{A}\right)^{-1}\right)\mathbf{A}^T\vec{\mathbb{1}}_i \\ &= \sum_{j=1}^{\infty} f_i^{(j)} \quad \text{where} \quad f_i^{(j)} \stackrel{\text{def}}{=} \vec{\mathbb{1}}_i^T\mathbf{A}\left(\mathbf{A}^T\mathbf{V}\mathbf{A}\right)^{-1/2}\mathbf{X}^j\left(\mathbf{A}^T\mathbf{V}\mathbf{A}\right)^{-1/2}\mathbf{A}^T\vec{\mathbb{1}}_i \quad .\end{aligned}$$

Furthermore, using the definition of $\mathbf{X}$ we have that for even j

$$\begin{aligned}f_i^{(j)} &= \left\|\mathbf{X}^{\frac{j}{2}}\left(\mathbf{A}^T\mathbf{V}\mathbf{A}\right)^{-1/2}\mathbf{A}^T\vec{\mathbb{1}}_i\right\|_2^2 \\ &= \left\|\left(\mathbf{A}^T\mathbf{V}\mathbf{A}\right)^{-1/2}\left(\mathbf{A}^T\left(\mathbf{V}-\mathbf{W}\right)\mathbf{A}\left(\mathbf{A}^T\mathbf{V}\mathbf{A}\right)^{-1}\right)^{\frac{j}{2}}\mathbf{A}^T\vec{\mathbb{1}}_i\right\|_2^2 \\ &= \left\|\sqrt{\mathbf{V}}\mathbf{A}\left(\mathbf{A}^T\mathbf{V}\mathbf{A}\right)^{-1}\left(\mathbf{A}^T\left(\mathbf{V}-\mathbf{W}\right)\mathbf{A}\left(\mathbf{A}^T\mathbf{V}\mathbf{A}\right)^{-1}\right)^{\frac{j}{2}}\mathbf{A}^T\vec{\mathbb{1}}_i\right\|_2^2\end{aligned}$$

For odd j, using our definition of $\mathbf{\Delta}^+$ and $\mathbf{\Delta}^-$ we have that

$$\begin{aligned}f_i^{(j)} &= \vec{\mathbb{1}}_i^T\mathbf{A}\left(\mathbf{A}^T\mathbf{V}\mathbf{A}\right)^{-1/2}\mathbf{X}^j\left(\mathbf{A}^T\mathbf{V}\mathbf{A}\right)^{-1/2}\mathbf{A}^T\vec{\mathbb{1}}_i \\ &= \vec{\mathbb{1}}_i^T\mathbf{A}\left(\left(\mathbf{A}^T\mathbf{V}\mathbf{A}\right)^{-1}\mathbf{A}^T\left(\mathbf{V}-\mathbf{W}\right)\mathbf{A}\right)^{\frac{j-1}{2}}\left(\mathbf{A}^T\mathbf{V}\mathbf{A}\right)^{-1}\mathbf{A}^T\left(\mathbf{V}-\mathbf{W}\right) \\ &\quad \times\mathbf{A}\left(\mathbf{A}^T\mathbf{V}\mathbf{A}\right)^{-1}\left(\mathbf{A}^T\left(\mathbf{V}-\mathbf{W}\right)\mathbf{A}\left(\mathbf{A}^T\mathbf{V}\mathbf{A}\right)^{-1}\right)^{\frac{j-1}{2}}\mathbf{A}^T\vec{\mathbb{1}}_i \\ &= \alpha_i^{(j)}-\beta_i^{(j)}\end{aligned}$$

where

$$\alpha_i^{(j)} \stackrel{\text{def}}{=} \left\| \sqrt{\Delta^+} \mathbf{A} \left(\mathbf{A}^T \mathbf{V} \mathbf{A} \right)^{-1} \left(\mathbf{A}^T \left(\mathbf{W} - \mathbf{V} \right) \mathbf{A} \left(\mathbf{A}^T \mathbf{V} \mathbf{A} \right)^{-1} \right)^{\frac{j-1}{2}} \mathbf{A}^T \vec{\mathbb{1}}_i \right\|_2^2 ,$$

$$\beta_i^{(j)} \stackrel{\text{def}}{=} \left\| \sqrt{\Delta^-} \mathbf{A} \left(\mathbf{A}^T \mathbf{V} \mathbf{A} \right)^{-1} \left(\mathbf{A}^T \left(\mathbf{W} - \mathbf{V} \right) \mathbf{A} \left(\mathbf{A}^T \mathbf{V} \mathbf{A} \right)^{-1} \right)^{\frac{j-1}{2}} \mathbf{A}^T \vec{\mathbb{1}}_i \right\|_2^2 .$$

Consequently, by Lemma 32 and the construction, we see that

$$\mathbb{E} \hat{f}_i = \mathbb{E} \left[\sum_{j=1}^{t} \hat{f}_i^{(j)} + \sum_{u=1}^{\infty} \frac{2^u}{2^u} \hat{f}_i^{(t+u)} \right] = \sum_{j=1}^{\infty} f_i^{(j)} = f_i$$

and therefore $\mathbb{E}\hat{h} = \sigma(\vec{w}) - \sigma(\vec{v})$ as desired. All that remains is to bound the variance of $\hat{f}_i$.

To bound the variance of $\hat{f}$, let $|\mathbf{X}| = \left(\mathbf{A}^T \mathbf{V} \mathbf{A} \right)^{-1/2} \mathbf{A}^T \left| \mathbf{W} - \mathbf{V} \right| \mathbf{A} \left(\mathbf{A}^T \mathbf{V} \mathbf{A} \right)^{-1/2}$. Note that $-\frac{1}{4}\mathbf{I} \preceq -|\mathbf{X}| \preceq \mathbf{X} \preceq |\mathbf{X}| \preceq \frac{1}{4}\mathbf{I}$ and consequently for all j

$$\begin{aligned} g_i^{(j)} &\stackrel{\text{def}}{=} \vec{\mathbb{1}}_i^T \mathbf{A} \left(\mathbf{A}^T \mathbf{V} \mathbf{A} \right)^{-1/2} |\mathbf{X}|^j \left(\mathbf{A}^T \mathbf{V} \mathbf{A} \right)^{-1/2} \mathbf{A}^T \vec{\mathbb{1}}_i \\ &\leq \frac{1}{4^{j-1}} \vec{\mathbb{1}}_i^T \mathbf{A} \left(\mathbf{A}^T \mathbf{V} \mathbf{A} \right)^{-1/2} |\mathbf{X}| \left(\mathbf{A}^T \mathbf{V} \mathbf{A} \right)^{-1/2} \mathbf{A}^T \vec{\mathbb{1}}_i \\ &\stackrel{\text{def}}{=} \frac{1}{v_i 4^{j-1}} \vec{\mathbb{1}}_i^T \mathbf{P}_v \mathbf{\Delta} \mathbf{P}_v \vec{\mathbb{1}}_i \end{aligned}$$

where $\mathbf{P}_v = \sqrt{\mathbf{V}} \mathbf{A} \left(\mathbf{A}^T \mathbf{V} \mathbf{A} \right)^{-1/2} \mathbf{A}^T \sqrt{\mathbf{V}}$ and $\mathbf{\Delta}$ is a diagonal matrix with $\mathbf{\Delta}_{ii} = \left| \frac{w_i - v_i}{v_i} \right|$. Using that $\mathbf{0} \preceq \mathbf{P}_v \preceq \mathbf{I}$, we have that for all j

$$\begin{aligned} (4^{j-1})^2 \sum_{i=1}^{m} \left(v_i g_i^{(j)} \right)^2 &= \sum_{i=1}^{m} \left(\vec{\mathbb{1}}_i^T \mathbf{P}_v \mathbf{\Delta} \mathbf{P}_v \vec{\mathbb{1}}_i \right)^2 = \operatorname{Tr}\left(\mathbf{P}_v \mathbf{\Delta} \mathbf{P}_v \mathbf{P}_v \mathbf{\Delta} \mathbf{P}_v \right) \\ &\leq \operatorname{Tr}\left(\mathbf{P}_v \mathbf{\Delta} \mathbf{\Delta} \mathbf{P}_v \right) = \operatorname{Tr}\left(\mathbf{\Delta} \mathbf{P}_v \mathbf{P}_v \mathbf{\Delta} \right) \\ &\leq \operatorname{Tr}\left(\mathbf{\Delta}^2 \right) = \sum_{i=1}^{m} \left(\frac{w_i - v_i}{v_i} \right)^2 \leq \alpha^2 \end{aligned}$$

and thus $\left\| \mathbf{V} \vec{g}^{(j)} \right\|_2 \leq \frac{4\alpha}{4^j}$. Furthermore, since $\mathbf{\Delta}^+ \preceq |\mathbf{W} - \mathbf{V}|$ and $\mathbf{\Delta}^- \preceq |\mathbf{W} - \mathbf{V}|$ we have that $\left| \alpha_i^{(j)} \right| \leq g_i^{(j)}$ and $\left| \beta_i^{(j)} \right| \leq g_i^{(j)}$. Consequently, by Lemma 32 again, we have

$$\begin{aligned} \left\| \mathbf{V} \hat{f}^{(j)} - \mathbf{V} \vec{f}^{(j)} \right\|_2^2 &= \sum_i v_i^2 \left(\hat{f}_i^{(j)} - f_i^{(j)} \right)^2 \\ &\leq 2 \sum_i v_i^2 \left(\hat{\alpha}_i^{(j)} - \alpha_i^{(j)} \right)^2 + 2 \sum_i v_i^2 \left(\hat{\beta}_i^{(j)} - \beta_i^{(j)} \right)^2 \\ &\leq 2\epsilon^2 \sum_i v_i^2 \left(\left(\alpha_i^{(j)} \right)^2 + \left(\beta_i^{(j)} \right)^2 \right) \\ &\leq 2\epsilon^2 \sum_i \left(v_i g_i^{(j)} \right)^2 \leq \frac{2\alpha^2 \epsilon^2}{\left(4^{j-1} \right)^2} . \end{aligned}$$

Putting this all together we have that

$$\begin{aligned}
\left\|\mathbf{V}\hat{f}-\mathbf{V}\vec{f}\right\|_2 &\leq \left\|2^u\mathbf{V}\hat{f}^{(t+u)}+\sum_{j=1}^{t}\mathbf{V}\hat{f}^{(j)}-\sum_{j=1}^{\infty}\mathbf{V}\vec{f}^{(j)}\right\|_2 \\
&\leq 2^u\left\|\mathbf{V}\hat{f}^{(t+u)}\right\|_2+\sum_{j=1}^{t}\left\|\mathbf{V}\hat{f}^{(j)}-\mathbf{V}\vec{f}^{(j)}\right\|_2+\sum_{j=t+1}^{\infty}\left\|\mathbf{V}\vec{f}^{(j)}\right\|_2 \\
&\leq 2^u\frac{4\alpha}{4^{t+u}}+\sum_{j=1}^{t}\frac{\sqrt{2}\alpha\epsilon}{4^{j-1}}+\sum_{j=t+1}^{\infty}\frac{4\alpha}{4^j} \\
&= O\left(\alpha\epsilon+\frac{\alpha}{4^t}\right).
\end{aligned}$$

Consequently, since $t = O(\log(\epsilon^{-1}))$ we have the desired result. □

7.2 The Stochastic Chasing $\vec{0}$ Game

To avoid computing leverage scores exactly, in Section 7.1 we showed how to estimate the difference of leverage scores and use these to update the leverage scores. However, if we only applied this technique the error of leverage scores would accumulate in the algorithm and we need to fix it. Naturally, one may wish to use dimension reduction to compute a multiplicative approximation to the leverage scores and update our computed value if the error is too large. However, this strategy would fail if there are too many rows with inaccurate leverage scores in the same iteration. In this case, we would change the central point too much that we are not able to recover. In this section, we present this update problem in a general form that we call *Stochastic Chasing 0 game* and provide an effective strategy for playing this game.

The *Stochastic chasing 0 game* is as follows. There is a player, a stochastic adversary, and a point $\vec{x} \in \mathbb{R}^m$. The goal of the player is to keep the point close to $\vec{0} \in \mathbb{R}^m$ in ℓ_∞ norm and the goal of the stochastic adversary is to move $\vec{x}$ away from $\vec{0}$. The game proceeds for an infinite number of iterations where in each iteration the stochastic adversary moves the current point $\vec{x}^{(k)} \in \mathbb{R}^m$ to some new point $\vec{x}^{(k)} + \vec{\Delta}^{(k)} \in \mathbb{R}^m$ and the player needs to respond. The stochastic adversary cannot move the $\vec{\Delta}^{(k)}$ arbitrarily, instead he is only allowed to choose a probability distribution $\mathcal{D}^{(k)}$ and sample $\vec{\Delta}^{(k)}$ from it. Furthermore, it is required that $\mathbb{E}_{\mathcal{D}^{(k)}}\vec{\Delta} = \vec{0}$ and $\left\|\vec{\Delta}\right\|_2^2 \leq c$ for some fixed c and all $\vec{\Delta} \in \mathcal{D}^{(k)}$. The player does not know $\vec{x}^{(k)}$ or the distribution $\mathcal{D}^{(k)}$ or the move $\vec{\Delta}^{(k)}$ of the stochastic adversary. All the player knows is some $\vec{y}^{(k)} \in \mathbb{R}^n$ that is close to $\vec{x}^{(k)}$ in ℓ_∞ norm. With this information, the player is allowed to choose one coordinate i and set $x_i^{(k+1)}$ to be zero and for other j, we have $x_j^{(k+1)} = x_j^{(k)} + \Delta_j^{(k)}$.

The question we would like to address is, what strategy the player should choose to keep $\vec{x}^{(k)}$ close to $\vec{0}$ in ℓ_∞ norm? We show that there is a trivial strategy that performs well: simply pick the

largest coordinate and set it to 0.

Algorithm 4: Stochastic chasing $\vec{0}$ game

Constant: $c > 0, R > 0$.
Let $\vec{x}^{(1)} = \vec{0} \in \mathbb{R}^m$.
for $k = 1$ *to* ∞ **do**
  Stochastic Adversary: Pick $\mathcal{D}^{(k)}$ such that $\mathbb{E}_{\mathcal{D}^{(k)}} \vec{\Delta} = \vec{0}$ and $\left\|\vec{\Delta}\right\|_2 \leq c$ all $\vec{\Delta} \in \mathcal{D}^{(k)}$.
  Stochastic Adversary: Pick $\vec{y}^{(k)} \in \mathbb{R}^m$ such that $\left\|\vec{y}^{(k)} - \vec{x}^{(k)}\right\|_\infty \leq R$.
  Player: Pick a coordinate $i^{(k)}$ using only $\vec{y}^{(k)}$.
  Sample $\vec{\Delta}^{(k)}$ from $\mathcal{D}^{(k)}$.
  Set $x_{i^{(k)}}^{(k+1)} = 0$ and $x_j^{(k+1)} = x_j^{(k)} + \Delta_j^{(k)}$ for all $j \neq i$.
end

Theorem 34. *Using the strategy* $i^{(k)} = \arg\max_i \left|y_i^{(k)}\right|$, *with probability at least* $1 - p$, *we have*

$$\left\|\vec{x}^{(k)}\right\|_\infty \leq 2(c + R) \log\left(4mk^2/p\right)$$

for all k *in the Stochastic Chasing* $\vec{0}$ *Game.*

Proof. Consider the potential function $\Phi(\vec{x}) = \sum_i e^{\alpha x_i} + \sum_i e^{-\alpha x_i}$ where α is to be determined. Now for all x we know that $e^x \leq 1 + x + \frac{x^2}{2} e^{|x|}$ and therefore for all $|\delta| \leq c$, x and α, we have

$$e^{\alpha x + \alpha\delta} \leq e^{\alpha x} + \alpha\delta e^{\alpha x} + \frac{1}{2}\alpha^2\delta^2 e^{\alpha x + |\alpha| c} \quad .$$

Consequently,

$$\begin{aligned}\mathbb{E}_{\vec{\Delta} \in \mathcal{D}^{(k)}} \Phi(\vec{x}^{(k)} + \vec{\Delta}) \leq \Phi(\vec{x}^{(k)}) &+ \alpha \mathbb{E}_{\vec{\Delta} \in \mathcal{D}^{(k)}} \left(\sum_{i \in [m]} e^{\alpha x_i^{(k)}} \Delta_i - \sum_{i \in [m]} e^{-\alpha x_i^{(k)}} \Delta_i \right) \\ &+ \frac{\alpha^2}{2} e^{\alpha \left\|\vec{\Delta}\right\|_\infty} \mathbb{E}_{\vec{\Delta} \in \mathcal{D}^{(k)}} \left(\sum_{i \in [m]} e^{\alpha x_i^{(k)}} \Delta_i^2 + \sum_{i \in [m]} e^{-\alpha x_i^{(k)}} \Delta_i^2 \right) \quad .\end{aligned}$$

Since $\mathbb{E}_{\mathcal{D}^{(k)}} \vec{\Delta} = \vec{0}$ and $\left\|\vec{\Delta}\right\|_2 \leq c$, we have $\mathbb{E}_{\vec{\Delta} \in \mathcal{D}^{(k)}} \left(\sum_i e^{\alpha x_i^{(k)}} \Delta_i - \sum_i e^{-\alpha x_i^{(k)}} \Delta_i \right) = 0$ and

$$\begin{aligned}\mathbb{E}_{\vec{\Delta} \in \mathcal{D}^{(k)}} \left(\sum_i e^{\alpha x_i^{(k)}} \Delta_i^2 + \sum_i e^{-\alpha x_i^{(k)}} \Delta_i^2 \right) &\leq \mathbb{E}_{\vec{\Delta} \in \mathcal{D}^{(k)}} \left(\sum_i \Delta_i^2 \right) \left(\max_i e^{\alpha x_i^{(k)}} + \max_i e^{-\alpha x_i^{(k)}} \right) \\ &\leq c^2 \left(\max_i e^{\alpha x_i^{(k)}} + \max_i e^{-\alpha x_i^{(k)}} \right) \quad .\end{aligned}$$

Letting $\eta^{(k)} = \max\left(\max_i e^{\alpha x_i^{(k)}}, \max_i e^{-\alpha x_i^{(k)}}\right)$, we then have

$$\mathbb{E}_{\vec{\Delta} \in \mathcal{D}^{(k)}} \Phi(\vec{x}^{(k)} + \vec{\Delta}) \leq \Phi(\vec{x}^{(k)}) + \alpha^2 e^{\alpha c} c^2 \eta^{(k)}.$$

Since $i^{(k)} = \arg\max_i \left|y_i^{(k)}\right|$ and $\left\|\vec{y}^{(k)} - \vec{x}^{(k)}\right\|_\infty \leq R$, the player setting $x_{i^{(k)}}^{(k+1)} = 0$ decreases Φ by at least $e^{-\alpha(R+c)} \eta^{(k)}$. Hence, we have

$$\mathbb{E}_{\vec{\Delta} \in \mathcal{D}^{(k)}} \Phi(\vec{x}^{(k+1)}) \quad \leq \quad \Phi(\vec{x}^{(k)}) + \alpha^2 e^{\alpha c} c^2 \eta^{(k)} - e^{-\alpha R} \eta^{(k)}.$$

Picking $\alpha = \frac{1}{2(c+R)}$, we have $e^{2\alpha(c+R)}(\alpha(c+R))^2 \leq 1$ and hence $\alpha^2 e^{\alpha c} c^2 \leq e^{-\alpha(R+c)}$. Therefore, we have that

$$\mathbb{E}_{\vec{\Delta}\in\mathcal{D}^{(k)}}\Phi(\vec{x}^{(k+1)}) \leq \mathbb{E}\Phi(\vec{x}^{(k)}) \leq ... \leq \Phi(\vec{x}^{(1)}) = 2m\,.$$

Consequently, by Markov's inequality we have that $\Pr[\Phi(\vec{x}^{(k)}) \geq \lambda_k] \leq \frac{2m}{\lambda_k}$ for any λ_k. Furthermore, since clearly $\Phi(\vec{x}) \geq e^{\alpha\|\vec{x}\|_\infty}$ we have that $\Pr[\|\vec{x}^{(k)}\|_\infty \geq \log(\lambda_k)/\alpha] \leq \frac{2m}{\lambda_k}$ for all k. Choosing $\lambda_k = \frac{4mk^2}{p}$ and taking a union bound over all k, we have that

$$\left\|\vec{x}^{(k)}\right\|_\infty \leq 2(c+R)\log\left(4mk^2/p\right)$$

for all k with probability at least

$$1 - \sum_{i=1}^{\infty}\frac{2m}{\lambda_k} = 1 - \sum_{k=1}^{\infty}\frac{p}{2k^2} \geq 1 - p \quad .$$

□

Part II
A User's Guide to Cutting Plane Methods

8 Introduction

Cutting plane methods have long been employed to obtain polynomial time algorithms for solving optimization problems. However, for many problems cutting plane methods are often regarded as inefficient both in theory and in practice. Here, in Part II we provide several techniques for applying cutting plane methods efficiently. Moreover, we illustrate the efficacy and versatility of these techniques by applying them to achieve improved running times for solving multiple problems including semidefinite programming, matroid intersection, and submodular flow.

We hope these results revive interest in ellipsoid and cutting plane methods. We believe these results demonstrate how cutting plan methods are often useful not just for showing that a problem is solvable in polynomial time, but in many yield substantial running time improvements. We stress that while some results in Part II are problem-specific, the techniques introduced here are quite general and are applicable to a wide range of problems.

In the remainder of this introduction we survey the key techniques we use to apply our cutting plane method (Section 8.1) and the key results we obtain on improving the running time for solving various optimization problems (Section 8.2). We conclude in Section 8.3 by providing an overview of where to find additional technical result in Part II.

8.1 Techniques

Although cutting plane methods are typically introduced as algorithms for finding a point in a convex set (as we did with the feasibility problem in Part I), this is often not the easiest way to apply the methods. Moreover, improperly applying results on the feasibility problem to solve convex optimization problems can lead to vastly sub-optimal running times. Our central goal, here, in Part II is to provide tools that allow cutting plane methods to be efficiently applied to solve complex optimization problems. Some of these tools are new and some are extensions of previously known techniques. Here we briefly survey the techniques we cover in Section 10 and Section 11.

Technique 0: From Feasibility to Optimization

In Section 10.1, we explain how to use our cutting plane method to solve convex optimization problems using an approximate subgradient oracle. Our result is based on a result of Nemirovski [85] in which he showed how to use a cutting plane method to solve convex optimization problems without smoothness assumptions on the function and with minimal assumptions on the size of the function's domain. We generalize his proof to accommodate for an approximate separation oracle, an extension which is essential for our applications. We use this result as the starting point for two new techniques we discuss below.

Technique 1: Dimension Reduction through Duality

In Section 10.2, we discuss how cutting plane methods can be applied to obtain both primal and dual solutions to convex optimization problems. Moreover, we show how this can be achieved while only applying the cutting plane method in the space, primal or dual, which has a fewer number of variables. Thus we show how to use duality to improve the convergence of cutting plane methods while still solving the original problem.

To illustrate this idea consider the following very simple linear program (LP)

$$\min_{x_i \geq 0, \sum x_i = 1} \sum_{i=1}^{n} w_i x_i$$

where $\vec{x} \in \mathbb{R}^n$ and $\vec{w} \in \mathbb{R}^n$. Although this LP has n variables, it should to be easy to solve purely on the grounds that it only has one equality constraint and thus dual linear program is simply

$$\max_{y \leq w_i \forall i} y \,,$$

i.e. a LP with only one variable. Consequently, we can apply our cutting plane method to solve it efficiently.

However, while this simple example demonstrates how we can use duality to decrease dimensions, it is not always obvious how to recover the optimal primal solution x variable given the optimal dual solution y. Indeed, for many problems their dual is significantly simpler than itself (primal), so some work is required to show that working in the space suffices to require a primal solution.

One such recent example of this approach proving successful is a recent linear programming result [75]. In this result, the authors show how to take advantage of this observation and get a faster LP solver and maximum flow algorithm. It is interesting to study how far this technique can extend, that is, in what settings can one recover the solution to a more difficult dual problem from the solution to its easier primal problem?

There is in fact another precedent for such an approach. Grötschel, Lovász and Schrijver[50] showed how to obtain the primal solution for linear program by using a cutting plane method to solve the linear program exactly. This is based on the observation that cutting plane methods are able to find the active constraints of the optimal solution and hence one can take dual of the linear program to get the dual solution. This idea was further extended in [69] which also observed that cutting plane methods are incrementally building up a LP relaxation of the optimization problem. Hence, one can find a dual solution by taking the dual of that relaxation.

In Section 10.2, we provide a fairly general technique to recover a dual optimal solution from an approximately optimal primal solution. Unfortunately, the performance of this technique seems quite problem-dependent. We therefore only analyze this technique for semidefinite programming (SDP), a classic and popular convex optimization problem. As a result, we obtain a faster SDP solver in both the primal and dual formulations of the problem.

Technique 2: Using Optimization Oracles Directly

In the seminal works of Grötschel, Lovász, Schrijver and independently Karp and Papadimitriou [49, 64], they showed the equivalence between optimization oracles and separation oracles, and gave a general method to construct a separation oracle for a convex set given an optimization oracle for that set, that is an oracle for minimizing linear functionals over the set. This seminal result led to the first weakly polynomial time algorithm for many algorithms such as submodular function minimization. Since then, this idea has been used extensively in various settings [62, 16, 17, 23].

Unfortunately, while this equivalence of separation and optimization is a beautiful and powerful tool for polynomial time solvability of problems, in many case it may lead to inefficient algorithms. In order to use this reduction to get a separation oracle, the optimization oracle may need to be called multiple times – essentially the number of times needed to run a cutting plane method and hence may be detrimental to obtaining small asymptotic running times. Therefore, it is an interesting question of whether there is a way of using an optimization oracle more directly.

In Section 11 we provide a partial answer to this question for the case of a broad class of problems, that we call the *intersection problem*. For these problems we demonstrate how to achieve running time improvements by using optimization oracles directly. The problem we consider is as follows. We wish to solve the problem for some cost vector $\vec{c} \in \mathbb{R}^n$ and convex set K. We assume that the convex set K can be decomposed as $K = K_1 \cap K_2$ such that $\max_{\vec{x} \in K_1} \langle \vec{c}, \vec{x} \rangle$ and $\max_{\vec{x} \in K_2} \langle \vec{c}, \vec{x} \rangle$ can each be solved efficiently. Our goal is to obtain a running time for this problem comparable to that of minimizing K given only a separation oracle for it.

We show that by considering a carefully regularized variant, we obtain a problem such that optimization oracles for K_1 and K_2 immediately yield a separation oracle for this regularized problem. By analyzing the regularizer and bounding the domains of the problem we are able to show that this allows us to efficiently compute highly accurate solutions to the intersection problem by applying our cutting plane method once. In other words, we do not need to use a complicated iterative scheme or directly invoke the equivalence between separation and optimization and thereby save $O(\mathsf{poly}(n))$ factors in our running times.

We note that this intersection problem can be viewed as a generalization of the matroid intersection problem and in Section 11.2, we show our reduction gives a faster algorithm in certain parameter regimes. As another example, in Section 11.3 we show our reduction gives a substantial polynomial improvement for the submodular flow problem. Furthermore, in Section 11.4 we show how our techniques allow us to minimize a linear function over the intersection of a convex set and an affine subspace in a number of iterations that depends only on the co-dimension of the affine space.

8.2 Applications

Our main goal in Part II is to provide general techniques for efficiently using cutting plane methods for various problems. Hence, in Part II we use minimally problem-specific techniques to achieve the best possible running time. However, we also demonstrate the efficacy of our approach by showing how techniques improve upon the previous best known running times for solve several classic problems in combinatorial and continuous optimization. Here we provide a brief overview of these applications, previous work on these problems, and our results.

In order to avoid deviating from our main discussion, our coverage of previous methods and techniques is brief. Given the large body of prior works on SDP, matroid intersection and submodular flow, it would be impossible to have an in-depth discussion on all of them. Therefore, this section focuses on running time comparisons and explanations of relevant preivous techniques.

Semidefinite Programming

In Section 10.2 we consider the classic semidefinite programming (SDP) problem:

$$\max_{\mathbf{X} \succeq \mathbf{0}} \mathbf{C} \bullet \mathbf{X} \text{ s.t. } \mathbf{A}_i \bullet \mathbf{X} = b_i \text{ (primal)} \qquad \min_{\vec{y}} \vec{b}^T \vec{y} \text{ s.t. } \sum_{i=1}^{n} y_i \mathbf{A}_i \succeq \mathbf{C} \text{ (dual)}$$

where $\mathbf{X}$, $\mathbf{C}$, $\mathbf{A}_i$ are $m \times m$ symmetric matrices, $\vec{b}, \vec{y} \in \mathbb{R}^n$, and $\mathbf{A} \bullet \mathbf{B} \stackrel{\text{def}}{=} \text{Tr}(\mathbf{A}^T \mathbf{B})$. For many problems, $n \ll m^2$ and hence the dual problem has fewer variables than the primal. There are many results and applications of SDP; see [106, 101, 83] for a survey on this topic. Since our focus is on polynomial time algorithms, we do not discuss pseudo-polynomial algorithms such as the spectral bundle method [51], multiplicative weight update methods [8, 9, 61, 3], etc.

Authors	Years	Running times
Nesterov, Nemirovsky[89]	1992	$\tilde{O}(\sqrt{n}(nm^\omega + n^{\omega-1}m^2))$
Anstreicher [7]	2000	$\tilde{O}((mn)^{1/4}(nm^\omega + n^{\omega-1}m^2))$
Krishnan, Mitchell [70]	2003	$\tilde{O}(m(n^\omega + m^\omega + S))$ (dual SDP)
This paper	2015	$\tilde{O}(m(n^2 + m^\omega + S))$

Table 8: Previous algorithms for solving a $n \times n$ SDP with m constraints and S non-zeros entries

Currently, there are two competing approaches for solving SDP problems, namely interior point methods (IPM) and cutting plane methods. Typically, IPMs require fewer iterations than the cutting plane methods, however each iteration of these methods is more complicated and possibly more computationally expensive. For SDP problems, interior point methods require the computations of the Hessian of the function $-\log\det\left(\mathbf{C} - \sum_{i=1}^n y_i \mathbf{A}_i\right)$ whereas cutting plane methods usually only need to compute minimum eigenvectors of the slack matrix $\mathbf{C} - \sum_{i=1}^n y_i \mathbf{A}_i$.

In [7], Anstreicher provided the current fastest IPM for solving the dual SDP problem using a method based on the volumetric barrier function. This method takes $O((mn)^{1/4})$ iterations and each iteration is as cheap as usual IPMs. For general matrices $\mathbf{C}, \mathbf{X}, \mathbf{A}_i$, each iteration takes $O(nm^\omega + n^{\omega-1}m^2)$ time where ω is the fast matrix multiplication exponent. If the constraint matrices $\mathbf{A}_i$ are rank one matrices, the iteration cost can be improved to $O(m^\omega + nm^2 + n^2m)$ [71]. If the matrices are sparse, then [40, 84] show how to use matrix completion inside the IPM. However, the running time depends on the extended sparsity patterns which can be much larger than the total number of non-zeros.

In [70], Krishnan and Mitchell observed that the separation oracle for dual SDP takes only $O(m^\omega + S)$ time, where $S = \sum_{i=1}^n \text{nnz}(\mathbf{A}_i)$ be the total number of non-zeros in the constant matrix. Hence, the cutting plane method by [105] gives a faster algorithm for SDP for many regimes. For $\omega = 2.38$, the cutting plane method is faster when $\mathbf{A}_i$ is not rank 1 and the problem is not too dense, i.e. $\sum_{i=1}^n \text{nnz}(\mathbf{A}_i) < n^{0.63}m^{2.25}$. While there are previous methods for using cutting plane methods to obtain primal solutions[69] , to the best of our knowledge, there are no worst case running time analysis for these techniques.

In Section 10.2, show how to alleviate this issue. We provide an improved algorithm for finding the dual solution and prove carefully how to obtain a comparable primal solution as well. See Figure 9.1 for a summary of the algorithms for SDP and their running times.

Matroid Intersection

In Section 11.2 we show how our optimization oracle technique can be used to improve upon the previous best known running times for matroid intersection. Matroid intersection is one of the most fundamental problems in combinatorial optimization. The first algorithm for matroid intersection is due to the seminal paper by Edmonds [26]. In Figures 9.2 and 9.3 we provide a summary of the previous algorithms for unweighted and weighted matroid intersection as well as the new running times we obtain in this paper. While there is no total ordering on the running times of these algorithms due to the different dependence on various parameters, we would like to point out that our algorithms outperform the previous ones in regimes where r is close to n and/or the oracle query costs are relatively expensive. In particular, in terms of oracle query complexity our algorithms are the first to achieve the quadratic bounds of $\tilde{O}(n^2)$ and $\tilde{O}(nr)$ for independence and rank oracles. We hope our work will revive the interest in the problem of which progress has been mostly stagnated for the past 20-30 years.

Authors	Years	Running times
Edmonds [26]	1968	not stated
Aigner, Dowling [2]	1971	$O(nr^2\mathcal{T}_{\text{ind}})$
Tomizawa, Iri [102]	1974	not stated
Lawler [72]	1975	$O(nr^2\mathcal{T}_{\text{ind}})$
Edmonds [28]	1979	not stated
Cunningham [21]	1986	$O(nr^{1.5}\mathcal{T}_{\text{ind}})$
This paper	2015	$O(n^2 \log n\mathcal{T}_{\text{ind}} + n^3 \log^{O(1)} n)$ $O(nr \log^2 n\mathcal{T}_{\text{rank}} + n^3 \log^{O(1)} n)$

Table 9: Previous algorithms for (unweighted) matroid intersection. Here n is the size of the ground set, $r = \max\{r_1, r_2\}$ is the maximum rank of the two matroids, $\mathcal{T}_{\text{ind}}$ is the time needed to check if a set is independent (independence oracle), and $\mathcal{T}_{\text{rank}}$ is the time needed to compute the rank of a given set (rank oracle).

Authors	Years	Running times
Edmonds [26]	1968	not stated
Tomizawa, Iri [102]	1974	not stated
Lawler [72]	1975	$O(nr^2\mathcal{T}_{\text{ind}} + nr^3)$
Edmonds [28]	1979	not stated
Frank [33]	1981	$O(n^2r(\mathcal{T}_{\text{circuit}} + n))$
Orlin, Ahuja [91]	1983	not stated
Brezovec, Cornuéjols, Glover[14]	1986	$O(nr(\mathcal{T}_{\text{circuit}} + r + \log n))$
Fujishige, Zhang [39]	1995	$O(n^2r^{0.5} \log rM \cdot \mathcal{T}_{\text{ind}})$
Shigeno, Iwata [96]	1995	$O((n + \mathcal{T}_{\text{circuit}})nr^{0.5} \log rM)$
This paper	2015	$O((n^2 \log n\mathcal{T}_{\text{ind}} + n^3 \log^{O(1)} n) \log nM)$ $O((nr \log^2 n\mathcal{T}_{\text{rank}} + n^3 \log^{O(1)} n) \log nM)$

Table 10: Previous algorithms for weighted matroid intersection. In additions to the notations used in the unweighted table, $\mathcal{T}_{\text{circuit}}$ is the time needed to find a fundamental circuit and M is the bit complexity of the weights.

Minimum-Cost Submodular Flow

In Section 11.3 we show how our optimization oracle technique can be used to improve upon the previous best known running times for (Minimum-cost) Submodular Flow. Submodular flow is a very general problem in combinatorial optimization which generalizes many problems such as minimum cost flow, the graph orientation, polymatroid intersection, directed cut covering [37]. In Figure 9.4 we provide an overview of the previous algorithms for submodular flow as well as the new running times we obtain in this paper.

Many of the running times are in terms of a parameter h, which is the time required for computing an "exchange capacity". To the best of our knowledge, the most efficient way of computing an exchange capacity is to solve an instance of submodular minimization which previously took time $\tilde{O}(n^4\text{EO} + n^5)$ (and now takes $\tilde{O}(n^2\text{EO} + n^3)$ time using our result in Part III). Readers may wish to substitute $h = \tilde{O}(n^2\text{EO} + n^3)$ when reading the table.

The previous fastest weakly polynomial algorithms for submodular flow are by [59, 30, 32], which take time $\tilde{O}(n^6\text{EO} + n^7)$ and $O(mn^5 \log nU \cdot \text{EO})$, assuming $h = \tilde{O}(n^2\text{EO} + n^3)$. Our algorithm for submodular flow has a running time of $\tilde{O}(n^2\text{EO} + n^3)$, which is significantly faster by roughly a factor of $\tilde{O}(n^4)$.

For strongly polynomial algorithms, our results do not yield a speedup but we remark that our faster strongly polynomial algorithm for submodular minimization in Part III improves the previous algorithms by a factor of $\tilde{O}(n^2)$ as a corollary (because h requires solving an instance of submodular minimization).

8.3 Overview

After providing covering some preliminaries on convex analysis in Section 9 we split the remainder of Part II into Section 10 and Section 11. In Section 10 we cover our algorithm for convex optimization using an approximate subgradient oracle (Section 10.1) as well as our technique on using duality to decrease dimensions and improve the running time of semidefinite programming (Section 10.2). In Section 11 we provide our technique for using minimization oracles to minimize functions over the intersection of convex sets and provide several applications including matroid intersection (Section 11.2), submodular flow (Section 11.3), and minimizing a linear function over the intersection of an affine subspace and a convex set (Section 11.4).

9 Preliminaries

In this section we review basic facts about convex functions that we use throughout Part II. We also introduce two oracles that we use throughout Part II, i.e. subgradient and optimization oracles, and provide some basic reductions between them. Note that we have slightly extended some definitions and facts to accommodate for the noisy separation oracles used in this paper.

First we recall the definition of strong convexity

Definition 35 (Strong Convexity)**.** A real valued function f on a convex set Ω is α-strongly convex if for any $\vec{x}, \vec{y} \in \Omega$ and $t \in [0, 1]$, we have

$$f(t\vec{x} + (1-t)\vec{y}) + \frac{1}{2}\alpha t(1-t)\left\|\vec{x} - \vec{y}\right\|^2 \leq tf(\vec{x}) + (1-t)f(\vec{y}).$$

Next we define an approximate subgradient.

Authors	Years	Running times
Fujishige [35]	1978	not stated
Grötschel, Lovász, Schrijver[49]	1981	weakly polynomial
Zimmermann [113]	1982	not stated
Barahona, Cunningham [12]	1984	not stated
Cunningham, Frank [22]	1985	$\rightarrow O(n^4 h \log C)$
Fujishige [36]	1987	not stated
Frank, Tardos [34]	1987	strongly polynomial
Cui, Fujishige [108]	1988	not stated
Fujishige, Röck, Zimmermann[38]	1989	$\rightarrow O(n^6 h \log n)$
Chung, Tcha [18]	1991	not stated
Zimmermann [114]	1992	not stated
McCormick, Ervolina [82]	1993	$O(n^7 h^* \log nCU)$
Wallacher, Zimmermann [109]	1994	$O(n^8 h \log nCU)$
Iwata [52]	1997	$O(n^7 h \log U)$
Iwata, McCormick, Shigeno [57]	1998	$O\left(n^4 h \min\left\{\log nC, n^2 \log n\right\}\right)$
Iwata, McCormick, Shigeno [58]	1999	$O\left(n^6 h \min\left\{\log nU, n^2 \log n\right\}\right)$
Fleischer, Iwata, McCormick[32]	1999	$O\left(n^4 h \min\left\{\log U, n^2 \log n\right\}\right)$
Iwata, McCormick, Shigeno [59]	1999	$O\left(n^4 h \min\left\{\log C, n^2 \log n\right\}\right)$
Fleischer, Iwata [30]	2000	$O(mn^5 \log nU \cdot \mathrm{EO})$
This paper	2015	$O(n^2 \log nCU \cdot \mathrm{EO} + n^3 \log^{O(1)} nCU)$

Figure 8.1: Previous algorithms for Submodular Flow with n vertices, maximum cost C and maximum capacity U. The factor h is the time for an exchange capacity oracle, h^* is the time for a "more complicated exchange capacity oracle" and EO is the time for evaluation oracle of the submodular function. The arrow,$\rightarrow$, indicates that it used currently best maximum submodular flow algorithm as subroutine which was non-existent at the time of the publication.

Definition 36 (Subgradient)**.** For any convex function f on a convex set Ω, the δ-subgradients of f at x are defined to be

$$\partial_\delta f(\vec{x}) \stackrel{\text{def}}{=} \{\vec{g} \in \Omega : f(\vec{y}) + \delta \geq f(\vec{x}) + \langle \vec{g}, \vec{y} - \vec{x} \rangle \text{ for all } \vec{y} \in \Omega\}.$$

Here we provide some basic facts regarding convexity and subgradients. These statements are natural extensions of well known facts regarding convex functions and their proof can be found in any standard textbook on convex optimization.

Fact 37. *For any convex set Ω and $\vec{x}$ be a point in the interior of Ω, we have the following:*

1. *If f is convex on Ω, then $\partial_0 f(\vec{x}) \neq \emptyset$ and $\partial_s f(\vec{x}) \subseteq \partial_t f(\vec{x})$ for all $0 \leq s \leq t$.Otherwise, we have $\left\|\vec{g}\right\|_2 > \frac{1}{2}\sqrt{\frac{\delta}{D}}$. For any $f(\vec{y}) \leq f(\vec{x})$, we have $\delta \geq \langle \vec{g}, \vec{y} - \vec{x} \rangle$ and hence*
2. *If f is a differential convex function on Ω, then $\nabla f(\vec{x}) \in \partial_0 f(\vec{x})$.*
3. *If f_1 and f_2 are convex function on Ω, $\vec{g}_1 \in \partial_{\delta_1} f_1(\vec{x})$ and $\vec{g}_2 \in \partial_{\delta_2} f_1(\vec{x})$, then $\alpha\vec{g}_1 + \beta\vec{g}_2 \in \partial_{\alpha\delta_1+\beta\delta_2}(\vec{g}_1 + \vec{g}_2)(\vec{x})$.*
4. *If f is α-strongly convex on Ω with minimizer x^*, then for any $\vec{y}$ with $f(\vec{y}) \leq f(\vec{x}^*) + \epsilon$, we have $\frac{1}{2}\alpha\left\|\vec{x}^* - \vec{y}\right\|^2 \leq \epsilon$.*

Next we provide a reduction from subgradients to separation oracles. We will use this reduction several times in Part II to simplify our construction of separation oracles.

Lemma 38. *Let f be a convex function. Suppose we have $\vec{x}$ and $\vec{g} \in \partial_\delta f(\vec{x})$ with $\left\|\vec{x}\right\|_2 \leq 1 \leq D$ and $\delta \leq 1$. If $\left\|\vec{g}\right\|_2 \leq \frac{1}{2}\sqrt{\frac{\delta}{D}}$, then $f(\vec{x}) \leq \min_{\|\vec{y}_2\|_2 \leq D} f(\vec{y}) + 2\sqrt{\delta D}$ and if $\left\|\vec{g}\right\|_2 \leq \frac{1}{2}\sqrt{\frac{\delta}{D}}$ then*

$$\{\left\|\vec{y}\right\|_2 \leq D : f(\vec{y}) \leq f(\vec{x})\} \subset \{\vec{y} : \vec{d}^T\vec{y} \leq \vec{d}^T\vec{x} + 2\sqrt{\delta D}\}$$

with $\vec{d} = \vec{g}/\left\|\vec{g}\right\|_2$. Hence, this gives a $(2\sqrt{\delta D}, 2\sqrt{\delta D})$-separation oracle on the set $\{\left\|\vec{x}\right\|_2 \leq D\}$.

Proof. Let $\vec{y}$ such that $\left\|\vec{y}\right\|_2 \leq D$. By the definition of δ-subgradient, we have

$$f(\vec{y}) + \delta \geq f(\vec{x}) + \langle \vec{g}, \vec{y} - \vec{x} \rangle .$$

If $\left\|\vec{g}\right\| \leq \frac{1}{2}\sqrt{\frac{\delta}{D}}$, then, we have $|\langle \vec{g}, \vec{y} - \vec{x} \rangle| \leq \sqrt{\delta D}$ because $\left\|\vec{x}\right\| \leq D$ and $\left\|\vec{y}\right\|_2 \leq D$. Therefore,

$$\min_{\left\|\vec{y}\right\|_2 \leq D} f(\vec{y}) + 2\sqrt{\delta D} \geq f(\vec{x}).$$

Otherwise, we have $\left\|\vec{g}\right\|_2 > \frac{1}{2}\sqrt{\frac{\delta}{D}}$. For any $f(\vec{y}) \leq f(\vec{x})$, we have $\delta \geq \langle \vec{g}, \vec{y} - \vec{x} \rangle$ and hence

$$2\sqrt{\delta D} \geq \left\langle \frac{\vec{g}}{\left\|\vec{g}\right\|}, \vec{y} - \vec{x} \right\rangle .$$

□

At several times in Part II we will wish to construct subgradient oracles or separation oracles given only the ability to approximately maximize a linear function over a convex set. In the remainder of this section we formally define such a *optimization oracle* and prove this equivalence.

Definition 39 (Optimization Oracle)**.** Given a convex set K and $\delta > 0$ a δ-optimization oracle for K is a function on $\mathbb{R}^n$ such that for any input $\vec{c} \in \mathbb{R}^n$, it outputs $\vec{y}$ such that

$$\max_{\vec{x} \in K} \langle \vec{c}, \vec{x} \rangle \leq \langle \vec{c}, \vec{y} \rangle + \delta.$$

We denote by $\mathrm{OO}_\delta(K)$ the time complexity of this oracle.

Lemma 40. *Given a convex set K, any ϵ-optimization oracle for K is a ϵ-subgradient oracle for $f(\vec{c}) \stackrel{\text{def}}{=} \max_{\vec{x} \in K} \langle \vec{c}, \vec{x} \rangle$.*

Proof. Let $\vec{x}_c$ be the output of ϵ-optimization oracle on the cost vector $\vec{c}$. We have

$$\max_{\vec{x} \in K} \langle \vec{c}, \vec{x} \rangle \leq \langle \vec{c}, \vec{x}_c \rangle + \epsilon.$$

Hence, for all $\vec{d}$, we have and therefore

$$\left\langle \vec{x}_c, \vec{d} - \vec{c} \right\rangle + f(\vec{c}) \leq f(\vec{d}) + \epsilon.$$

Hence, $\vec{x}_c \in \partial_\delta f(\vec{c})$. □

Combining these lemmas shows that having an ϵ-optimization oracle for a convex set K contained in a ball of radius D yields a $O(\sqrt{D\epsilon}, \sqrt{D\epsilon})$ separation oracle for $\max_{x \in K} \langle \vec{c}, \vec{x} \rangle$. We use these ideas to construction separation oracles throughout Part II.

10 Convex Optimization

In this section we show how to apply our cutting plane method to efficiently solve problems in convex optimization. First, in Section 10.1 we show how to use our result to minimize a convex function given an approximate subgradient oracle. Then, in Section 10.2 we illustrate how this result can be used to obtain both primal and dual solutions for a standard convex optimization problems. In particular, we show how our result can be used to obtain improved running times for semidefinite programming across a range of parameters.

10.1 From Feasibility to Optimization

In this section we consider the following standard optimization problem. We are given a convex function $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ and we want to find a point $\vec{x}$ that approximately solves the minimization problem

$$\min_{\vec{x} \in \mathbb{R}^n} f(\vec{x})$$

given only a subgradient oracle for f.

Here we show how to apply the cutting plane method from Part I turning the small width guarantee of the output of that algorithm into a tool to find an approximate minimizer of f. Our result is applicable to any convex optimization problem armed with a separation or subgradient oracle. This result will serve as the foundation for many of our applications in Part II.

Our approach is an adaptation of Nemiroski's method [85] which applies the cutting plane method to solve convex optimiziation problems, with only minimal assumption on the cutting plane method. The proof here is a generalization that accommodates for the noisy separation oracle used in this paper. In the remainder of this subsection we provide a key definition we will use in our algorithm (Defintion 41), provide our main result (Theorem 42), and conclude with a brief discussion of this result.

Definition 41. For any compact set K, we define the *minimum width* by $\text{MinWidth}(K) \stackrel{\text{def}}{=} \min_{\|\vec{a}\|_2=1} \max_{\vec{x},\vec{y}\in K} \langle \vec{a}, \vec{x}-\vec{y}\rangle$.

Theorem 42. *Let f be a convex function on $\mathbb{R}^n$ and Ω be a convex set that contains a minimizer of f. Suppose we have a (η,δ)-separation oracle for f and Ω is contained inside $B_\infty(R)$. Using $B_\infty(R)$ as the initial polytope for our Cutting Plane Method, for any $0<\alpha<1$, we can compute $\vec{x}\in\mathbb{R}^n$ such that*

$$f(\vec{x}) - \min_{\vec{y}\in\Omega} f(\vec{y}) \le \eta + \alpha\left(\max_{\vec{y}\in\Omega} f(\vec{y}) - \min_{\vec{y}\in\Omega} f(\vec{y})\right) \quad . \tag{10.1}$$

with an expected running time of

$$O\left(n\text{SO}_{\eta,\delta}(f)\log\left(\frac{n\kappa}{\alpha}\right) + n^3\log^{O(1)}\left(\frac{n\kappa}{\alpha}\right)\right),$$

where $\delta = \Theta\left(\frac{\alpha\text{MinWidth}(\Omega)}{n^{3/2}\ln(\kappa)}\right)$ and $\kappa = \frac{R}{\text{MinWidth}(\Omega)}$. Furthermore, we only need the oracle defined on the set $B_\infty(R)$.

Proof. Let $\vec{x}^* \in \arg\min_{\vec{x}\in\Omega} f(\vec{x})$. Since $B_\infty(R)\supset\Omega$ contains a minimizer of f, by the definition of (η,δ)-separation oracles, our Cutting Plane Method (Theorem 31) either returns a point $\vec{x}$ that is almost optimal or returns a polytope P of small width. In the former case we have a point $\vec{x}$ such that $f(\vec{x}) \le \min_{\vec{y}} f(\vec{y}) + \eta$. Hence, the error is clearly at most $\eta + \alpha\left(\max_{\vec{z}\in\Omega} f(\vec{z}) - \min_{\vec{x}\in\Omega} f(\vec{x})\right)$ as desired. Consequently, we assume the latter case.

Theorem 31 shows $\text{MinWidth}(P) < Cn\epsilon\ln(R/\epsilon)$ for some universal constant C. Picking

$$\epsilon = C'\frac{\alpha\text{MinWidth}(\Omega)}{n\ln\left(\frac{n\kappa}{\alpha}\right)} \tag{10.2}$$

for small enough constant C', we have $\text{MinWidth}(P^{(i)}) < \alpha\text{MinWidth}(\Omega)$. Let $\Omega^\alpha = \vec{x}^* + \alpha(\Omega - \vec{x}^*)$, namely, $\Omega^\alpha = \{\vec{x}^* + \alpha(\vec{z}-\vec{x}^*) : \vec{z}\in\Omega\}$. Then, we have

$$\text{MinWidth}(\Omega^\alpha) = \alpha\text{MinWidth}(\Omega) > \text{MinWidth}(P).$$

Therefore, Ω^α is not a subset of $P^{(i)}$ and hence there is some point $\vec{y}\in\Omega^\alpha\backslash P$. Since $\Omega^\alpha\subseteq\Omega\subseteq B_\infty(R)$, we know that $\vec{y}$ does not violate any of the constraints of $P^{(0)}$ and therefore must violate one of the constraints added by querying the separation oracle. Therefore, for some $j\le i$, we have

$$\left\langle \vec{c}^{(j-1)}, \vec{y}\right\rangle > \left\langle \vec{c}^{(j-1)}, \vec{x}^{(j-1)}\right\rangle + c_s\epsilon/\sqrt{n} \quad .$$

By the definition of $(\eta, c_s\epsilon/\sqrt{n})$-separation oracle (Definition 2), we have $f(\vec{y}) > f(\vec{x}^{(j-1)})$. Since $\vec{y}\in\Omega^\alpha$, we have $\vec{y} = (1-\alpha)\vec{x}^* + \alpha\vec{z}$ for some $\vec{z}\in\Omega$. Thus, the convexity of f implies that

$$f(\vec{y}) \le (1-\alpha)f(\vec{x}^*) + \alpha f(\vec{z}).$$

Therefore, we have

$$\min_{1\le k\le i} f(\vec{x}^{(k)}) - \min_{\vec{x}\in\Omega} f(\vec{x}) < f(\vec{y}) - f(\vec{x}^*) \le \alpha\left(\max_{\vec{z}\in\Omega} f(\vec{z}) - \min_{\vec{x}\in\Omega} f(\vec{x})\right).$$

Hence, we can simply output the best $\vec{x}$ among all $\vec{x}^{(j)}$ and in either case $\vec{x}$ satisfies (10.1).

Note that we need to call (η,δ)-separation oracle with $\delta = \Omega(\epsilon/\sqrt{n})$ to ensure we do not cut out $\vec{x}^*$. Theorem 31 shows that the algorithm takes $O(n\text{SO}_{\eta,\delta}(f)\log(nR/\epsilon) + n^3\log^{O(1)}(nR/\epsilon))$ expected time, as promised. Furthermore, the oracle needs only be defined on $B_\infty(R)$ as our cutting plane method guarantees $\vec{x}^{(k)}\in B_\infty(R)$ for all k (although if needed, an obvious separating hyperplane can be returned for a query point outside $B_\infty(R)$). □

Observe that this algorithm requires no information about Ω (other than that $\Omega \subseteq B_\infty(R)$) and does not guarantee that the output is in Ω. Hence, even though Ω can be complicated to describe, the algorithm still gives a guarantee related to the gap $\max_{\vec{x}\in\Omega} f(\vec{x}) - \min_{\vec{x}\in\Omega} f(\vec{x})$. For specific applications, it is therefore advantageous to pick a Ω as large as possible while the bound on function value is as small as possible.

Before indulging into specific applications, we remark on the dependence on κ. Using John's ellipsoid, it can be shown that any convex set Ω can be transformed linearly such that (1) $B_\infty(1)$ contains Ω and, (2) $\text{MinWidth}(\Omega) = \Omega(n^{-3/2})$. In other words, κ can be effectively chosen as $O(n^{3/2})$. Therefore if we are able to find such a linear transformation, the running time is simply $O\left(n\text{SO}(f)\log(n/\alpha) + n^3 \log^{O(1)}(n/\alpha)\right)$. Often this can be done easily using the structure of the particular problem and the running time does not depend on the size of domain at all.

10.2 Duality and Semidefinite Programming

In this section we illustrate how our result in Section 10.1 can be used to obtain both primal and dual solutions for standard problems in convex optimization. In particular we show how to obtain improved running times for semidefinite programming.

To explain our approach, consider the following minimax problem

$$\min_{\vec{y}\in Y} \max_{\vec{x}\in X} \langle \mathbf{A}\vec{x}, \vec{y}\rangle + \langle \vec{c}, \vec{x}\rangle + \left\langle \vec{d}, \vec{y} \right\rangle \tag{10.3}$$

where $\vec{x} \in \mathbb{R}^m$ and $\vec{y} \in \mathbb{R}^n$. When $m \gg n$, solving this problem by directly using Part I could lead to an inefficient algorithm with running time at least m^3. In many situations, for any fixed $\vec{y}$, the problem $\max_{\vec{x}\in X} \langle \mathbf{A}\vec{x}, \vec{y}\rangle$ is very easy and hence one can use it as a separation oracle and apply Part I and this would gives a running time almost independent of m. However, this would only give us the $\vec{y}$ variable and it is not clear how to recover $\vec{x}$ variable from it.

In this section we show how to alleviate this issue and give semidefinite programming (SDP) as a concrete example of how to apply this general technique. We do not write down the general version as the running time of the technique seems to be problem specific and faster SDP is already an interesting application.

For the remainder of this section we focus on the semidefinite programming (SDP) problem:

$$\max_{\mathbf{X}\succeq \mathbf{0}} \mathbf{C} \bullet \mathbf{X} \text{ s.t. } \mathbf{A}_i \bullet \mathbf{X} = b_i \tag{10.4}$$

and its dual

$$\min_{\vec{y}} \vec{b}^T \vec{y} \text{ s.t. } \sum_{i=1}^n y_i \mathbf{A}_i \succeq \mathbf{C} \tag{10.5}$$

where $\mathbf{X}$, $\mathbf{C}$, $\mathbf{A}_i$ are $m \times m$ symmetric matrices and $\vec{b}, \vec{y} \in \mathbb{R}^n$. Our approach is partially inspired by one of the key ideas of [51, 70]. These results write down the dual SDP in the form

$$\min_y \vec{b}^T \vec{y} - K \min(\lambda_{\min}(\sum_{i=1}^n y_i \mathbf{A}_i - \mathbf{C}), 0) \tag{10.6}$$

for some large number K and use non-smooth optimization techniques to solve the dual SDP problem. Here, we follow the same approach but instead write it as a max-min problem $\min_{\vec{y}} f_K(\vec{y})$ where

$$f_K(\vec{y}) = \max_{\text{Tr}\mathbf{X}\leq K, \mathbf{X}\succeq \mathbf{0}} \left(\vec{b}^T \vec{y} + \left\langle \mathbf{X}, \mathbf{C} - \sum_{i=1}^n y_i \mathbf{A}_i \right\rangle \right). \tag{10.7}$$

Thus the SDP problem in fact assumes the form (10.3) and many ideas in this section can be generalized to the minimax problem (10.3).

To get a dual solution, we notice that the cutting plane method maintains a subset of the primal feasible solution $\mathrm{conv}(\mathbf{X}_i)$ such that

$$\min_{\vec{y}} \vec{b}^T\vec{y} + \max_{\mathrm{Tr}\mathbf{X}\leq K, \mathbf{X}\succeq \mathbf{0}} \left\langle \mathbf{X}, \mathbf{C} - \sum_{i=1}^n y_i\mathbf{A}_i \right\rangle \sim \min_{\vec{y}} \vec{b}^T\vec{y} + \max_{\mathbf{X}\in \mathrm{conv}(\mathbf{X}_i)} \left\langle \mathbf{X}, \mathbf{C} - \sum_{i=1}^n y_i\mathbf{A}_i \right\rangle.$$

Applying minimax theorem, this shows that there exists an approximation solution $\mathbf{X}$ in $\mathrm{conv}(\mathbf{X}_i)$ for the primal problem. Hence, we can restrict the primal SDP on the polytope $\mathrm{conv}(\mathbf{X}_i)$, this reduces the primal SDP into a linear program which can be solved very efficiently. This idea of getting primal/dual solution from the cutting plane method is quite general and is the main purpose of this example. As a by-product, we have a faster SDP solver in both primal and dual! We remark that this idea has been used as a heuristic to obtain [69] for getting the primal SDP solution and our contribution here is mainly the asymptotic time analysis.

We first show how to construct the separation oracle for SDP. For that we need to compute smallest eigenvector of a matrix. Below, for completeness we provide a folklore result showing we can do this using fast matrix multiplication.

Lemma 43. *Given a $n\times n$ symmetric matrix $\mathbf{Y}$ such that $-R\mathbf{I}\preceq \mathbf{Y}\preceq R\mathbf{I}$, for any $\epsilon>0$, with high probability in n in time $O(n^{\omega+o(1)}\log^{O(1)}(R/\epsilon))$ we can find a unit vector $\vec{u}$ such that $\vec{u}^T\mathbf{Y}\vec{u}\geq \lambda_{\max}(\mathbf{Y})-\epsilon$.*

Proof. Let $\mathbf{B}\stackrel{\mathrm{def}}{=}\frac{1}{R}\mathbf{Y}+\mathbf{I}$. Note that $\mathbf{B}\succeq\mathbf{0}$. Now, we consider the repeated squaring $\mathbf{B}_0=\mathbf{B}$ and $\mathbf{B}_{k+1}=\frac{\mathbf{B}_k^2}{\mathrm{Tr}\mathbf{B}_k^2}$. Let $0\leq\lambda_1\leq\lambda_2\leq\cdots\leq\lambda_n$ be the eigenvalues of $\mathbf{B}$ and $\vec{v}_i$ be the corresponding eigenvectors. Then, it is easy to see the the eigenvalues of $\mathbf{B}_k$ are $\frac{\lambda_i^{2^k}}{\sum_{i=1}^n\lambda_i^{2^k}}$.

Let $\vec{q}$ be a random unit vector and $\vec{r}\stackrel{\mathrm{def}}{=}\mathbf{B}_k\vec{q}$. Now $\vec{q}=\sum\alpha_i\vec{v}_i$ for some α_i such that $\sum\alpha_i^2=1$. Letting

$$\vec{p}=\frac{\sum_{\lambda_i>(1-\delta)\lambda_n}\alpha_i\lambda_i^{2^k}\vec{v}_i}{\sum_{i=1}^n\lambda_i^{2^k}}$$

we have

$$\|\vec{r}-\vec{p}\|_2=\left\|\frac{\sum_{\lambda_i\leq(1-\delta)\lambda_n}\alpha_i\lambda_i^{2^k}\vec{v}_i}{\sum_{i=1}^n\lambda_i^{2^k}}\right\|_2\leq\frac{\sum_{\lambda_i\leq(1-\delta)\lambda_n}\lambda_i^{2^k}}{\sum_{i=1}^n\lambda_i^{2^k}}\leq(1-\delta)^{2^k}n.$$

Letting $k=\log_2\left(\frac{\log(n^{3/2}/\delta)}{\delta}\right)$, we have $\|\vec{r}-\vec{p}\|_2\leq\delta/\sqrt{n}$. Since $\mathbf{0}\preceq\mathbf{B}\preceq 2\mathbf{I}$, we have

$$\begin{aligned}\sqrt{\vec{r}^T\mathbf{B}\vec{r}} &\geq \sqrt{\vec{p}^T\mathbf{B}\vec{p}}-\sqrt{(\vec{r}-\vec{p})^T\mathbf{B}(\vec{r}-\vec{p})}\\ &\geq \sqrt{\vec{p}^T\mathbf{B}\vec{p}}-2\delta/\sqrt{n}.\end{aligned}$$

Note that $\vec{p}$ involves only eigenvectors between $(1-\delta)\lambda_n$ to λ_n. Hence, we have

$$\sqrt{\vec{r}^T\mathbf{B}\vec{r}}\geq\sqrt{(1-\delta)\lambda_n}\|\vec{p}\|_2-2\delta/\sqrt{n}.$$

With constant probability, we have $\alpha_n = \Omega(1/\sqrt{n})$. Hence, we have $\left\|\vec{p}\right\|_2 = \Omega(1/\sqrt{n})$. Using $\mathbf{B} \preceq 2\mathbf{I}$ and $\left\|\vec{p}\right\|_2 \geq \left\|\vec{r}\right\|_2 - \delta/\sqrt{n}$ we have that so long as δ is a small enough universal constant

$$\begin{aligned}\frac{\sqrt{\vec{r}^T\mathbf{B}\vec{r}}}{\left\|\vec{r}\right\|_2} &\geq \frac{\sqrt{(1-\delta)\lambda_n}\left\|\vec{p}\right\|_2 - 2\delta/\sqrt{n}}{\left\|\vec{p}\right\|_2 + \delta/\sqrt{n}} \\ &= (1 - O(\delta))\sqrt{\lambda_n} - O(\delta) \\ &= \sqrt{\lambda_n} - O(\delta\sqrt{R}).\end{aligned}$$

Therefore, we have $\frac{\vec{r}^T\mathbf{Y}\vec{r}}{\left\|\vec{r}\right\|^2} \geq \lambda_{\max}(\mathbf{Y}) - O(R\delta)$. Hence, we can find vector $\vec{r}$ by computing k matrix multiplications. [24] showed that fast matrix multiplication is stable under Frobenius norm, i.e., for any $\eta > 0$, using $O(\log(n/b))$ bits, we can find $\mathbf{C}$ such that $\left\|\mathbf{C} - \mathbf{AB}\right\|_F \leq \frac{1}{b}\left\|\mathbf{A}\right\|\left\|\mathbf{B}\right\|$ in time $O(n^{\omega+\eta})$ where ω is the matrix multiplicative constant. Hence, this algorithm takes only $O(n^{\omega+o(1)}\log^{O(1)}(\delta^{-1}))$ time. The result follows from renormalizing the vector $\vec{r}$, repeating the algorithm $O(\log n)$ times to boost the probability and taking $\delta = \Omega(\epsilon/R)$. □

The following lemma shows how to compute a separation for f_K defined in (10.7).

Lemma 44. *Suppose that $\left\|\mathbf{A}_i\right\|_F \leq M$ and $\left\|\mathbf{C}\right\|_F \leq M$. For any $0 < \epsilon < 1$ and $\vec{y}$ with $\left\|\vec{y}\right\|_2 = O(L)$, with high probability in m, we can compute a (ϵ, ϵ)-separation of f_K on $\{\left\|\vec{x}\right\|_2 \leq L\}$ at $\vec{y}$ in time $O(S + m^{\omega+o(1)}\log^{O(1)}(nKML/\epsilon))$ where where S is the sparsity of the problem defined as $\operatorname{nnz}(\mathbf{C}) + \sum_{i=1}^n \operatorname{nnz}(\mathbf{A}_i)$.*

Proof. Note that $-O(nML)\mathbf{I} \preceq \mathbf{C} - \sum_{i=1}^n y_i\mathbf{A}_i \preceq O(nML)\mathbf{I}$. Using Lemma 43, we can find a vector $\vec{v}$ with $\left\|\vec{v}\right\|_2 = K$ in time $O(m^{\omega+o(1)}\log^{O(1)}(nKML/\delta))$ such that

$$\vec{v}^T\left(\mathbf{C} - \sum_{i=1}^n y_i\mathbf{A}_i\right)\vec{v} \geq \max_{\operatorname{Tr}\mathbf{X}\leq K, \mathbf{X}\succeq\mathbf{0}} \left\langle \mathbf{X}, \mathbf{C} - \sum_{i=1}^n y_i\mathbf{A}_i \right\rangle - \delta. \tag{10.8}$$

In other words, we have a δ-optimization oracle for the function f_K. Lemma 40 shows this yields a δ-subgradient oracle and Lemma 38 then shows this yields a $\left(O(\sqrt{\delta L}), O(\sqrt{\delta L})\right)$-separation oracle on the set $\{\left\|\vec{x}\right\|_2 \leq L\}$. By picking $\delta = \epsilon^2/L$, we have the promised oracle. □

With the separation oracle in hand, we are ready to give the algorithm for SDP:

Theorem 45. *Given a primal-dual semidefinite programming problem in the form (10.4) and (10.5), suppose that for some $M \geq 1$ we have*

1. *$\left\|b\right\|_2 \leq M$, $\left\|\mathbf{C}\right\|_F \leq M$ and $\left\|\mathbf{A}_i\right\|_F \leq M$ for all i.*
2. *The primal feasible set lies inside the region $\operatorname{Tr}\mathbf{X} \leq M$.*
3. *The dual feasible set lies inside the region $\left\|\vec{y}\right\|_\infty \leq M$.*

Let OPT *be the optimum solution of (10.4) and (10.5). Then, with high probability, we can find $\mathbf{X}$ and $\vec{y}$ such that*

1. *$\mathbf{X} \succeq \mathbf{0}$, $\operatorname{Tr}\mathbf{X} = O(M)$, $\sum_i |b_i - \langle \mathbf{X}, \mathbf{A}_i\rangle| \leq \epsilon$ for all i and $\mathbf{C} \bullet \mathbf{X} \geq \text{OPT} - \epsilon$.*
2. *$\left\|\vec{y}\right\|_\infty = O(M)$, $\sum_{i=1}^n y_i\mathbf{A}_i \succeq \mathbf{C} - \epsilon\mathbf{I}$ and $\vec{b}^T\vec{y} \leq \text{OPT} + \epsilon$.*

in expected time $O\left(\left(nS+n^3+nm^{\omega+o(1)}\right)\log^{O(1)}\left(\frac{nM}{\epsilon}\right)\right)$ *where* S *is the sparsity of the problem defined as* $\mathrm{nnz}(\mathbf{C})+\sum_{i=1}^n \mathrm{nnz}(\mathbf{A}_i)$ *and* ω *is the fast matrix multiplication constant.*

Proof. Let $K \geq M$ be some parameter to be determined. Since the primal feasible set is lies inside the region $\mathrm{Tr}\mathbf{X} \leq M \leq K$, we have

$$\begin{aligned}
\min_{\sum_{i=1}^n y_i\mathbf{A}_i \succeq \mathbf{C}} \vec{b}^T\vec{y} &= \max_{\mathbf{X}\succeq\mathbf{0},\mathrm{Tr}\mathbf{X}\leq K,\mathbf{A}_i\bullet\mathbf{X}=b_i} \mathbf{C}\bullet\mathbf{X} \\
&= \max_{\mathbf{X}\succeq\mathbf{0},\mathrm{Tr}\mathbf{X}\leq K} \min_{\vec{y}} \mathbf{C}\bullet\mathbf{X} - \sum_i y_i\left(\mathbf{A}_i\bullet\mathbf{X}-b_i\right) \\
&= \min_{\vec{y}} \max_{\mathbf{X}\succeq\mathbf{0},\mathrm{Tr}\mathbf{X}\leq K} \left(\vec{b}^T\vec{y} + (\mathbf{C}-\sum_i y_i\mathbf{A}_i)\bullet\mathbf{X}\right) \\
&= \min_{\vec{y}} f_K(\vec{y}).
\end{aligned}$$

Lemma 44 shows that it takes $\mathrm{SO}_{\delta,\delta}(f_K) = O(S+m^{\omega+o(1)}\log(nKML/\delta))$ time to compute a (δ,δ)-separation oracle of f_K for any point $\vec{y}$ with $\|\vec{y}\|_\infty = O(L)$ where L is some parameter with $L \geq M$. Taking the radius $R = L$, Theorem 42 shows that it takes $O\left(n\mathrm{SO}_{\delta,\delta}(f_K)\log\left(\frac{n}{\alpha}\right)+n^3\log^{O(1)}\left(\frac{n}{\alpha}\right)\right)$ expected time with $\delta = \Theta\left(\alpha n^{-3/2}L\right)$ to find $\vec{y}$ such that

$$f_K(\vec{y}) - \min_{\left\|\vec{y}\right\|_\infty \leq L} f_K(\vec{y}) \leq \delta + \alpha\left(\max_{\left\|\vec{y}\right\|_\infty\leq L} f_K(\vec{y}) - \min_{\left\|\vec{y}\right\|_\infty\leq L} f_K(\vec{y})\right) \leq \delta + 2\alpha\left(nML+2nKML\right).$$

Picking $\alpha = \frac{\epsilon}{7nMKL}$, we have $f_K(\vec{y}) \leq \min_{\vec{y}} f_K(\vec{y}) + \epsilon$. Therefore,

$$\vec{b}^T\vec{y} + K\max(\lambda_{\max}(\mathbf{C}-\sum_{i=1}^n y_i\mathbf{A}_i),0) \leq \mathrm{OPT}+\epsilon.$$

Let $\beta = \max(\lambda_{\max}(\mathbf{C}-\sum_{i=1}^n y_i\mathbf{A}_i),0)$. Then, we have that $\sum_{i=1}^n y_i\mathbf{A}_i \succeq \mathbf{C}-\beta\mathbf{I}$ and

$$\begin{aligned}
\vec{b}^T\vec{y} &\geq \min_{\sum_{i=1}^n y_i\mathbf{A}_i\succeq\mathbf{C}-\beta\mathbf{I}} \vec{b}^T\vec{y} \\
&= \max_{\mathbf{X}\succeq\mathbf{0}\mathbf{A}_i\bullet\mathbf{X}=b_i} (\mathbf{C}-\beta\mathbf{I})\bullet\mathbf{X} \\
&\geq \mathrm{OPT}-\beta M
\end{aligned}$$

because $\mathrm{Tr}\mathbf{X} \leq M$. Hence, we have

$$\mathrm{OPT}-\beta M+\beta K \leq \vec{b}^T\vec{y} + K\max(\lambda_{\max}(\mathbf{C}-\sum_{i=1}^n y_i\mathbf{A}_i),0) \leq \mathrm{OPT}+\epsilon$$

Putting $K = M+1$, we have $\beta \leq \epsilon$. Thus,

$$\sum_{i=1}^n y_i\mathbf{A}_i \succeq \mathbf{C}-\epsilon\mathbf{I}.$$

This gives the result for the dual with the running time $O\left(\left(nS+n^3+nm^{\omega+o(1)}\right)\log^{O(1)}\left(\frac{nML}{\epsilon}\right)\right)$.

Our Cutting Plane Method accesses the sub-problem

$$\max_{\mathbf{X}\succeq\mathbf{0},\mathrm{Tr}\mathbf{X}\leq K}(\mathbf{C}-\sum_i y_i\mathbf{A}_i)\bullet\mathbf{X}$$

only through the separation oracle. Let $\vec{z}$ be the output of our Cutting Plane Method and $\{\vec{v}_i\vec{v}_i^T\}_{i=1}^{O(n)}$ be the matrices used to construct the separation for the $O(n)$ hyperplanes the algorithm maintains at the end. Let $\vec{u}$ be the maximum eigenvector of $\mathbf{C}-\sum_{i=1}^n z_i\mathbf{A}_i$. Now, we consider a realization of f_K

$$\tilde{f}_K(\vec{y})=\vec{b}^T\vec{y}+\max_{\mathbf{X}\in\mathrm{conv}(K\vec{u}\vec{u}^T,\vec{v}_i\vec{v}_i^T)}\left\langle\mathbf{X},\mathbf{C}-\sum_{i=1}^n y_i\mathbf{A}_i\right\rangle.$$

Since applying our Cutting Plane Method to either f_K or $\tilde{f}_K$ gives the same result, the correctness of the our Cutting Plane Method shows that

$$\tilde{f}_K(\vec{z})\leq\min_{\left\|\vec{y}\right\|_\infty\leq L}\tilde{f}_K(\vec{y})+\epsilon.$$

Note that the function $\tilde{f}_K$ is defined such that $\tilde{f}_K(\vec{z})=f_K(\vec{z})$. Hence, we have

$$\min_{\left\|\vec{y}\right\|_\infty\leq L}f_K(\vec{y})\leq f_K(\vec{z})\leq\tilde{f}_K(\vec{z})\leq\min_{\left\|\vec{y}\right\|_\infty\leq L}\tilde{f}_K(\vec{y})+\epsilon.$$

Also, note that $\tilde{f}_K(\vec{x})\leq f_K(\vec{x})$ for all $\vec{x}$. Hence, we have

$$\min_{\left\|\vec{y}\right\|_\infty\leq L}f_K(\vec{y})-\epsilon\leq\min_{\left\|\vec{y}\right\|_\infty\leq L}\tilde{f}(\vec{y})\leq\min_{\left\|\vec{y}\right\|_\infty\leq L}f_K(\vec{y}).$$

Now, we consider the primal version of $\tilde{f}$, namely

$$g(\mathbf{X})\stackrel{\mathrm{def}}{=}\min_{\left\|\vec{y}\right\|_\infty\leq L}\vec{b}^T\vec{y}+\left\langle\mathbf{X},\mathbf{C}-\sum_{i=1}^n y_i\mathbf{A}_i\right\rangle.$$

Sion's minimax theorem [98] shows that

$$\mathrm{OPT}\geq\max_{\mathbf{X}\in\mathrm{conv}(K\vec{u}\vec{u}^T,\vec{v}_i\vec{v}_i^T)}g(\mathbf{X})=\min_{\left\|\vec{y}\right\|_\infty\leq L}\tilde{f}(\vec{y})\geq\mathrm{OPT}-\epsilon.$$

Therefore, to get the primal solution, we only need to find $\vec{u}$ by Lemma 43 and solve the maximization problem on g. Note that

$$\begin{aligned}g(\mathbf{X}) &= \min_{\left\|\vec{y}\right\|_\infty\leq L}\sum_{i=1}^n y_i\left(b_i-\langle\mathbf{X},\mathbf{A}_i\rangle\right)+\langle\mathbf{X},\mathbf{C}\rangle\\ &= -L\sum_i\left|b_i-\langle\mathbf{X},\mathbf{A}_i\rangle\right|+\langle\mathbf{X},\mathbf{C}\rangle.\end{aligned}$$

For notation simplicity, we write $K\vec{u}\vec{u}^T=\vec{v}_0\vec{v}_0^T$. Then, $\mathbf{X}=\sum_{j=0}^{O(n)}\alpha_j\vec{v}_j\vec{v}_j^T$ for some $\sum\alpha_j=1$ and $\alpha_j\geq 0$. Substituting this into the function g, we have

$$g(\vec{\alpha})=-L\sum_j\left|b_i-\sum_j\alpha_j\vec{v}_j^T\mathbf{A}_i\vec{v}_j\right|+\sum_j\alpha_j\vec{v}_j^T\mathbf{C}\vec{v}_j.$$

Hence, this can be easily written as a linear program with $O(n)$ variables and $O(n)$ constraints in time $O(nS)$. Now, we can apply interior point method to find $\vec{\alpha}$ such that

$$g(\vec{\alpha}) \geq \max_{\mathbf{X} \in \text{conv}(K\vec{u}\vec{u}^T, \vec{v}_i\vec{v}_i^T)} g(\mathbf{X}) - \epsilon \geq \text{OPT} - 2\epsilon.$$

Let the corresponding approximate solution be $\widetilde{\mathbf{X}} = \sum \alpha_j \vec{v}_j \vec{v}_j^T$. Then, we have

$$\left\langle \widetilde{\mathbf{X}}, \mathbf{C} \right\rangle - L \sum_i |b_i - \langle \mathbf{X}, \mathbf{A}_i \rangle| \geq \text{OPT} - 2\epsilon.$$

Now, we let $\tilde{b}_i = \left\langle \widetilde{\mathbf{X}}, \mathbf{A}_i \right\rangle$. Then, we note that

$$\begin{aligned} \left\langle \widetilde{\mathbf{X}}, \mathbf{C} \right\rangle &\leq \max_{\mathbf{X} \succeq \mathbf{0} \mathbf{A}_i \bullet \mathbf{X} = \tilde{b}_i} \mathbf{C} \bullet \mathbf{X} \\ &= \min_{\sum_{i=1}^n y_i \mathbf{A}_i \succeq \mathbf{C}} \tilde{b}_i^T \vec{y} \\ &\leq \text{OPT} + M \sum_i \left| b_i - \left\langle \widetilde{\mathbf{X}}, \mathbf{A}_i \right\rangle \right| \end{aligned}$$

because $\left\|\vec{y}\right\|_\infty \leq M$. Hence, we have

$$\text{OPT} + (M - L) \sum_i \left| b_i - \left\langle \widetilde{\mathbf{X}}, \mathbf{A}_i \right\rangle \right| \geq \left\langle \widetilde{\mathbf{X}}, \mathbf{C} \right\rangle - L \sum_i \left| b_i - \left\langle \widetilde{\mathbf{X}}, \mathbf{A}_i \right\rangle \right| \geq \text{OPT} - 2\epsilon.$$

Now, we put $L = M + 2$, we have

$$\sum_i \left| b_i - \left\langle \widetilde{\mathbf{X}}, \mathbf{A}_i \right\rangle \right| \leq \epsilon.$$

This gives the result for the primal. Note that it only takes $O(n^{5/2} \log^{O(1)}(nM/\epsilon))$ to solve a linear program with $O(n)$ variables and $O(n)$ constraints because we have an explicit interior point deep inside the feasible set, i.e. $\alpha_i = \frac{1}{m}$ for some parameter m [76].[5] Hence, the running time is dominated by the cost of cutting plane method which is $O\left(\left(nS + n^3 + nm^{\omega+o(1)}\right) \log^{O(1)}\left(\frac{nM}{\epsilon}\right)\right)$ by putting $L = M + 2$. □

We leave it as an open problem if it is possible to improve this result by reusing the computation in the separation oracle and achieve a running time of $O\left(\left(nS + n^3 + nm^2\right) \log^{O(1)}\left(\frac{nM}{\epsilon}\right)\right)$.

11 Intersection of Convex Sets

In this section we introduce a general technique to optimize a linear function over the intersection of two convex sets, whenever the linear optimization problem on each of them can be done efficiently. At the very high level, this is accomplished by applying cutting plane to a suitably regularized version of the problem. In Section 11.1 we present the technique and in the remaining sections we provide several applications including, matroid intersection (Section 11.2), submodular flow (Section 11.3), and minimizing over the intersection of an affine subspace and a convex set (Section 11.4).

[5] Without this, the running time of interior point method depends on the bit complexity of the linear programs.

11.1 The Technique

Throughout this section we consider variants of the following general optimization problem

$$\max_{\vec{x}\in K_1\cap K_2} \langle \vec{c}, \vec{x}\rangle \tag{11.1}$$

where $\vec{x}, \vec{c} \in \mathbb{R}^n$, K_1 and K_2 are convex subsets of $\mathbb{R}^n$. We assume that

$$\max_{\vec{x}\in K_1} \|\vec{x}\|_2 < M,\ \max_{\vec{x}\in K_2} \|\vec{x}\|_2 < M,\ \|\vec{c}\|_2 \le M \tag{11.2}$$

for some constant $M \ge 1$ and we assume that

$$K_1 \cap K_2 \neq \emptyset. \tag{11.3}$$

Instead of a separation oracle, we assume that K_1 and K_2 each have optimization oracles (see Section 9).

To solve this problem we first introduce a relaxation for the problem (11.1) that we can optimize efficiently. Because we have only the optimization oracles for K_1 and K_2, we simply have variables $\vec{x}$ and $\vec{y}$ for each of them in the objective. Since the output should (approximately) be in the intersection of K_1 and K_2, a regularization term $-\frac{\lambda}{2}\|\vec{x}-\vec{y}\|_2^2$ is added to force $\vec{x} \approx \vec{y}$ where λ is a large number to be determined later. Furthermore, we add terms to make the problem strong concave.

Lemma 46. *Assume (11.2) and (11.3). For $\lambda \ge 1$, let*

$$f_\lambda(\vec{x},\vec{y}) \stackrel{\text{def}}{=} \frac{1}{2}\langle \vec{c}, \vec{x}\rangle + \frac{1}{2}\langle \vec{c}, \vec{y}\rangle - \frac{\lambda}{2}\|\vec{x}-\vec{y}\|_2^2 - \frac{1}{2\lambda}\|\vec{x}\|_2^2 - \frac{1}{2\lambda}\|\vec{y}\|_2^2 \quad . \tag{11.4}$$

There is an unique maximizer $(\vec{x}_\lambda, \vec{y}_\lambda)$ for the problem $\max_{\vec{x}\in K_1, \vec{y}\in K_2} f_\lambda(\vec{x},\vec{y})$. The maximizer $(\vec{x}_\lambda, \vec{y}_\lambda)$ is a good approximation of the solution of (11.1), i.e. $\|\vec{x}_\lambda - \vec{y}_\lambda\|_2^2 \le \frac{6M^2}{\lambda}$ and

$$\max_{\vec{x}\in K_1\cap K_2} \langle \vec{c}, \vec{x}\rangle \le f_\lambda(\vec{x}_\lambda, \vec{y}_\lambda) + \frac{M^2}{\lambda}. \tag{11.5}$$

Proof. Let $\vec{x}^*$ be a maximizer of $\max_{\vec{x}\in K_1\cap K_2}\langle \vec{c}, \vec{x}\rangle$. By assumption (11.2), $\|\vec{x}^*\|_2 \le M$, and therefore

$$f_\lambda(\vec{x}^*, \vec{x}^*) = \langle \vec{c}, \vec{x}^*\rangle - \frac{\|\vec{x}^*\|_2^2}{\lambda} \ge \max_{\vec{x}\in K_1\cap K_2}\langle \vec{c}, \vec{x}\rangle - \frac{M^2}{\lambda} \quad . \tag{11.6}$$

This shows (11.5). Since f_λ is strongly concave in $\vec{x}$ and $\vec{y}$, there is a unique maximizer $(\vec{x}_\lambda, \vec{y}_\lambda)$. Let $\mathrm{OPT}_\lambda = f_\lambda(\vec{x}_\lambda, \vec{y}_\lambda)$. Then, we have

$$\begin{aligned}\mathrm{OPT}_\lambda &\le \frac{1}{2}\|\vec{c}\|_2\|\vec{x}_\lambda\|_2 + \frac{1}{2}\|\vec{c}\|_2\|\vec{y}_\lambda\|_2 - \frac{\lambda}{2}\|\vec{x}_\lambda - \vec{y}_\lambda\|_2^2 \\ &\le \frac{M^2}{2} + \frac{M^2}{2} - \frac{\lambda}{2}\|\vec{x}_\lambda - \vec{y}_\lambda\|_2^2.\end{aligned}$$

On the other hand, using $\lambda \ge 1$, (11.6) shows that

$$\mathrm{OPT}_\lambda \ge f_\lambda(\vec{x}^*, \vec{x}^*) \ge \max_{\vec{x}\in K_1\cap K_2}\langle \vec{c}, \vec{x}\rangle - \frac{M^2}{\lambda} \ge -2M^2.$$

Hence, we have

$$\|\vec{x}_\lambda - \vec{y}_\lambda\|_2^2 \le \frac{2\left(M^2 - \mathrm{OPT}_\lambda\right)}{\lambda} \le \frac{6M^2}{\lambda}. \tag{11.7}$$

□

Now we write $\max f_\lambda(\vec{x},\vec{y})$ as a max-min problem. The reason for doing this is that the dual approximate solution is much easier to obtain and there is a way to read off a primal approximate solution from a dual approximate solution. This is analogous to the idea in [73] which showed how to convert a cut solution to a flow solution by adding regularization terms into the problem.

Lemma 47. *Assume (11.2) and (11.3). Let $\lambda \geq 2$. For any $\vec{x} \in K_1$ and $\vec{y} \in K_2$, the function f_λ can be represented as*

$$f_\lambda(\vec{x},\vec{y}) = \min_{(\vec{\theta}_1,\vec{\theta}_2,\vec{\theta}_3)\in\Omega} g_\lambda(\vec{x},\vec{y},\vec{\theta}_1,\vec{\theta}_2,\vec{\theta}_3) \tag{11.8}$$

where $\Omega = \{(\vec{\theta}_1,\vec{\theta}_2,\vec{\theta}_3) : \left\|\vec{\theta}_1\right\|_2 \leq 2M,\ \left\|\vec{\theta}_2\right\|_2 \leq M,\ \left\|\vec{\theta}_3\right\|_2 \leq M\}$ and

$$g_\lambda(\vec{x},\vec{y},\vec{\theta}_1,\vec{\theta}_2,\vec{\theta}_3) = \left\langle \frac{\vec{c}}{2} + \lambda\vec{\theta}_1 + \frac{\vec{\theta}_2}{\lambda}, \vec{x} \right\rangle + \left\langle \frac{\vec{c}}{2} - \lambda\vec{\theta}_1 + \frac{\vec{\theta}_3}{\lambda}, \vec{y} \right\rangle + \frac{\lambda}{2}\|\vec{\theta}_1\|_2^2 + \frac{1}{2\lambda}\|\vec{\theta}_2\|_2^2 + \frac{1}{2\lambda}\|\vec{\theta}_3\|_2^2. \tag{11.9}$$

Let $h_\lambda(\vec{\theta}_1,\vec{\theta}_2,\vec{\theta}_3) = \max_{\vec{x}\in K_1,\vec{y}\in K_2} g_\lambda(\vec{x},\vec{y},\vec{\theta}_1,\vec{\theta}_2,\vec{\theta}_3)$. For any $(\vec{\theta}'_1,\vec{\theta}'_2,\vec{\theta}'_3)$ such that $h_\lambda(\vec{\theta}'_1,\vec{\theta}'_2,\vec{\theta}'_3) \leq \min_{(\vec{\theta}_1,\vec{\theta}_2,\vec{\theta}_3)\in\Omega} h_\lambda(\vec{\theta}_1,\vec{\theta}_2,\vec{\theta}_3) + \epsilon$, we know $\vec{z} = \frac{1}{2}(\vec{\theta}'_2 + \vec{\theta}'_3)$ satisfies

$$\max_{\vec{x}\in K_1\cap K_2} \langle \vec{c},\vec{x}\rangle \leq \langle \vec{c},\vec{z}\rangle + \frac{20M^2}{\lambda} + 20\lambda^3\epsilon.$$

and $\left\|\vec{z}-\vec{x}_\lambda\right\|_2 + \left\|\vec{z}-\vec{y}_\lambda\right\|_2 \leq 4\sqrt{2\lambda\epsilon} + \sqrt{\frac{6M^2}{\lambda}}$ where $(\vec{x}_\lambda,\vec{y}_\lambda)$ is the unique maximizer for the problem $\max_{\vec{x}\in K_1,\vec{y}\in K_2} f_\lambda(\vec{x},\vec{y})$.

Proof. Note that for any $\left\|\vec{\xi}\right\|_2 \leq \alpha$, we have

$$-\frac{1}{2}\|\vec{\xi}\|_2^2 = \min_{\left\|\vec{\theta}\right\|_2\leq\alpha} \left\langle \vec{\theta},\vec{\xi}\right\rangle + \frac{1}{2}\|\vec{\theta}\|_2^2$$

Using this and (11.2), we have (11.8) for all $\vec{x} \in K_1$ and $\vec{y} \in K_2$ as desired. Since Ω is closed and bounded set and the function g_λ is concave in $(\vec{x},\vec{y})$ and convex in $(\vec{\theta}_1,\vec{\theta}_2,\vec{\theta}_3)$, Sion's minimax theorem [98] shows that

$$\max_{\vec{x}\in K_1,\vec{y}\in K_2} f_\lambda(\vec{x},\vec{y}) = \min_{(\vec{\theta}_1,\vec{\theta}_2,\vec{\theta}_3)\in\Omega} h_\lambda(\vec{\theta}_1,\vec{\theta}_2,\vec{\theta}_3) \tag{11.10}$$

Since f_λ is strongly concave, there is an unique maximizer $(\vec{x}_\lambda,\vec{y}_\lambda)$ of f_λ. Since h_λ is strongly convex, there is a unique minimizer $(\vec{\theta}^*_1,\vec{\theta}^*_2,\vec{\theta}^*_3)$. By the definition of f_λ and h_λ, we have

$$h_\lambda(\vec{\theta}^*_1,\vec{\theta}^*_2,\vec{\theta}^*_3) \geq g_\lambda(\vec{x}_\lambda,\vec{y}_\lambda,\vec{\theta}^*_1,\vec{\theta}^*_2,\vec{\theta}^*_3) \geq f_\lambda(\vec{x}_\lambda,\vec{y}_\lambda) \quad .$$

Using (11.10), the equality above holds and hence $(\vec{\theta}^*_1,\vec{\theta}^*_2,\vec{\theta}^*_3)$ is the minimizer of $g_\lambda(\vec{x}_\lambda,\vec{y}_\lambda,\vec{\theta}_1,\vec{\theta}_2,\vec{\theta}_3)$ over $(\vec{\theta}_1,\vec{\theta}_2,\vec{\theta}_3)$. Since the domain Ω is large enough that $(\vec{\theta}^*_1,\vec{\theta}^*_2,\vec{\theta}^*_3)$ is an interior point in Ω, the optimality condition of g_λ shows that we have $\vec{\theta}^*_2 = \vec{x}_\lambda$ and $\vec{\theta}^*_3 = \vec{y}_\lambda$.

Since h_λ is $\frac{1}{\lambda}$ strongly convex, we have $\left\|\vec{\theta}'_1 - \vec{\theta}^*_1\right\|_2^2 + \left\|\vec{\theta}'_2 - \vec{\theta}^*_2\right\|_2^2 + \left\|\vec{\theta}'_3 - \vec{\theta}^*_3\right\|_2^2 \leq 2\lambda\epsilon$ (Fact 37). Since $\vec{\theta}^*_2 = \vec{x}_\lambda$ and $\vec{\theta}^*_3 = \vec{y}_\lambda$, we have

$$\left\|\vec{\theta}'_2 - \vec{x}_\lambda\right\|_2^2 + \left\|\vec{\theta}'_3 - \vec{y}_\lambda\right\|_2^2 \leq 2\lambda\epsilon. \tag{11.11}$$

Therefore, we have $\left\|\vec{x}_\lambda - \vec{y}_\lambda\right\|_2 \geq \left\|\vec{\theta}'_2 - \vec{\theta}'_3\right\|_2 - 2\sqrt{2\lambda\epsilon}$, $\left\|\vec{x}_\lambda\right\|_2 \geq \left\|\vec{\theta}'_2\right\|_2 - \sqrt{2\lambda\epsilon}$ and $\left\|\vec{y}_\lambda\right\|_2 \geq \left\|\vec{\theta}'_3\right\|_2 - \sqrt{2\lambda\epsilon}$. Using these, $\left\|\vec{x}_\lambda\right\|_2 \leq M$ and $\left\|\vec{y}_\lambda\right\|_2 \leq M$, we have

$$\begin{aligned}
f_\lambda(\vec{\theta}'_2, \vec{\theta}'_3) &= \frac{1}{2}\left\langle \vec{c}, \vec{\theta}'_2 \right\rangle + \frac{1}{2}\left\langle \vec{c}, \vec{\theta}'_3 \right\rangle - \frac{\lambda}{2}\|\vec{\theta}'_2 - \vec{\theta}'_3\|_2^2 - \frac{1}{2\lambda}\|\vec{\theta}'_2\|_2^2 - \frac{1}{2\lambda}\|\vec{\theta}'_3\|_2^2 \\
&\geq \frac{1}{2}\left\langle \vec{c}, \vec{x}_\lambda \right\rangle + \frac{1}{2}\left\langle \vec{c}, \vec{y}_\lambda \right\rangle - M\sqrt{2\lambda\epsilon} \\
&\quad -\frac{\lambda}{2}\left(\left\|\vec{x}_\lambda - \vec{y}_\lambda\right\|_2 + 2\sqrt{2\lambda\epsilon}\right)^2 \\
&\quad -\frac{1}{2\lambda}\left(\left\|\vec{x}_\lambda\right\|_2 + \sqrt{2\lambda\epsilon}\right)^2 - \frac{1}{2\lambda}\left(\left\|\vec{y}_\lambda\right\|_2 + \sqrt{2\lambda\epsilon}\right)^2 \\
&= \frac{1}{2}\left\langle \vec{c}, \vec{x}_\lambda \right\rangle + \frac{1}{2}\left\langle \vec{c}, \vec{y}_\lambda \right\rangle - \frac{\lambda}{2}\|\vec{x}_\lambda - \vec{y}_\lambda\|_2^2 - \frac{1}{2\lambda}\|\vec{x}_\lambda\|_2^2 - \frac{1}{2\lambda}\|\vec{y}_\lambda\|_2^2 \\
&\quad -M\sqrt{2\lambda\epsilon} - 2\lambda\sqrt{2\lambda\epsilon}\left\|\vec{x}_\lambda - \vec{y}_\lambda\right\|_2 - 4\lambda^2\epsilon \\
&\quad -\frac{1}{\lambda}\left\|\vec{x}_\lambda\right\|_2\sqrt{2\lambda\epsilon} - \epsilon - \frac{1}{\lambda}\left\|\vec{y}_\lambda\right\|_2\sqrt{2\lambda\epsilon} - \epsilon.
\end{aligned}$$

Using $\left\|\vec{x}_\lambda - \vec{y}_\lambda\right\|_2 \leq \sqrt{\frac{6M^2}{\lambda}}$ (Lemma 46), $\left\|\vec{x}_\lambda\right\|_2 < M$ and $\left\|\vec{y}_\lambda\right\|_2 < M$, we have

$$\begin{aligned}
f_\lambda(\vec{\theta}'_2, \vec{\theta}'_3) &\geq f_\lambda(\vec{x}_\lambda, \vec{y}_\lambda) \\
&\quad -M\sqrt{2\lambda\epsilon} - 2\lambda\sqrt{2\lambda\epsilon}\left\|\vec{x}_\lambda - \vec{y}_\lambda\right\|_2 - 4\lambda^2\epsilon \\
&\quad -\frac{1}{\lambda}\left\|\vec{x}_\lambda\right\|_2\sqrt{2\lambda\epsilon} - \epsilon - \frac{1}{\lambda}\left\|\vec{y}_\lambda\right\|_2\sqrt{2\lambda\epsilon} - \epsilon. \\
&\geq f_\lambda(\vec{x}_\lambda, \vec{y}_\lambda) \\
&\quad -M\sqrt{2\lambda\epsilon} - 2\lambda\sqrt{12\epsilon}M - 4\lambda^2\epsilon \\
&\quad -2M\sqrt{2\frac{\epsilon}{\lambda}} - 2\epsilon.
\end{aligned}$$

Since $\lambda \geq 2$, we have

$$f_\lambda(\vec{\theta}'_2, \vec{\theta}'_3) \geq f_\lambda(\vec{x}_\lambda, \vec{y}_\lambda) - 20M\lambda\sqrt{\epsilon} - 10\lambda^2\epsilon.$$

Let $\vec{z} = \frac{\vec{\theta}'_2 + \vec{\theta}'_3}{2}$. Lemma 46 shows that

$$\begin{aligned}
\max_{\vec{x} \in K_1 \cap K_2} \langle \vec{c}, \vec{x} \rangle &\leq \max_{\vec{x} \in K_1, \vec{y} \in K_2} f_\lambda(\vec{x}, \vec{y}) + \frac{M^2}{\lambda} \\
&\leq f_\lambda(\vec{\theta}'_2, \vec{\theta}'_3) + \frac{M^2}{\lambda} + 20M\lambda\sqrt{\epsilon} + 10\lambda^2\epsilon \\
&\leq \langle \vec{c}, \vec{z} \rangle + \frac{20M^2}{\lambda} + 20\lambda^3\epsilon
\end{aligned}$$

because $20M\lambda\sqrt{\epsilon} \leq 10\frac{M^2}{\lambda} + 10\lambda^3\epsilon$. Furthermore, we have

$$\begin{aligned}
\left\|\vec{z} - \vec{x}_\lambda\right\|_2 + \left\|\vec{z} - \vec{y}_\lambda\right\|_2 &\leq \left\|\vec{\theta}'_2 - \vec{x}_\lambda\right\|_2 + \left\|\vec{\theta}'_3 - \vec{y}_\lambda\right\|_2 + \left\|\vec{\theta}'_2 - \vec{\theta}'_3\right\|_2 \\
&\leq 4\sqrt{2\lambda\epsilon} + \sqrt{\frac{6M^2}{\lambda}}.
\end{aligned}$$

□

We now apply our cutting plane method to solve the optimization problem (11.1). First we show how to transform the optimization oracles for K_1 and K_2 to get a separation oracle for h_λ, with the appropriate parameters.

Lemma 48. *Suppose we have a ϵ-optimization oracle for K_1 and K_2 for some $0 < \epsilon < 1$. Then on the set $\{\left\|\vec{\theta}\right\|_2 \le D\}$, we have a $(O(\sqrt{\epsilon\lambda D}), O(\sqrt{\epsilon\lambda D}))$-separation oracle for h_λ with time complexity $\mathrm{OO}_\epsilon(K_1) + \mathrm{OO}_\epsilon(K_2)$.*

Proof. Recall that the function h_λ is defined by

$$\begin{aligned} & h_\lambda(\vec{\theta}_1, \vec{\theta}_2, \vec{\theta}_3) \\ = & \max_{\vec{x}\in K_1, \vec{y}\in K_2} \left(\left\langle \frac{\vec{c}}{2} + \lambda\vec{\theta}_1 + \frac{\vec{\theta}_2}{\lambda}, \vec{x} \right\rangle + \left\langle \frac{\vec{c}}{2} - \lambda\vec{\theta}_1 + \frac{\vec{\theta}_3}{\lambda}, \vec{y} \right\rangle + \frac{\lambda}{2}\left\|\vec{\theta}_1\right\|_2^2 + \frac{1}{2\lambda}\left\|\vec{\theta}_2\right\|_2^2 + \frac{1}{2\lambda}\left\|\vec{\theta}_3\right\|_2^2 \right) \\ = & \max_{\vec{x}\in K_1} \left\langle \frac{\vec{c}}{2} + \lambda\vec{\theta}_1 + \frac{\vec{\theta}_2}{\lambda}, \vec{x} \right\rangle + \max_{\vec{y}\in K_2} \left\langle \frac{\vec{c}}{2} - \lambda\vec{\theta}_1 + \frac{\vec{\theta}_3}{\lambda}, \vec{y} \right\rangle + \frac{\lambda}{2}\left\|\vec{\theta}_1\right\|_2^2 + \frac{1}{2\lambda}\left\|\vec{\theta}_2\right\|_2^2 + \frac{1}{2\lambda}\left\|\vec{\theta}_3\right\|_2^2. \end{aligned}$$

Lemma 40 shows how to compute the subgradient of functions of the form $f(\vec{c}) = \max_{\vec{x}\in K} \langle \vec{c}, \vec{x}\rangle$ using the optimization oracle for K. The rest of the term are differentiable so its subgradient is just the gradient. Hence, by addition rule for subgradients (Fact 37), we have a $O(\epsilon\lambda)$-subgradient oracle for f_λ using a $O(\epsilon)$-optimization oracle for K_1 and K_2. The result then follows from Lemma 38. □

Theorem 49. *Assume (11.2) and (11.3). Suppose that we have ϵ-optimization oracle for every $\epsilon > 0$. For $0 < \delta < 1$, we can find $\vec{z} \in \mathbb{R}^n$ such that*

$$\max_{\vec{x}\in K_1\cap K_2} \langle \vec{c}, \vec{x}\rangle \le \delta + \langle \vec{c}, \vec{z}\rangle$$

and $\left\|\vec{z} - \vec{x}\right\|_2 + \left\|\vec{z} - \vec{y}\right\|_2 \le \delta$ for some $\vec{x} \in K_1$ and $\vec{y} \in K_2$ in time

$$O\left(n\left(\mathrm{OO}_\eta(K_1) + \mathrm{OO}_\eta(K_2)\right) \log\left(\frac{nM}{\delta}\right) + n^3 \log^{O(1)}\left(\frac{nM}{\delta}\right)\right)$$

where $\eta = \Omega\left(\left(\frac{\delta}{nM}\right)^{O(1)}\right)$.

Proof. Setting $\lambda = \frac{40M^2}{\delta^2}$ and $\epsilon = \frac{\delta^7}{10^7 M^6}$ in Lemma 47 we see that so long as we obtain any approximate solution $(\vec{\theta}'_1, \vec{\theta}'_2, \vec{\theta}'_3)$ such that

$$h_\lambda(\vec{\theta}'_1, \vec{\theta}'_2, \vec{\theta}'_3) \le \min_{(\vec{\theta}_1, \vec{\theta}_2, \vec{\theta}_3)\in\Omega} h_\lambda(\vec{\theta}_1, \vec{\theta}_2, \vec{\theta}_3) + \epsilon,$$

then we obtain the point we want. To apply Theorem 42, we use

$$\tilde{h}(\vec{\theta}_1, \vec{\theta}_2, \vec{\theta}_3) = \begin{cases} h_\lambda(\vec{\theta}_1, \vec{\theta}_2, \vec{\theta}_3) & \text{if } (\vec{\theta}_1, \vec{\theta}_2, \vec{\theta}_3) \in \Omega \\ +\infty & \text{else} \end{cases}.$$

Lemma 48 shows that for any $\gamma > 0$ we can obtain a (γ, γ)-separation oracle of $h_\lambda(\vec{\theta})$ by using sufficiently accurate optimization oracles. Since Ω is just a product of ℓ^2 balls, we can produce a separating hyperplane easily when $(\vec{\theta}_1, \vec{\theta}_2, \vec{\theta}_3) \notin \Omega$. Hence, we can obtain a (γ, γ)-separation oracle

of $\tilde{h}(\vec{\theta})$. For simplicity, we use $\vec{\theta}$ to represent $(\vec{\theta}_1, \vec{\theta}_2, \vec{\theta}_3)$. Note that $B_\infty(2M) \supseteq \Omega$ and therefore we can apply Theorem 42 with $R = 2M$ to compute $\vec{\theta'}$ such

$$\tilde{h}(\vec{\theta'}) - \min_{\vec{\theta}\in\Omega} \tilde{h}(\vec{\theta}) \leq \gamma + \alpha \left(\max_{\vec{\theta}\in\Omega} \tilde{h}(\vec{\theta}) - \min_{\vec{\theta}\in\Omega} \tilde{h}(\vec{\theta}) \right)$$

in time $O\left(n\mathrm{SO}_{\gamma,\gamma} \log\left(\frac{n\kappa}{\alpha}\right) + n^3 \log^{O(1)}\left(\frac{n\kappa}{\alpha}\right)\right)$ where $\gamma = \Omega\left(\alpha \mathrm{MinWidth}(\Omega)/n^{O(1)}\right) = \Omega\left(\alpha M/n^{O(1)}\right)$ and $\kappa = \frac{2M}{\mathrm{MinWidth}(\Omega)} = O(1)$. Using $\lambda \geq 1$ and $M \geq 1$, we have

$$\max_{\vec{\theta}\in\Omega} \tilde{h}(\vec{\theta}) - \min_{\vec{\theta}\in\Omega} \tilde{h}(\vec{\theta}) \leq O\left(\lambda M^2\right) \leq O\left(\frac{M^4}{\delta^2}\right).$$

Setting $\alpha = \Theta\left(\frac{\delta^9}{M^{10}}\right)$ with some small enough constant, we have that we can find $\vec{\theta'}$ such that

$$\begin{aligned} h_\lambda(\vec{\theta'}) &\leq \min_{\vec{\theta}\in P} h_\lambda(\vec{\theta}) + \gamma + \alpha O\left(\frac{M^4}{\delta^2}\right) \\ &= \min_{\vec{\theta}\in P} h_\lambda(\vec{\theta}) + O\left(\frac{\delta^7}{M^6}\right) \\ &= \min_{\vec{\theta}\in P} h_\lambda(\vec{\theta}) + \epsilon \end{aligned}$$

in time $O\left(n\mathrm{SO}_{\gamma,\gamma} \log\left(\frac{nM}{\delta}\right) + n^3 \log^{O(1)}\left(\frac{nM}{\delta}\right)\right)$ where $\gamma = \Omega\left(\left(\frac{\delta}{nM}\right)^{O(1)}\right)$. Lemma 48 shows that the cost of (γ,γ)-separation oracle is just $O(\mathrm{OO}_\eta(K_1) + \mathrm{OO}_\eta(K_2))$ where $\eta = \Omega\left(\left(\frac{\delta}{nM}\right)^{O(1)}\right)$. □

Remark 50. Note that the algorithm does not promise that we obtain a point close to $K_1 \cap K_2$. It only promises to give a point that is close to both some point in K_1 and some point in K_2. It appears to the authors that a further assumption is needed to get a point close to $K_1 \cap K_2$. For example, if K_1 and K_2 are two almost parallel lines, it would be difficult to get an algorithm that does not depend on the angle. However, as far as we know, most algorithms tackling this problem are pseudo-polynomial and have polynomial dependence on the angle. Our algorithm depends on the logarithmic of the angle which is useful for combinatorial problems.

This reduction is very useful for problems in many areas including linear programming, semi-definite programming and algorithmic game theory. In the remainder of this section we demonstrate its power by applying it to classical combinatorial problems.

There is however one issue with applying our cutting plane algorithm to these problems. As with other convex optimization methods, only an approximately optimal solution is found. On the other hand, typically an exact solution is insisted in combinatorial optimization. To overcome this gap, we introduce the following lemma which (1) transforms the objective function so that there is only one optimal solution and (2) shows that an approximate solution is close to the optimal solution whenever it is unique. As we shall see in the next two subsections, this allows us to round an approximate solution to an optimal one.

Lemma 51. *Given a linear program* $\min_{\mathbf{A}\vec{x}\geq\vec{b}} \vec{c}^T\vec{x}$ *where* $\vec{x}, \vec{c} \in \mathbb{Z}^n$, $\vec{b} \in \mathbb{Z}^m$ *and* $\mathbf{A} \in \mathbb{Z}^{m\times n}$. *Suppose* $\{\mathbf{A}\vec{x} \geq \vec{b}\}$ *is an integral polytope (i.e. all extreme points are integral) contained in the set* $\{\|\vec{x}\|_\infty \leq M\}$. *Then we can find a random cost vector* $\vec{z} \in \mathbb{Z}^n$ *with* $\|\vec{z}\|_\infty \leq O(n^2M^2\|\vec{c}\|_\infty)$ *such that with constant probability,* $\min_{\mathbf{A}\vec{x}\geq\vec{b}} \vec{z}^T\vec{x}$ *has an unique minimizer* $\vec{x}^*$ *and this minimizer is one of the minimizer(s) of* $\min_{\mathbf{A}\vec{x}\geq\vec{b}} \vec{c}^T\vec{x}$. *Furthermore, if there is an interior point* $\vec{y}$ *such that* $\vec{z}^T\vec{y} < \min_{\mathbf{A}\vec{x}\geq\vec{b}} \vec{z}^T\vec{x} + \delta$, *then* $\|\vec{y} - \vec{x}^*\|_\infty \leq 2nM\delta$.

Proof. The first part of the lemma follows by randomly perturbing the cost vector $\vec{c}$. We consider a new cost vector $\vec{z} = 100n^2M^2\vec{c} + \vec{r}$ where each coordinate of $\vec{r}$ is sampled randomly from $\{0, 1, \cdots, 10nM\}$. [67, Lem 4] shows that the linear program $\min_{\mathbf{A}\vec{x}\geq\vec{b}} \vec{z}^T\vec{x}$ has a unique minimizer with constant probability. Furthermore, it is clear that the minimizer of $\min_{\mathbf{A}\vec{x}\geq\vec{b}} \vec{z}^T\vec{x}$ is a minimizer of $\min_{\mathbf{A}\vec{x}\geq\vec{b}} \vec{c}^T\vec{x}$ (as $\vec{r}_i \ll 100n^2M^2|\vec{c}_i|$).

Now we show the second part of the lemma. Given an interior point $\vec{y}$ of the polytope $\{\mathbf{A}\vec{x} \geq \vec{b}\}$, we can write $\vec{y}$ as a convex combination of the vertices of $\{\mathbf{A}\vec{x} \geq \vec{b}\}$, i.e. $\vec{y} = \sum t_i\vec{v}_i$. Note that $\vec{z}^T\vec{y} = \sum t_i\vec{z}^T\vec{v}_i$. If all $\vec{v}_i$ are not the minimizer, then $\vec{z}^T\vec{v}_i \geq \mathrm{OPT} + 1$ and hence $\vec{z}^T\vec{y} \geq \mathrm{OPT} + 1$ which is impossible. Hence, we can assume that $\vec{v}_1$ is the minimizer. Hence, $\vec{z}^T\vec{v}_i = \mathrm{OPT}$ if $i = 1$ and $\vec{z}^T\vec{v}_i \geq \mathrm{OPT}+1$ otherwise. We then have $\vec{z}^T\vec{y} \geq \mathrm{OPT}+(1-t_1)$ which gives $1-t_1 < \delta$. Finally, the claim follows from $\left\|\vec{y} - \vec{v}_1\right\|_\infty \leq \sum_{i\neq 1} t_i\left\|\vec{v}_i - \vec{v}_1\right\|_\infty \leq 2nM\delta$. □

11.2 Matroid Intersection

Let $M_1 = (E, \mathcal{I}_1)$ and $M_2 = (E, \mathcal{I}_2)$ be two matroids sharing the same ground set. In this section we consider the weighted matroid intersection problem

$$\min_{S\in\mathcal{I}_1\cap\mathcal{I}_2} \vec{w}(S).$$

where $\vec{w} \in \mathbb{R}^E$ and $w(S) \stackrel{\text{def}}{=} \sum_{e\in S} w_e$.

For any matroid $M = (E, \mathcal{I})$, it is well known that the polytope of all independent sets has the following description [28]:

$$\mathrm{conv}(\mathcal{I}_1) = \{\vec{x} \in \mathbb{R}^E \text{ s.t. } 0 \leq x(S) \leq r(S) \text{ for all } S \subseteq E\} \tag{11.12}$$

where r is the rank function for M, i.e. $r(S)$ is the size of the largest independent set that is a subset of S. Furthermore, the polytope of the matroid intersection satisfies $\mathrm{conv}(\mathcal{I}_1 \cap \mathcal{I}_2) = \mathrm{conv}(\mathcal{I}_1) \cap \mathrm{conv}(\mathcal{I}_2)$.

It is well known that the optimization problem

$$\min_{S\in\mathcal{I}_1} w(S) \text{ and } \min_{S\in\mathcal{I}_2} w(S)$$

can be solved efficiently by the greedy method. Given a matroid (polytope), the greedy method finds a maximum weight independent subset by maintaining a candidate independent subset S and iteratively attempts to add new element to S in descending weight. A element i is added to S if $S \cup \{i\}$ is still independent. A proof of this algorithm is well-known and can be found in any standard textbook on combinatorial optimization.

Clearly, the greedy method can be implemented by $O(n)$ calls to the independence oracle (also called membership oracle). For rank oracle, it requires $O(r \log n)$ calls by finding the next element to add via binary search. Therefore, we can apply Theorem 49 to get the following result (note that this algorithm is the fastest if r is close to n for the independence oracle).

Theorem 52. *Suppose that the weights $\vec{w}$ are integer with $\left\|w\right\|_\infty \leq M$. Then, we can find*

$$S \in \arg\min_{S\in\mathcal{I}_1\cap\mathcal{I}_2} w(S)$$

in time $O\left(n\mathrm{GO}\log(nM) + n^3\log^{O(1)}(nM)\right)$ where GO is the cost of greedy method for $\mathcal{I}_1$ and $\mathcal{I}_2$.

Proof. Applying Lemma 51, we can find a new cost $\vec{z}$ such that

$$\min_{\vec{x}\in \mathrm{conv}(\mathcal{I}_1)\cap \mathrm{conv}(\mathcal{I}_2)} \vec{z}^T\vec{x}$$

has an unique solution. Note that for any $\vec{x} \in \mathrm{conv}(\mathcal{I}_1)$, we have $\left\|\vec{x}\right\|_\infty \le 1$. Hence, applying theorem 49, we can find $\vec{q}$ such that $\vec{q}^T\vec{z} \le \mathrm{OPT} + \epsilon$ and $\left\|\vec{q}-\vec{x}\right\|_2 + \left\|\vec{q}-\vec{y}\right\|_2 \le \epsilon$ for some $\vec{x} \in \mathrm{conv}(\mathcal{I}_1)$ and $\vec{y} \in \mathrm{conv}(\mathcal{I}_2)$. Using (11.12), we have the coordinate wise minimum of $\vec{x}, \vec{y}$, i.e. $\min\{\vec{x}, \vec{y}\}$, is in $\mathrm{conv}(\mathcal{I}_1) \cap \mathrm{conv}(\mathcal{I}_2)$. Since $\left\|\vec{q} - \min\{\vec{x}, \vec{y}\}\right\|_2 \le \left\|\vec{q}-\vec{x}\right\|_2 + \left\|\vec{q}-\vec{y}\right\|_2 \le \epsilon$, we have

$$(\min\{\vec{x}, \vec{y}\})^T \vec{z} \le \mathrm{OPT} + nM\epsilon.$$

Hence, we have a feasible point $\min\{\vec{x}, \vec{y}\}$ which has value close to optimal and Lemma 51 shows that $\left\|\min(\vec{x}, \vec{y}) - \vec{s}\right\|_\infty \le 2n^2M^2\epsilon$ where $\vec{s}$ is the optimal solution. Hence, we have $\left\|\vec{q}-\vec{s}\right\|_\infty \le 2n^2M^2\epsilon+\epsilon$. Picking $\epsilon = \frac{1}{6n^2M^2}$, we have $\left\|\vec{q}-\vec{s}\right\|_\infty < \frac{1}{2}$ and hence, we can get the optimal solution by rounding to the nearest integer.

Since optimization over $\mathcal{I}_1$ and $\mathcal{I}_2$ involves applying greedy method on certain vectors, it takes only $O(\mathrm{GO})$ time. Theorem 49 shows it only takes $O\left(n\mathrm{GO}\log(nM) + n^3\log^{O(1)}(nM)\right)$ in finding such $\vec{q}$. □

This gives the following corollary.

Corollary 53. *We have $O(n^2\mathcal{T}_{ind}\log(nM)+n^3\log^{O(1)} nM)$ and $O(nr\mathcal{T}_{rank}\log n\log(nM)+n^3\log^{O(1)} nM)$ time algorithms for weighted matroid intersection. Here $\mathcal{T}_{ind}$ is the time needed to check if a subset is independent, and $\mathcal{T}_{rank}$ is the time needed to compute the rank of a given subset.*

Proof. By Theorem 52, it suffices to show that the optimization oracle for the matroid polytope can be implemented in $O(n\mathcal{T}_{\mathrm{ind}})$ and $O(r\mathcal{T}_{\mathrm{rank}}\log n)$ time. This is simply attained by the well-known greedy algorithm which iterates through all the positively-weighted elements in decreasing order, and adds an element to our candidate independent set whenever possible.

For the independence oracle, this involves one oracle call for each element. On the other hand, for the rank oracle, we can find the next element to add by binary search which takes time $O(\mathcal{T}_{\mathrm{rank}}\log n)$. Since there are at most r elements to add, we have the desired running time. □

11.3 Submodular Flow

Let $G = (V, E)$ be a directed graphwith $|E| = m$, let f be a submodular function on $\mathbb{R}^V$ with $|V| = n$, $f(\emptyset) = 0$ and $f(V) = 0$, and let A be the incidence matrix of G. In this section we consider the submodular flow problem

$$\begin{aligned} \text{Minimize} \quad & \langle c, \varphi\rangle \\ \text{subject to} \quad & l(e) \le \varphi(e) \le u(e) \quad \forall e \in E \\ & x(v) = (A\varphi)(v) \quad \forall v \in V \\ & \sum_{v\in S} x(v) \le f(S) \quad \forall S \subseteq V \end{aligned} \tag{11.13}$$

where $c \in \mathbb{Z}^E$, $l \in \mathbb{Z}^E$, $u \in \mathbb{Z}^E$ where $C = \left\|\vec{c}\right\|_\infty$ and $U = \max\left(\left\|u\right\|_\infty, \left\|l\right\|_\infty, \max_{S\subset V}|f(S)|\right)$. Here c is the cost on edges, φ is the flow on edges, l and u are lower and upper bounds on the amount of flow on the edges, and $x(v)$ is the net flow out of vertex v. The submodular function f upper bounds the total net flow out of any subset S of vertices by $f(S)$.

Theorem 54. *Suppose that the cost vector $\vec{c}$ is integer weight with $\left\|\vec{c}\right\|_\infty \le C$ and the capacity vector and the submodular function satisfy $U = \max\left(\left\|u\right\|_\infty, \left\|l\right\|_\infty, \max_{S\subset V}|f(S)|\right)$. Then, we can solve the submodular flow problem (11.13) in time $O\left(n^2 \mathrm{EO}\log(mCU) + n^3\log^{O(1)}(mCU)\right)$ where* EO *is the cost of function evaluation.*

Proof. First, we can assume $l(e) \le u(e)$ for every edge e, otherwise, the problem is infeasible. Now, we apply a similar transformation in [49] to modify the graph. We create a new vertex v_0. For every vertex v in V, we create a edge from v_0 to v with capacity lower bounded by 0, upper bounded by $4nU$, and with cost $2mCU$. Edmonds and Giles showed that the submodular flow polytope is integral [29]. Hence, there is an integral optimal flow on this new graph. If the optimal flow passes through the newly created edge, then it has cost at least $2mCU - mCU$ because the cost of all other edges in total has at least $-mCU$. That means the optimal flow has the cost larger than mCU which is impossible. So the optimal flow does not use the newly created edges and vertex and hence the optimal flow in the new problem gives the optimal solution of the original problem. Next, we note that for any φ on the original graph such that $l(e) \le \varphi(e) \le u(e)$, we can send suitable amount of flow from v_0 to v to make φ feasible. Hence, this modification makes the feasibility problem trivial.

Lemma 51 shows that we can assume the new problem has an unique solution and it only blows up C by a $(mU)^{O(1)}$ factors.

Note that the optimal value is an integer and its absolute value at most mCU. By binary search, we can assume we know the optimal value OPT. Now, we reduce the problem to finding a feasible φ with $\{\langle d,\varphi\rangle \le \mathrm{OPT}+\epsilon\}$ with ϵ determined later. Let P_ϵ be the set of such φ. Note that $P_\epsilon = K_{1,\epsilon}\cap K_{2,\epsilon}$ where

$$\begin{aligned} K_{1,\epsilon} &= \left\{ x\in\mathbb{R}^V \text{ such that } \begin{array}{c} l(e)\le\varphi(e)\le u(e) \quad \forall e\in E \\ x(v) = (A\varphi)(v) \quad \forall v\in V \\ \langle d,\varphi\rangle \le \mathrm{OPT}+\epsilon \end{array} \text{ for some } \varphi \right\}, \\ K_{2,\epsilon} &= \left\{ y\in\mathbb{R}^V \text{ such that } \sum_{v\in S} y(v) \le f(S) \quad \forall S\subseteq V, \sum_{v\in V} y(v) = f(V) \right\}. \end{aligned}$$

Note that the extra condition $\sum_v y(v) = f(V)$ is valid because $\sum_v y(v) = \sum_v (A\varphi)(v) = 0$ and $f(V)=0$, and $K_{1,\epsilon}$ has radius bounded by $O((mCU)^{O(1)})$ and $K_{2,\epsilon}$ has radius bounded by $O(nU)$. Furthermore, for any vector $\vec{c}\in\mathbb{R}^V$, we note that

$$\begin{aligned} \max_{x\in K_{1,\epsilon}} \langle c,x\rangle &= \max_{l\le\varphi\le u, \langle d,\varphi\rangle\le \mathrm{OPT}+\epsilon, x=A\varphi} \langle c,x\rangle \\ &= \max_{l\le\varphi\le u, \langle d,\varphi\rangle\le \mathrm{OPT}+\epsilon} \langle c, A\varphi\rangle \\ &= \max_{l\le\varphi\le u, \langle d,\varphi\rangle\le \mathrm{OPT}+\epsilon} \left\langle A^T c,\varphi\right\rangle. \end{aligned}$$

To solve this problem, again we can do a binary search on $\langle d,\varphi\rangle$ and reduce the problem to

$$\max_{l\le\varphi\le u, \langle d,\varphi\rangle = K} \left\langle A^T c,\varphi\right\rangle$$

for some value of K. Since $A^T c$ is fixed, this is a linear program with only the box constraints and an extra equality constraint. Hence, it can be solved in nearly linear time [76, Thm 17, ArXiv v1]. As the optimization oracle for $K_{1,\epsilon}$ involves only computing $A^T c$ and solving this simple linear program, it takes only $O(n^2\log^{O(1)}(mCU/\epsilon))$ time. On the other hand, since $K_{2,\epsilon}$ is just a base

polyhedron, the optimization oracle for $K_{2,\epsilon}$ can be done by greedy method and only takes $O(n\mathrm{EO})$ time.

Applying Theorem 49, we can find q such that $\|q-x\|_2 + \|q-y\|_2 \le \delta$ for some $x \in K_{1,\epsilon}$, $y \in K_{2,\epsilon}$ and δ to be chosen later. According to the definition of $K_{1,\epsilon}$, there is φ such that $l(e) \le \varphi(e) \le u(e)$ and $x(v) = (A\varphi)(v)$ for all v and $\langle d, \varphi\rangle \le \mathrm{OPT} + \epsilon$. Since $\|y-x\|_2 \le 2\delta$, that means $|y(v) - (A\varphi)(v)| \le 2\delta$ for all v.

- Case 1) If $y(v) \ge (A\varphi)(v)$, then we can replace $y(v)$ by $(A\varphi)(v)$, note that y is still in $K_{2,\epsilon}$ because of the submodular constraints.
- Case 2) If $y(v) \le (A\varphi)(v)$, then we can send a suitable amount of flow from v_0 to v to make φ feasible $y(v) \le (A\varphi)(v)$.

Note that under this modification, we increased the objective value by $(\delta n)(2mCU)$ because the new edge cost $2mCU$ per unit of flow. Hence, we find a flow φ which is feasible in new graph with objective value $\epsilon + (\delta n)(2mCU)$ far from optimum value. By picking $\delta = \frac{1}{2mnCU}$, we have the value 2ϵ far from OPT. Now, we use Lemma 51 to shows that when ϵ is small enough, i.e, $\frac{1}{(mCU)^c}$ for some constant c, then we can guarantee that $\|y-x^*\|_\infty \le \frac{1}{4}$ where x^* is the optimal demand. Now, we note that $\|q-y\|_2 \le \delta$ and we note that we only modify y by a small amount, we in fact have $\|q-x^*\|_\infty < \frac{1}{2}$. Hence, we can read off the solution x^* by rounding q to the nearest integer. Note that we only need to solve the problem $K_{1,\epsilon} \cap K_{2,\epsilon}$ to $\frac{1}{(mCU)^{\Theta(1)}}$ accuracy and the optimization oracle for $K_{1,\epsilon}$ and $K_{2,\epsilon}$ takes time $O(n^2 \log^{O(1)}(mCU))$ and $O(n\mathrm{EO})$ respectively. Hence, Theorem 49 shows that it takes $O\left(n^2\mathrm{EO}\log(mCU) + n^3\log^{O(1)}(mCU)\right)$ time to find x^* exactly.

After getting x^*, one can find φ^* by solving a min cost flow problem using interior point method [74], which takes $O(m\sqrt{n}\log^{O(1)}(mCU))$ time. □

11.4 Affine Subspace of Convex Set

In this section, we give another example about using optimization oracle directly via regularization. We consider the following optimization problem

$$\max_{\vec{x}\in K \text{ and } \mathbf{A}\vec{x}=\vec{b}} \langle \vec{c}, \vec{x}\rangle \tag{11.14}$$

where $\vec{x}, \vec{c} \in \mathbb{R}^n$, K is a convex subset of $\mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{r\times n}$ and $\vec{b} \in \mathbb{R}^m$. We suppose that $r \ll n$ and thus, the goal of this subsection is to show how to obtain an algorithm takes only $\tilde{O}(r)$ many iterations. To do this, we assume a slightly stronger optimization oracle for K:

Definition 55. Given a convex set K and $\delta > 0$. A δ-2nd-order-optimization oracle for K is a function on $\mathbb{R}^n$ such that for any input $\vec{c} \in \mathbb{R}^n$ and $\lambda > 0$, it outputs $\vec{y}$ such that

$$\max_{\vec{x}\in K}\left(\langle \vec{c}, \vec{x}\rangle - \lambda\|\vec{x}\|^2\right) \le \delta + \langle \vec{c}, \vec{y}\rangle - \lambda\|\vec{y}\|^2.$$

We denote by $\mathrm{OO}^{(2)}_{\delta,\lambda}(K)$ the time complexity of this oracle.

The strategy for solving this problem is very similar to the intersection problem and hence some details are omitted.

Theorem 56. *Assume that* $\max_{\vec{x}\in K}\|\vec{x}\|_2 < M$, $\left\|\vec{b}\right\|_2 < M$, $\left\|\vec{c}\right\|_2 < M$, $\left\|\mathbf{A}\right\|_2 < M$ *and* $\lambda_{\min}(\mathbf{A}) > 1/M$. *Assume that* $K \cap \{\mathbf{A}\vec{x} = \vec{b}\} \neq \emptyset$ *and we have* ϵ*-2nd-order-optimization oracle for every* $\epsilon > 0$. *For* $0 < \delta < 1$, *we can find* $\vec{z} \in K$ *such that*

$$\max_{\vec{x}\in K \text{ and } \mathbf{A}\vec{x}=\vec{b}} \langle \vec{c}, \vec{x}\rangle \leq \delta + \langle \vec{c}, \vec{z}\rangle$$

and $\left\|\mathbf{A}\vec{z} - \vec{b}\right\|_2 \leq \delta$. *This algorithm takes time*

$$O\left(r\mathrm{OO}^{(2)}_{\eta,\lambda}(K)\log\left(\frac{nM}{\delta}\right) + r^3\log^{O(1)}\left(\frac{nM}{\delta}\right)\right)$$

where r *is the number of rows in* $\mathbf{A}$, $\eta = \left(\frac{\delta}{nM}\right)^{\Theta(1)}$ *and* $\lambda = \left(\frac{\delta}{nM}\right)^{\Theta(1)}$.

Proof. The proof is based on the minimax problem

$$\mathrm{OPT}_\lambda \stackrel{\text{def}}{=} \min_{\left\|\vec{\eta}\right\|_2\leq\lambda} \max_{\vec{x}\in K} \langle \vec{c}, \vec{x}\rangle + \left\langle \vec{\eta}, \mathbf{A}\vec{x} - \vec{b}\right\rangle - \frac{1}{\lambda}\left\|\vec{x}\right\|_2^2$$

where $\lambda = \left(\frac{\delta}{nM}\right)^c$ for some large constant c. We note that

$$\begin{aligned}\mathrm{OPT}_\lambda &= \max_{\vec{x}\in K} \min_{\left\|\vec{\eta}\right\|_2\leq\lambda} \langle \vec{c}, \vec{x}\rangle + \left\langle \vec{\eta}, \mathbf{A}\vec{x} - \vec{b}\right\rangle - \frac{1}{\lambda}\left\|\vec{x}\right\|_2^2 \\ &= \max_{\vec{x}\in K} \langle \vec{c}, \vec{x}\rangle - \lambda\left\|\mathbf{A}\vec{x} - \vec{b}\right\|_2 - \frac{1}{\lambda}\left\|\vec{x}\right\|_2^2.\end{aligned}$$

Since $\lambda_{\min}(\mathbf{A}) > 1/M$ and the set K is bounded by M, one can show that the saddle point $(\vec{x}^*, \vec{\eta}^*)$ of the minimax problem gives a good enough solution $\vec{x}$ for the original problem for large enough constant c.

For any $\vec{\eta}$, we define

$$\vec{x}_{\vec{\eta}} = \arg\max_{\vec{x}\in K} \langle \vec{c}, \vec{x}\rangle + \left\langle \vec{\eta}, \mathbf{A}\vec{x} - \vec{b}\right\rangle - \frac{1}{\lambda}\left\|\vec{x}\right\|_2^2.$$

Since the problem is strongly concave in $\vec{x}$, one can prove that

$$\left\|\vec{x}_{\vec{\eta}} - \vec{x}^*\right\|_2 \leq \left(\frac{nM}{\delta}\right)^{O(c)} \left\|\vec{\eta} - \vec{\eta}^*\right\|_2.$$

Hence, we can first find an approximate minimizer of the function $f(\vec{\eta}) = \max_{\vec{x}\in K} \langle \vec{c}, \vec{x}\rangle + \left\langle \vec{\eta}, \mathbf{A}\vec{x} - \vec{b}\right\rangle - \frac{1}{\lambda}\left\|\vec{x}\right\|_2^2$ and use the oracle to find $\vec{x}_{\vec{\eta}}$.

To find an approximate minimizer of f, we note that the subgradient of f can be found using the optimization oracle similar to Theorem 49. Hence, the result follows from our cutting plane method and the fact that $\vec{\eta} \in \mathbb{R}^r$. □

Remark 57. In [74], they considered the special case $K = \{\vec{x} : 0 \leq x_i \leq 1\}$ and showed that it can be solved in $\tilde{O}(\sqrt{r})$ iterations using interior point methods. This gives the current fastest algorithm for the maximum flow problem on directed weighted graphs. Our result generalizes their result to any convex set K but with $\tilde{O}(r)$ iterations. This suggests the following open problem: under what condition on K can one optimize linear functions over affine subspaces of K with r constraints in $\tilde{O}(\sqrt{r})$ iterations?

Part III
Submodular Function Minimization

12 Introduction

Submodular functions and submodular function minimization (SFM) are fundamental to the field of combinatorial optimization. Examples of submodular functions include graph cut functions, set coverage function, and utility functions from economics. Since the seminal work by Edmonds in 1970 [27], submodular functions and the problem of minimizing such functions (i.e. submodular function minimization) have served as a popular modeling and optimization tool in various fields such as theoretical computer science, operations research, game theory, and most recently, machine learning. Given its prevalence, fast algorithms for SFM are of immense interest both in theory and in practice.

Throughout Part III, we consider the standard formulation of SFM: we are given a submodular function f defined over the subsets of a n-element ground set. The values of f are integers, have absolute value at most M, and are evaluated by querying an oracle that takes time EO. Our goal is to produce an algorithm that solves this SFM problem, i.e. finds a minimizer of f, while minimizing both the number of oracle calls made and the total running time.

We provide new $O(n^2 \log nM \cdot \text{EO} + n^3 \log^{O(1)} nM)$ and $O(n^3 \log^2 n \cdot \text{EO} + n^4 \log^{O(1)} n)$ time algorithms for SFM. These algorithms improve upon the previous fastest weakly and strongly polynomial time algorithms for SFM which had a a running time of $O((n^4 \cdot \text{EO} + n^5) \log M)$ [54] and $O(n^5 \cdot \text{EO} + n^6)$ [90] respectively. Consequently, we improve the running times in both regimes by roughly a factor of $O(n^2)$.

Both of our algorithms bear resemblance to the classic approach of Grötschel, Lovász and Schrijver [49, 50] using the Lovász extension. In fact our weakly polynomial time algorithm directly uses the Lovász extension as well as the results of Part II to achieve these results. Our strongly polynomial time algorithm also uses the Lovász extension, along with more modern tools from the past 15 years.

At a high level, our strongly polynomial algorithms apply our cutting plane method in conjunction with techniques originally developed by Iwata, Fleischer, and Fujishige (IFF) [56]. Our cutting plane method is performed for enough iterations to sandwich the feasible region in a narrow strip from which useful structural information about the minimizers can be deduced. Our ability to derive the new information hinges on a significant extension of IFF techniques.

Over the past few decades, SFM has drawn considerable attention from various research communities, most recently in machine learning [11, 68]. Given this abundant interest in SFM, we hope that our ideas will be of value in various practical applications. Indeed, one of the critiques against existing theoretical algorithms is that their running time is too slow to be practical. Our contribution, on the contrary, shows that this school of algorithms can actually be made fast theoretically and we hope it may potentially be competitive against heuristics which are more commonly used.

12.1 Previous Work

Here we provide a brief survey of the history of algorithms for SFM. For a more comprehensive account of the rich history of SFM, we refer the readers to recent surveys [81, 55].

The first weakly and strongly polynomial time algorithms for SFM were based on the ellipsoid method [65] and were established in the foundational work of Grötschel, Lovász and Schrijver in 1980's [49, 50]. Their work was complemented by a landmark paper by Cunningham in 1985 which provided a pseudopolynomial algorithm that followed a flow-style algorithmic framework [20]. His tools foreshadowed much of the development in SFM that would take place 15 years later. Indeed, modern algorithms synthesize his framework with inspirations from various max flow algorithms.

The first such "flow style" strongly polynomial algorithms for SFM were discovered independently in the breakthrough papers by Schrijver [93] and Iwata, Fleischer, and Fujishige (IFF) [56]. Schrijver's algorithm has a running of $O(n^8 \cdot \text{EO} + n^9)$ and borrows ideas from the push-relabel algorithms [46, 25] for the maximum flow problem. On the other hand, IFF's algorithm runs in time $O(n^7 \log n \cdot \text{EO})$ and $O(n^5 \cdot \text{EO} \log M)$, and applies a flow-scaling scheme with the aid of certain proximity-type lemmas as in the work of Tardos [100]. Their method has roots in flow algorithms such as [52, 47].

Subsequent work on SFM provided algorithms with considerably faster running time by extending the ideas in these two "genesis" papers [93, 56] in various novel directions [107, 31, 54, 90, 60]. Currently, the fastest weakly and strongly polynomial time algorithms for SFM have a running time of $O((n^4 \cdot \text{EO} + n^5) \log M)$ [54] and $O(n^5 \cdot \text{EO} + n^6)$ [90] respectively. Despite this impressive track record, the running time has not been improved in the last eight years.

We remark that all of the previous algorithms for SFM proceed by maintaining a convex combination of $O(n)$ BFS's of the base polyhedron, and incrementally improving it in a relatively local manner. As we shall discuss in Section 12.2, our algorithms do not explicitly maintain a convex combination. This may be one of the fundamental reasons why our algorithms achieve a faster running time.

Finally, beyond the distinction between weakly and strongly polynomial time algorithms for SFM, there has been interest in another type of SFM algorithm, known as fully combinatorial algorithms in which only additions and subtractions are permitted. Previous such algorithms include [60, 54, 53]. We do not consider such algorithms in the remainder of the paper and leave it as an open question if it is possible to turn our algorithms into fully combinatorial ones.

12.2 Our Results and Techniques

In Part III we show how to improve upon the previous best known running times for SFM by a factor of $O(n^2)$ in both the strongly and weakly polynomial regimes. In Table 11 summarizes the running time of the previous algorithms as well as the running times of the fastest algorithms presented in this paper.

Both our weakly and strongly polynomial algorithms for SFM utilize a convex relaxation of the submodular function, called the *Lovász extension*. Our algorithms apply our cutting plane method from Part I using a separation oracle given by the subgradient of the Lovász extension. To the best of the author's knowledge, Grötschel, Lovász and Schrijver were the first to formulate this convex optimization framework for SFM [49, 50].

For weakly polynomial algorithms, our contribution is two-fold. First, we show that cutting plane methods such as Vaidya's [105] can be applied to SFM to yield faster algorithms. Second, as our cutting plane method, Theorem 42, improves upon previous cutting plane algorithms and consequently the running time for SFM as well. This gives a running time of $O(n^2 \log nM \cdot \text{EO} +$

Authors	Years	Running times	Remarks
Grötschel, Lovász, Schrijver [49, 50]	1981,1988	$\widetilde{O}(n^5 \cdot \mathrm{EO} + n^7)$[81]	first weakly and strongly
Cunningham [20]	1985	$O(Mn^6 \log nM \cdot \mathrm{EO})$	first pseudopoly
Schrijver [93]	2000	$O(n^8 \cdot \mathrm{EO} + n^9)$	first combin. strongly
Iwata, Fleischer, Fujishige[56]	2000	$O(n^5 \cdot \mathrm{EO} \log M)$ $O(n^7 \log n \cdot \mathrm{EO})$	first combin. strongly
Iwata, Fleischer [31]	2000	$O(n^7 \cdot \mathrm{EO} + n^8)$	
Iwata [54]	2003	$O((n^4 \cdot \mathrm{EO} + n^5) \log M)$ $O((n^6 \cdot \mathrm{EO} + n^7) \log n)$	current best weakly
Vygen [107]	2003	$O(n^7 \cdot \mathrm{EO} + n^8)$	
Orlin [90]	2007	$O(n^5 \cdot \mathrm{EO} + n^6)$	current best strongly
Iwata, Orlin [60]	2009	$O((n^4 \cdot \mathrm{EO} + n^5) \log nM)$ $O((n^5 \cdot \mathrm{EO} + n^6) \log n)$	
Our algorithms	2015	$O(n^2 \log nM \cdot \mathrm{EO} + n^3 \log^{O(1)} nM)$ $O(n^3 \log^2 n \cdot \mathrm{EO} + n^4 \log^{O(1)} n)$	

Table 11: Algorithms for submodular function minimization. Note that some of these algorithms were published in both conferences and journals, in which case the year we provided is the earlier one.

$n^3 \log^{O(1)} nM$), an improvement over the previous best algorithm by Iwata [54] by a factor of almost $O(n^2)$.

Our strongly polynomial algorithms, on the other hand, require substantially more innovation. We first begin with a very simple geometric argument that SFM can be solved in $O(n^3 \log n \cdot \mathrm{EO})$ oracle calls (but in exponential time). This proof only uses Grunbaum's Theorem from convex geometry and is completely independent from the rest of the paper. It was the starting point of our method and suggests that a running time of $\widetilde{O}(n^3 \cdot \mathrm{EO} + n^{O(1)})$ for submodular minimization is in principle achievable.

To make this existence result algorithmic, we first run cutting plane, Theorem 31, for enough iterations such that we compute either a minimizer or a set P containing the minimizers that fits within in a narrow strip. This narrow strip consists of the intersection of two approximately parallel hyperplanes. If our narrow strip separates P from one of the faces $x_i = 0$, $x_i = 1$, we can effectively eliminate the element i from our consideration and reduce the dimension of our problem by 1. Otherwise a pair of elements p, q can be identified for which q is guaranteed to be in any minimizer containing p (but p may not be contained in a minimizer). Our first algorithm deduces only one such pair at a time. This technique immediately suffices to achieve a $\widetilde{O}(n^4 \cdot \mathrm{EO} + n^5)$ time algorithm for SFM (See Section 15.3). We then improve the running time to $\widetilde{O}(n^3 \cdot \mathrm{EO} + n^4)$ by showing how to deduce many such pairs simultaneously. Similar to past algorithms, this structural information is deduced from a point in the so-called base polyhedron (See Section 13).

Readers well-versed in SFM literature may recognize that our strongly polynomial algorithms are reminiscent of the scaling-based approach first used by IFF [56] and later in [54, 60]. While both approaches share the same skeleton, there are differences as to how structural information about minimizers is deduced. A comparison of our algorithms and previous ones are presented in Section 16.

Finally, there is one more crucial difference between these algorithms which we believe is responsible for much of our speedup. One common feature shared by all the previous algorithms is

that they maintain a convex combination of $O(n)$ BFS's of the base polyhedron, and incrementally improve on it by introducing new BFS's by making local changes to existing ones. Our algorithms, on the other hand, choose new BFS's by the cutting plane method. Because of this, our algorithm considers the geometry of the existing BFS's where each of them has influences over the choice of the next BFS. In some sense, our next BFS is chosen in a more "global" manner.

12.3 Organization

The rest of Part III is organized as follows. We first begin with a gentle introduction to submodular functions in Section 13. In Section 14, we apply our cutting plane method to SFM to obtain a faster weakly polynomial algorithms. In Section 15 we then present our results for achieving better strongly polynomial algorithms, where a warm-up $\widetilde{O}(n^4 \cdot \text{EO} + n^5)$ algorithm is given before the full-fledged $\widetilde{O}(n^3 \cdot \text{EO} + n^4)$ algorithm. Finally, we end the part with a discussion and comparison between our algorithms and previous ones in Section 16.

We note that there are a few results in Part III that can be read fairly independently of the rest of the paper. In Theorem *67* we show how Vaidya's algorithm can be applied to SFM to obtain a faster weakly polynomial running time. Also in Theorem 71 we present a simple geometric argument that SFM can be solved with $O(n^3 \log n \cdot \text{EO})$ oracle calls but with exponential time. These results can be read with only a working knowledge of the Lovász extension of submodular functions.

13 Preliminaries

Here we introduce background on submodular function minimization (SFM) and notation that we use throughout Part III. Our exposition is kept to a minimal amount sufficient for our purposes. We refer interested readers to the extensive survey by McCormick [81] for further intuition.

13.1 Submodular Function Minimization

Throughout the rest of the paper, let $V = \{1, ..., n\} = [n]$ denote a ground set and let $f : 2^V \longrightarrow \mathbb{Z}$ denote a *submodular function* defined on subsets of this ground set. We use V and $[n]$ interchangeably and let $[0] \stackrel{\text{def}}{=} \emptyset$. We abuse notation by letting $S + i \stackrel{\text{def}}{=} S \cup \{i\}$ and $S - i \stackrel{\text{def}}{=} S \backslash \{i\}$ for an element $i \in V$ and a set $S \subseteq 2^V$. Formally, we call a function submodular if it obeys the following property of *diminishing marginal differences*:

Definition 58 (Submodularity)**.** A function $f : 2^V \longrightarrow \mathbb{Z}$ is *submodular* if $f(T + i) - f(T) \leq f(S + i) - f(S)$ for any $S \subseteq T$ and $i \in V \backslash T$.

For convenience we assume without loss of generality that $f(\emptyset) = 0$ by replacing $f(S)$ by $f(S) - f(\emptyset)$ for all S. We also let $M \stackrel{\text{def}}{=} \max_{S \in 2^V} |f(S)|$.

The central goal of Part III is to design algorithms for SFM, i.e. computing the minimizer of f. We call such an algorithm *strongly polynomial* if its running time depends only polynomially on n and EO, the time needed to compute $f(S)$ for a set S, and we call such an algorithm *weakly polynomial* if it also depends polylogarithmically on M.

13.2 Lovász Extension

Our new algorithms for SFM all consider a convex relaxation of a submodular function, known as the Lovász extension, and then carefully apply our cutting plane methods to it. Here we formally introduce the Lovász extension and present basic facts that we use throughout Part III.

The Lovász extension of $\hat{f} : [0,1]^n \longrightarrow \mathbb{R}$ of our submodular function f is defined for all $\vec{x}$ by

$$\hat{f}(\vec{x}) \stackrel{\text{def}}{=} \mathbb{E}_{t\sim[0,1]}[f(\{i : x_i \geq t\})],$$

where $t \sim [0,1]$ is drawn uniformly at random from $[0,1]$. The Lovász extension allows us to reduce SFM to minimizing a convex function defined over the interior of the hypercube. Below we state that the Lovász extension is a convex relaxation of f and that it can be evaluated efficiently.

Theorem 59. *The Lovász extension $\hat{f}$ satisfies the following properties:*

1. *$\hat{f}$ is convex and $\min_{\vec{x}\in[0,1]^n} \hat{f}(\vec{x}) = \min_{S\subset[n]} f(S)$;*
2. *$f(S) = \hat{f}(I_S)$, where I_S is the characteristic vector for S, i.e. $I_S(i) = \begin{cases} 1 & \text{if } i \in S \\ 0 & \text{if } i \notin S \end{cases}$;*
3. *If S is a minimizer of f, then I_S is a minimizer of $\hat{f}$;*
4. *Suppose $x_1 \geq \cdots \geq x_n \geq x_{n+1} \stackrel{\text{def}}{=} 0$, then*

$$\hat{f}(\vec{x}) = \sum_{i=1}^{n} f([i])(x_i - x_{i+1}) = \sum_{i=1}^{n} (f([i]) - f([i-1]))x_i\,.$$

Proof. See [50] or any standard textbook on combinatorial optimization, e.g. [94]. □

Next we show that we can efficiently compute a subgradient of the Lovász or alternatively, a separating hyperplane for the set of minimizers of our submdoular function f. First we remind the reader of the definition of a separation oracle, and then we prove the necessary properties of the hyperplane, Theorem 61.

Definition 60 (separation oracle, Defintion 1 restated for Lovász extension)**.** Given a point $\bar{x}$ and a convex function $\hat{f}$ over a convex set P, $\vec{a}^T\vec{x} \leq b$ is a separating hyperplane if $\vec{a}^T\bar{x} \geq b$ and any minimizer x^* of $\hat{f}$ over P satisfies $\vec{a}^T x^* \leq b$.

Theorem 61. *Given a point $\bar{x} \in [0,1]^n$ assume without loss of generality (by re-indexing the coordinates) that $\bar{x}_1 \geq \cdots \geq \bar{x}_n$. Then the following inequality is a valid separating hyperplane for $\vec{x}$ and f:*

$$\sum_{i=1}^{n} (f([i]) - f([i-1]))x_i \leq \hat{f}(\bar{x})$$

i.e., it satisfies the following:

1. *(separating) $\bar{x}$ lies on $\sum_{i=1}^{n}(f([i]) - f([i-1]))x_i \leq \hat{f}(\bar{x})$.*
2. *(valid) For any $\vec{x}$, we have $\sum_{i=1}^{n}(f([i]) - f([i-1]))x_i \leq \hat{f}(\vec{x})$. In particular, $\sum_{i=1}^{n}(f([i]) - f([i-1]))x_i^* \leq \hat{f}(\bar{x})$ for any minimizer $\vec{x}^*$, i.e. the separating hyperplane does not cut out any minimizer.*

Moreover, such a hyperplane can be computed with n oracle calls to f and in time $O(n \cdot EO + n^2)$.

Proof. Note that by Theorem 59 we have that $\sum_{i\in[n]}(f([i])-f([i-1]))x_i = \hat{f}(\bar{x})$ and thus the hyperplane satisfies the separating condition. Moreover, clearly computing it only takes time $O(n\cdot \mathrm{EO}+n^2)$ as we simply need to sort the coordinates and evaluate f at n points, i.e. each of the $[i]$. All that remains is to show that the hyperplane satisfies the valid condition.

Let $L^{(t)} \stackrel{\text{def}}{=} \{i : x_i \geq t\}$. Recall that $\hat{f}(\vec{x}) = \mathbb{E}_{t\sim[0,1]}[f(L_t)]$. Thus $\hat{f}(\vec{x})$ can be written as a convex combination $\hat{f}(\vec{x}) = \sum_t \alpha_t f(L^{(t)})$, where $\alpha_t \geq 0$ and $\sum_t \alpha_t = 1$. However, by diminishing marginal differences we see that for all t

$$\begin{aligned}\sum_{i\in[n]}(f([i])-f([i-1]))\left(I_{L^{(t)}}\right)_i &= \sum_{i\in L^{(t)}}(f([i])-f([i-1])) \\ &\leq \sum_{i\in L^{(t)}}\left(f([i]\cap L^{(t)})-f([i-1]\cap L^{(t)})\right) \\ &= f(L^{(t)})-f(\emptyset)=f(L^{(t)})\end{aligned}$$

and therefore since $\sum_t \alpha_t I_{L^{(t)}} = \vec{x}$ we have

$$\sum_{i\in[n]}(f[i]-f([i-1])x_i = \sum_t \alpha_t \sum_{i=1}^{n}(f([i])-f([i-1]))\left(I_{L^{(t)}}\right)_i \leq \sum_t \alpha_t f(L^{(t)}) = \hat{f}(\vec{x}).$$

□

13.3 Polyhedral Aspects of SFM

Here we provide a natural primal dual view of SFM that we use throughout the analysis. We provide a dual convex optimization program to minimizing the Lovász extension and provide several properties of these programs. We believe the material in this section helps crystallize some of the intuition behind our algorithm and we make heavy use of the notation presented in this section. However, we will not need to appeal to the strong duality of these programs in our proofs.

Consider the following primal and dual programs, where we use the shorthands $y(S) = \sum_{i\in S} y_i$ and $y_i^- = \min\{0, y_i\}$. Here the primal constraints are often called the *base polyhedron* $\mathcal{B}(f) \stackrel{\text{def}}{=} \{\vec{y} \in \mathbb{R}^n : y(S) \leq f(S) \forall S \not\subseteq V, y(V) = f(V)\}$ *and the dual program directly corresponds to minimizing the Lovász extension and thus* f.

Primal	Dual
$\max y^-(V)$ $y(S) \leq f(S) \forall S \not\subseteq V$ $y(V) = f(V)$	$\min \hat{f}(\vec{x})$ $0 \leq \vec{x} \leq 1$

Theorem 62. *$\vec{h}$ is a basic feasible solution (BFS) of the base polyhedron $\mathcal{B}(f)$ if and only if*

$$h_i = f(\{v_1, ..., v_i\}) - f(\{v_1, ..., v_{i-1}\})$$

for some permutation $v_1, ..., v_n$ of the ground set V. We call $v_1, ..., v_n$ the defining permutation of $\vec{h}$. We call v_i precedes v_j for $i < j$.

This theorem gives a nice characterization of the BFS's of $\mathcal{B}(f)$. It also gives the key observation underpinning our approach: **the coefficients of each separating hyperplane in Theorem 61 precisely corresponds to a primal BFS (Theorem 62)**. Our analysis relies heavily on this connection. We re-state Theorem 61 in the language of BFS.

Lemma 63. *We have $\vec{h}^T\vec{x} \leq \hat{f}(\vec{x})$ for any $\vec{x} \in [0,1]^n$ and BFS $\vec{h}$.*

Proof. Any BFS is given by some permutation. Thus this is just Theorem 61 in disguise. □

We also note that since the objective function of the primal program is non-linear, we cannot say that the optimal solution to the primal program is a BFS. Instead we only know that it is a convex combination of the BFS's that satisfy the following property. A proof can be found in any standard textbook on combinatorial optimization.

Theorem 64. *The above primal and dual programs have no duality gap. Moreover, there always exists a primal optimal solution $\vec{y} = \sum_k \lambda^{(k)}\vec{h}^{(k)}$ with $\sum_k \lambda^{(k)} = 1$ (a convex combination of BFS $\vec{h}^{(k)}$) s.t. any i with $y_i < 0$ precedes any j with $y_j > 0$ in the defining permutation for each BFS $\vec{h}^{(k)}$.*

Our algorithms will maintain collections of BFS and use properties of $\vec{h} \in \mathcal{B}(f)$, i.e. convex combination of BFS. To simplify our analysis at several points we will want to assume that such a vector $\vec{h} \in \mathcal{B}(f)$ is *non-degenerate*, meaning it has both positive and negative entries. Below, we prove that such degenerate points in the base polytope immediately allow us to trivially solve the SFM problem.

Lemma 65 (Degenerate Hyperplanes)**.** *If $\vec{h} \in \mathcal{B}(f)$ is non-negative then $\emptyset$ is a minimizer of f and if $\vec{h}$ is non-positive then V is a minimizer of f.*

Proof. While this follows immediately from Theorem 64, for completeness we prove this directly. Let $S \in 2^V$ be arbitrary. If $\vec{h} \in \mathcal{B}(f)$ is non-negative then by the we have

$$f(S) \geq \vec{h}(S) = \sum_{i \in S} h_i \geq 0 = f(\emptyset)\,.$$

On the other hand if $\vec{h}$ is non-positive then by definition we have

$$f(S) \geq \vec{h}(S) = \sum_{i \in S} h_i \geq \sum_{i \in V} h_i = h(V) = f(V)\,.$$

□

14 Improved Weakly Polynomial Algorithms for SFM

In this section we show how our cutting plane method can be used to obtain a $O(n^2 \log nM \cdot \mathrm{EO} + n^3 \log^{O(1)} nM)$ time algorithm for SFM. Our main result in this section is the following theorem, which shows how directly applying our results from earlier parts to minimize the Lovász extension yields the desired running time.

Theorem 66. *We have an $O(n^2 \log nM \cdot EO + n^3 \log^{O(1)} nM)$ time algorithm for submodular function minimization.*

Proof. We apply Theorem 42 to the Lovász extension $\hat{f} : [0,1]^n \longrightarrow \mathbb{R}$ with the separation oracle given by Theorem 61. $\hat{f}$ fulfills the requirement on the domain as its domain $\Omega = [0,1]^n$ is symmetric about the point $(1/2, \ldots, 1/2)$ and has exactly $2n$ constraints.

In the language of Theorem 42, our separation oracle is a $(0,0)$-separation oracle with $\eta = 0$ and $\delta = 0$.

We first show that $\delta = 0$. Firstly, our separating hyperplane can be written as

$$\sum_{i=1}^{n}(f([i]) - f([i-1]))x_i \leq \hat{f}(\bar{x}) = \sum_{i=1}^{n}(f([i]) - f([i-1]))\bar{x}_i,$$

where the equality follows from Theorem 59. Secondly, for any $\vec{x}$ with $\hat{f}(\vec{x}) \leq \hat{f}(\bar{x})$ we have by Theorem 61 that

$$\sum_{i=1}^{n}(f([i]) - f([i-1]))x_i \leq \hat{f}(\vec{x}) \leq \hat{f}(\bar{x})$$

which implies that $\vec{x}$ is not cut away by the hyperplane.

Next we show that $\eta = 0$. Our separating hyperplane induces a valid halfspace whenever it is not nonzero, i.e. $f([i]) \neq f([i-1])$ for some i. In the case that it is zero $f([i]) = f([i-1])\forall i$, by the same argument above, we have $\hat{f}(\bar{x}) = \sum_{i=1}^{n}(f([i]) - f([i-1]))\bar{x}_i = 0$ and

$$\hat{f}(\vec{x}) \geq \sum_{i=1}^{n}(f([i]) - f([i-1]))x_i = 0 = \hat{f}(\bar{x}).$$

In other words, $\bar{x}$ is an exact minimizer, i.e. $\eta = 0$.

Note that $\left|\hat{f}(\vec{x})\right| = \left|\mathbb{E}_{t\sim[0,1]}[f(\{i : x_i \geq t\})]\right| \leq M$ as $M = \max_S |f(S)|$. Now plugging in $\alpha = \frac{1}{4M}$ in the guarantee of Theorem 31, we can find a point x^* such that

$$\begin{aligned}
\hat{f}(x^*) - \min_{\vec{x}\in[0,1]^n} \hat{f}(\vec{x}) &\leq \frac{1}{4M}\left(\max_{\vec{x}\in[0,1]^n} \hat{f}(\vec{x}) - \min_{\vec{x}\in[0,1]^n} \hat{f}(\vec{x})\right) \\
&\leq \frac{1}{4M}(2M) \\
&< 1
\end{aligned}$$

We claim that $\min_{t\in[0,1]} f(\{i : x_i^* \geq t\})$ is minimum. To see this, recall from 59 that $\hat{f}$ has an integer minimizer and hence $\min_{\vec{x}\in[0,1]^n} \hat{f}(\vec{x}) = \min_S f(S)$. Moreover, $\hat{f}(x^*)$ is a convex combination of $f(\{i : x_i^* \geq t\})$ which gives

$$1 > \hat{f}(x^*) - \min_{\vec{x}\in[0,1]^n} \hat{f}(\vec{x}) = \hat{f}(x^*) - \min_S f(S) \geq \min_{t\in[0,1]} f(\{i : x_i^* \geq t\}) - \min_S f(S).$$

Since f is integer-valued, we must then have $\min_{t\in[0,1]} f(\{i : x_i^* \geq t\}) = \min_S f(S)$ as desired. Since our separation oracle can be computed by n oracle calls and runs in time $O(n \cdot \mathrm{EO} + n^2)$, by Theorem 42 the overall running time is then $O(n^2 \log nM \cdot \mathrm{EO} + n^3 \log^{O(1)} nM)$ as claimed. □

Needless to say the proof above completely depends on Theorem 42. We remark that one can use the Vaidya's cutting plane instead of ours to get a time complexity $O(n^2 \log nM \cdot \mathrm{EO} + n^{\omega+1} \log^{O(1)} n \cdot \log M)$. There is actually an alternate argument that gives a time complexity of $O(n^2 \log M \cdot \mathrm{EO} + n^{O(1)} \cdot \log M)$. Thus it requires slightly fewer oracle calls at the expense of slower running time. A proof is offered in this section, which can be skipped without any risk of discontinuation. This proof relies the following cutting plane method.

Theorem 67 ([13])**.** *Given any convex set $K \subset [0,1]^n$ with a separation oracle of cost SO, in time $O(kSO + kn^{O(1)})$ one can find either find a point $\vec{x} \in K$ or find a polytope P such that $K \subset P$ and the volume of K is at most $\left(\frac{2}{3}\right)^k$.*

The Theorem allows us to decrease the volume of the feasible region by a factor of $\left(\frac{2}{3}\right)^k$ after k iterations. Similar to above, we apply cutting plane to minimize $\hat{f}$ over the hypercube $[0,1]^n$ for $O(n \log M)$ iterations, and outputs *any* integral point in the remaining feasible region P.

Lemma 68. *Let x^* achieve the minimum function value $\hat{f}(x^*)$ among the points used to query the separation oracle. Then*

1. *$x^* \in P^{(k)}$, the current feasible region.*
2. *Any $\vec{x}$ with $\hat{f}(\vec{x}) \leq \hat{f}(x^*)$ belongs to $P^{(k)}$.*
3. *suppose $x^*_{i_1} \geq \cdots \geq x^*_{i_n}$ and let $S_j = \{i_1, \ldots, i_j\}$. Then $S_l \in \arg\min_{S_j} f(S_j)$ also belongs to $P^{(k)}$.*

Proof. For any separating hyperplane $\vec{h}^T x \leq \hat{f}(\bar{x})$ given by $\bar{x}$, we have by Lemma 63 that $\vec{h}^T x^* \leq \hat{f}(x^*)$. Since $\hat{f}(x^*)$ is the minimum among all $\hat{f}(\bar{x})$, $\vec{h}^T x^* \leq \hat{f}(\bar{x})$ and hence x^* is not removed by any new separating hyperplane. In other words, $x^* \in P^{(k)}$. The argument for (2) is analogous.

For (3), recall that by the definition of Lovász extension $\hat{f}(x^*)$ is a convex combination of $f(S_j)$ and thus the indicator variable I_{S_l} for S_l satisfies $f(I_{S_l}) \leq \hat{f}(x^*)$. By Lemma 63 again, this implies $\vec{h}^T I_{S_l} \leq f(I_{S_l}) \leq \hat{f}(x^*) \leq \hat{f}(\bar{x})$ for any separating hyperplane $\vec{h}^T x \leq \hat{f}(\bar{x})$. □

Theorem 69. *Suppose that we run Cutting Plane in Theorem 67 for $O(n \log M)$ iterations. Then S_l from the last lemma also minimizes f.*

Proof. We use the notations from the last lemma. After $k = Kn \log_{2/3} M$ iterations, the volume of the feasible region $P^{(k)}$ is at most $1/M^{Kn}$. By the last lemma, $I_{S_l} \in P^{(k)}$.

Suppose for the sake of contradiction that S minimizes f but $f(S) < f(S_l)$. Since f is integer-valued, $f(S) + 1 \leq f(S_l)$. Let $r \stackrel{\text{def}}{=} 1/6M$. Consider the set $B \stackrel{\text{def}}{=} \{\vec{x} : 0 \leq x_i \leq r \; \forall \text{i} \notin S, \, 1 - r \leq x_i \leq 1 \; \forall i \in S\}$. We claim that for $\vec{x} \in B$,

$$\hat{f}(\vec{x}) \leq f(S) + 1.$$

To show this, note that $f(\{i : x_i \geq t\}) = f(S)$ for $r < t \leq 1 - r$ as $x_i \leq r$ for $i \notin S$ and $x_i \geq 1 - r$ for $i \in S$. Now using conditional probability and $|f(T)| \leq M$ for any T,

$$\begin{aligned}
\hat{f}(\vec{x}) &= \mathbb{E}_{t \sim [0,1]}[f(\{i : x_i \geq t\})] \\
&= (1 - 2r)\, \mathbb{E}[f(\{i : x_i \geq t\}) | r < t \leq 1 - r] + \\
&\quad r\left(\mathbb{E}[f(\{i : x_i \geq t\}) | 0 \leq t \leq r] + \mathbb{E}[f(\{i : x_i \geq t\}) | 1 - r \leq t \leq 1]\right) \\
&= (1 - r)\, f(S) + r\left(\mathbb{E}[f(\{i : x_i \geq t\}) | 0 \leq t \leq r + \mathbb{E}[f(\{i : x_i \geq t\}) | 1 - r \leq t \leq 1]\right) \\
&\leq (1 - 2r)\, f(S) + 2rM \\
&\leq f(S) + 4rM \\
&\leq f(S) + 1
\end{aligned}$$

But now $B \subseteq P^{(k)}$ as $\hat{f}(\vec{x}) \leq f(S) + 1 \leq f(S_l)$ and by (2) of the last lemma. This would lead to a contradiction since

$$\text{vol}(B) = \frac{1}{(6M)^n} > \frac{1}{M^{Kn}} \geq \text{vol}(P^{(k)})$$

for sufficiently large K. □

Corollary 70. *There is an $O(n^2 \log M \cdot EO + n^{O(1)} \log M)$ time algorithm for submodular function minimization.*

Proof. This simply follows from the last lemma, Theorem 67, and the fact that our separation oracle runs in time $O(n \cdot \mathrm{EO} + n^2)$. □

Curiously, we obtained $O(\log M)$ rather than $O(\log nM)$ as in our algorithm. We leave it as an open problem whether one can slightly improve our running time to $O(n^2 \log M \cdot \mathrm{EO} + n^3 \log^{O(1)} n \cdot \log M)$. The rest of this paper is devoted to obtaining better strongly polynomial running time.

15 Improved Strongly Polynomial Algorithms for SFM

In this section we show how our cutting plane method can be used to obtain a $\widetilde{O}(n^3 \cdot \mathrm{EO} + n^4)$ time algorithm for SFM, which improves over the currently fastest $O(n^5 \cdot \mathrm{EO} + n^6)$ time algorithm by Orlin.

15.1 Improved Oracle Complexity

We first present a simple geometric argument that f can be minimized with just $O(n^3 \log n \cdot \mathrm{EO})$ oracle calls. While this is our desired query complexity (and it improves upon the previous best known bounds by a factor of $O(n^2)$ unfortunately the algorithm runs in exponential time. Nevertheless, it does provide some insight into how our more efficient algorithms should proceed and it alone, does suggests that information theoretically, $O(n^3 \log n \cdot \mathrm{EO})$ calls suffice to solve SFM. In the rest of the paper, we combine this insight with some of the existing SFM tools developed over the last decade to get improved polynomial time algorithms.

Theorem 71. *Submodular functions can be minimized with $O(n^3 \log n \cdot EO)$ oracle calls.*

Proof. We use the cutting plane method in Theorem 67 with the separation oracle given by Theorem 61. This method reduce the volume of the feasible region by a factor of $(\frac{2}{3})^k$ after k iterations if the optimal has not found yet.

Now, we argue that after $O(n \log n)$ iterations of this procedure we have either found a minimizer of f or we have enough information to reduce the dimension of the problem by 1. To see this, first note that if the separation oracle ever returns a degenerate hyperplane, then by Lemma 65 then either $\emptyset$ or V is the minimizer, which we can determine in time $O(\mathrm{EO} + n)$. Otherwise, after $100n \log n$ iterations, our feasible region P must have a volume of at most $1/n^{10n}$. In this case, we claim that the remaining integer points in P all lie on a hyperplane. This holds, as if this was not the case, then there is a simplex $\triangle$, with integral vertices $v_0, v_1, \ldots, v_n$, contained in P. But then

$$\mathrm{vol}(P) \geq \mathrm{vol}(\triangle) = \frac{1}{n!} \left|\det\left(v_1 - v_0 \; v_2 - v_0 \; \ldots \; v_n - v_0\right)\right| \geq \frac{1}{n!}$$

where the last inequality holds since the determinant of an integral matrix is integral, yielding a contradiction.

In other words after $O(n \log n)$ iterations, we have reduced the dimension of all viable solutions by at least 1. Thus, we can recurse by applying the cutting plane method to the lower dimensional feasible region, i.e. P is (replaced by) the convex combination of all the remaining integer points. There is a minor technical issue we need to address as our space is now lower dimensional and the starting region is not necessarily the hypercube anymore and the starting volume is not necessarily equal to 1.

We argue that the starting volume is bounded by $n^{O(n)}$. If this is indeed the case, then our previous argument still works as the volume goes down by a factor of $1/n^{O(n)}$ in $O(n \log n)$ iterations.

Let $v \in P$ be an integer point. Now the $\dim(P)$-dimensional ball of radius $\sqrt{n}$ centered at v must contain all the other integer points in P as any two points of $\{0,1\}^n$ are at most $\sqrt{n}$ apart. Thus the volume of P is bounded by the volume of the ball which is $n^{O(n)}$. Now to get the volume down to $1/n^{10n}$, the number of iterations is still $O(n \log n)$.

In summary, we have reduced our dimension by 1 using $O(n \log n)$ iterations which requires $O(n^2 \log n \cdot \mathrm{EO})$ oracle calls (as each separating hyperplane is computed with $n \cdot \mathrm{EO}$ oracle calls). This can happen at most n times. The overall query complexity is then $O(n^3 \log n \cdot \mathrm{EO})$.

Note that the minimizer $\vec{x}$ obtained may not be integral. This is not a problem as the definition of Lovász extension implies that if $\hat{f}(\vec{x})$ is minimal, then $f(\{i : x_i \geq t\})$ is minimal for any $t \in [0,1]$.

We remark that this algorithm does not have a polynomial runtime. Even though all the integral vertices of P lie on a hyperplane, the best way we know of that identifies it takes exponential time by checking for all the integer points $\{0,1\}^n$. □

Remark 72. Note that this algorithm works for minimizing any convex function over the hypercube that obtains its optimal value at a vertex of the hypercube. Formally, our proof of Theorem 71 holds whenever a function $f : 2^V \longrightarrow \mathbb{R}^n$ admits a convex relaxation $\hat{f}$ with the following properties:

1. For every $S \subseteq V$, $\hat{f}(I_S) = f(S)$.
2. Every $\hat{f}(\vec{x})$ can be written as a convex combination $\sum_{S \in \mathcal{S}} \alpha_S f(S)$, where $\sum \alpha_S = 1$, $|\mathcal{S}| = O(n)$, and $\mathcal{S}$ can be computed without any oracle call.
3. A subgradient $\partial \hat{f}(\vec{x})$ of $\hat{f}$ at any point $\vec{x} \in [0,1]^n$ can be computed with $O(n \cdot \mathrm{EO})$ oracle calls.

In this case, the proof of Theorem 71, implies that $\hat{f}$ and f can be minimized with $O(n^3 \log n \cdot \mathrm{EO})$ oracle calls by using the separating hyperplane $\partial \hat{f}(\bar{x})^T(\vec{x} - \bar{x}) \leq 0$.

15.2 Technical Tools

To improve upon the running time of the algorithm in the previous section, we use more structure of our submodular function f. Rather than merely showing that we can decrease the dimension of our SFM problem by 1 we show how we can reduce the degrees of freedom of our problem in a more principled way. In Section 15.2.1 we formally define the abstraction we use for this and discuss how to change our separation oracle to accommodate this abstraction, and in Section 15.2.2 we show how we can deduce these constraints. These tools serve as the foundation for the faster strongly polynomial time SFM algorithms we present in Section 15.3 and Section 15.4.

15.2.1 SFM over Ring Family

For the remainder of the paper we consider a more general problem than SFM in which we wish to compute a minimizer of our submodular function f over a ring family of the ground set $V = [n]$. A ring family $\mathcal{F}$ is a collection of subsets of V such that for any $S_1, S_2 \in \mathcal{F}$, we have $S_1 \cup S_2, S_1 \cap S_2 \in \mathcal{F}$. Thus SFM corresponds to the special case where $\mathcal{F}$ consists of every subset of V. This generalization has been considered before in the literature and was essential to the IFF algorithm.

It is well known that any ring family $\mathcal{F}$ over V can be represented by a directed graph $D = (V, A)$ where $S \in \mathcal{F}$ iff S contains all of the descendants of any $i \in S$. An equivalent definition is that for

any arc $(i,j) \in A$, $i \in S$ implies $j \in S$. It is customary to assume that A is acyclic as any (directed) cycle of A can be contracted (see section 15.3.1).

We denote by $R(i)$ the set of descendants of i (including i itself) and $Q(i)$ the set of ancestors of i (including i itself). Polyhedrally, an arc $(i,j) \in A$ can be encoded as the constraint $x_i \leq x_j$ as shown by the next lemma.

Lemma 73. *Let $\mathcal{F}$ be a ring family over V and $D = (V, A)$ be its directed acyclic graph representation. Suppose $f : V \longrightarrow \mathbb{R}$ is submodular with Lovász extension $\hat{f}$. Then the characteristic vector I_S of any minimizer $S = \arg\min_{S \in \mathcal{F}} f(S)$ over $\mathcal{F}$ is also the solution to*

$$\begin{aligned} \min & \hat{f}(\vec{x}) \\ & x_i \leq x_j \, \forall (i,j) \in A \\ & 0 \leq \vec{x} \leq 1 \end{aligned} \tag{15.1}$$

Proof. Let x^* be a minimizer, and $L^{(t)} = \{i : x_i^* \geq t\}$. It is easy to check that the indicator variable $I_{L^{(t)}}$ satisfies (15.1) since x^* does. Moreover, recall that $\hat{f}(x^*) = \mathbb{E}_{t \sim [0,1]}[f(L_t)]$. Thus $\hat{f}(x^*)$ can be written as a convex combination $\hat{f}(x^*) = \sum_t \alpha_t f(L^{(t)}) = \sum_t \alpha_t \hat{f}(I_{L^{(t)}})$, where $\alpha_t > 0$ and $\sum_t \alpha_t = 1$. Thus all such $\hat{f}(I_{L^{(t)}})$ are minimal, i.e. (15.1) has no "integrality gap". □

We also modify our separation oracle to accommodate for this generalization as follows. Before doing so we need a definition which relates our BFS to the ring family formalism.

Definition 74. A permutation $(v_1, \ldots, v_n)$ of V is said to be *consistent* with an arc (i,j) if j precedes i in $(v_1, \ldots, v_n)$. Similarly, a BFS of the base polyhedron is consistent with (i,j) if j precedes i in its defining permutation. $(v_1, \ldots, v_n)$ (or a BFS) is consistent with A if it is consistent with every $(i,j) \in A$.

Readers may find it helpful to keep in mind the following picture which depicts the relative positions between $R(i), i, Q(i)$ in the defining permutation of $\vec{h}$ that is consistent with A:

$$\cdots\cdots \; R(i) \backslash \{i\} \; \cdots\cdots \; i \; \cdots\cdots \; Q(i) \backslash \{i\} \; \cdots\cdots$$

In Theorem 61, given $\bar{x} \in [0,1]^n$ our separating hyperplane is constructed by sorting the entries of $\bar{x}$. This hyperplane is associated with some BFS $\vec{h}$ of the base polyhedron. As we shall see towards the end of the section, we would like $\vec{h}$ to be consistent with every arc $(i,j) \in A$.

This task is easy initially as $\bar{x}$ satisfies $x_i \leq x_j$ for $(i,j) \in A$ for the starting polytope of (15.1). If $x_i < x_j$, nothing special has to be done as j must precede i in the ordering. On the other hand, whenever $x_i = x_j$, we can always break ties by ranking j ahead of i.

However, a technical issue arises due to the fact that our cutting plane algorithm may drop constraints from the current feasible region P. In other words, $\bar{x}$ may violate $x_i \geq 0$, $x_j \leq 1$ or $x_i \leq x_j$ if it is ever dropped. Fortunately this can be fixed by reintroducing the constraint. We summarize the modification needed in the pseudocode below and formally show that it fulfills our requirement.

Lemma 75. *Our modified separation oracle returns either some BFS $\vec{h} = 0$ or a valid separating hyperplane, i.e.*

1. *$\bar{x}$ either lies on the separating hyperplane or is cut away by it.*
2. *Any minimizer of (15.1) is not cut away by the separating hyperplane.*

Algorithm 5: Modified Separation Oracle

Input: $\bar{x} \in \mathbb{R}^n$ and the set of arcs A
if $\bar{x}_i < 0$ *for some* i **then**
  Output: $x_i \geq 0$
else if $\bar{x}_j > 1$ *for some* j **then**
  Output: $x_j \leq 1$
else if $\bar{x}_i > \bar{x}_j$ *for some* $(i,j) \in A$ **then**
  Output: $x_i \leq x_j$
else
  Let $i_1, \ldots, i_n$ be a permutation of V such that $\bar{x}_{i_1} \geq \ldots \geq \bar{x}_{i_n}$ and for all $(i,j) \in A$, j precedes i in $i_1, \ldots, i_n$.
  Output: $\vec{h}^T\vec{x} \leq \hat{f}(\bar{x})$, where $\vec{h}$ is the BFS defined by the permutation $i_1, \ldots, i_n$.

Such a hyperplane can be computed with n oracle calls to f and in time $O(n \cdot EO + n^2)$.

Proof. If we get $x_i \geq 0$, $x_j \leq 1$ or $x_i \leq x_j$ (if loop or the first two else loops), then clearly $\bar{x}$ is cut away by it and any minimizer must of course satisfy $x_i \geq 0$, $x_j \leq 1$ and $x_i \leq x_j$ as they are the constraints in (15.1). This proves (1) and (2) for the case of getting $x_i \geq 0$, $x_j \leq 1$ or $x_i \leq x_j$.

Thus it remains to consider the case $\vec{h}^T\vec{x} \leq \hat{f}(\bar{x})$ (last else loop). First of all, $\bar{x}$ lies on it as $\hat{f}(\bar{x}) = \vec{h}^T\bar{x}$. This proves (1). For (2), we have from Lemma 63 that $\vec{h}^T\vec{x} \leq \hat{f}(\vec{x})$. If x^* is a minimizer of (15.1), we must then have $\vec{h}^T x^* \leq \hat{f}(x^*) \leq \hat{f}(\bar{x})$ as $\bar{x}$ is also feasible for (15.1).

Finally we note that the running time is self-evident. □

We stress again that the main purpose of modifying our separation oracle is to ensure that any BFS $\vec{h}$ used to define a new separating hyperplane must be consistent with every $(i,j) \in A$.

15.2.2 Identifying New Valid Arcs

The reason for considering the ring family generalization of SFM is that our algorithms (and some previous algorithms too) work by adding new arcs to our digraph D. This operation yields a strongly polynomial algorithm since there are only $2 \cdot \binom{n}{2}$ possible arcs to add. Of course, a new arc (i,j) is valid only if $i \in S_{\min} \implies j \in S_{\min}$ for some minimizer $S_{\min}$. Here we show how to identify such valid arcs by extracting information from certain nice elements of the base polyhedron.

This is guaranteed by the next four lemmas, which are stated in a way different from previous works e.g. our version is extended to the ring family setting. This is necessary as our algorithms require a more general formulation. We also give a new polyhedral proof, which is mildly simpler than the previous combinatorial proof. On the other hand, Lemma 80 is new and unique to our work. It is an important ingredient of our $\widetilde{O}(n^3 \cdot \mathrm{EO} + n^4)$ time algorithm.

Recall that each BFS of the base polyhedron is defined by some permutation of the ground set elements.

First, we prove the following two lemmas which show that should we ever encounter a non-degenerate point in the base polytope with a coordinate of very large value, then we can immediately conclude that that coordinate must be or must not be in solution to SFM over the ring family.

Lemma 76. *If $\vec{y} \in \mathcal{B}(f)$ is non-degenerate and satisfies $y_i > -(n-1)\min_j y_j$, then i is not in any minimizer of f (over the ring family $\mathcal{A}$).*

Proof. We proceed by contradiction and suppose that S is a minimizer of f that contains i. Now since $\vec{y}$ is non-degenerate we know that $\min_j y_j \leq 0$ and by the definition of $\vec{y}$ we have the following contradiction

$$0 < y_i + (n-1)\min_j y_j \leq \sum_{j \in S} y_j = \vec{y}(S) \leq f(S) \leq f(\emptyset) = 0\,.$$

□

Lemma 77. *If $\vec{y} \in \mathcal{B}(f)$ is non-degenerate and satisfies $y_i < -(n-1)\max_j y_j$, then i is in every minimizer of f (over the ring family A).*

Proof. We proceed by contradiction and suppose that S is a minimizer of f that does not contain i. Now since $\vec{y}$ is non-degenerate we know that $\max_j y_j \geq 0$ and therefore

$$\sum_{j \in [n]} y_j = y_i + \sum_{j \in S} y_j + \sum_{j \in V-(S+i)} y_j < -(n-1)\max_j y_j + \sum_{j \in S} y_j + (|V| - |S| - 1)\max_j y_j \leq \sum_{j \in S} y_j\,.$$

However by the definition of $\vec{y}$ we have

$$\sum_{j \in S} y_j = \vec{y}(S) \leq f(S) \leq f(V) = \sum_{j \in [n]} y_j\,.$$

Thus we have a contradiction and the result follows. □

Now we are ready to present conditions under which a new valid arc can be added. We begin with a simple observation. Let $\texttt{upper}(i) \stackrel{\text{def}}{=} f(R(i)) - f(R(i) - i)$ and $\texttt{lower}(i) \stackrel{\text{def}}{=} f(V \backslash Q(i) + i) - f(V \backslash Q(i))$. As the names suggest, they bound the value of h_i for any BFS used.

Lemma 78. *For any BFS $\vec{h}$ used to construct a separating hyperplane given by our modified separation oracle, we have $\texttt{lower}(i) \leq h_i \leq \texttt{upper}(i)$.*

Proof. Note that by Lemma 75, $\vec{h}$ is consistent with every $(j_1, j_2) \in A$ and hence i must precede $Q(i)$ and be preceded by $R(i)$. Let S be the set of elements preceding i in the defining permutation of $\vec{h}$. Then $h_i = f(S + i) - f(S) \leq f(R(i)) - f(R(i) - i)$ because of diminishing return and $R(i) - i \subseteq S$. The lower bound follows from the same argument as $Q(i) - i$ comes after i, and so $Q(i) \subseteq V \backslash S$. □

In the following two lemmas, we show that if $\texttt{upper}(i)$ is ever sufficiently positive or $\texttt{lower}(i)$ is sufficiently negative, then we find a new arc.

While these lemmas may appear somewhat technical but actually has an intuitive interpretation. Suppose an element p is in a minimizer $S_{\min}$ of f over the ring family D. Then $R(p)$ must also be part of $S_{\min}$. Now if $f(R(p))$ is very large relative to $f(R(p) - p)$, there should be some element $q \in S_{\min} \backslash R(p)$ compensating for the discrepancy. The lemma says that such an element q can in fact be found efficiently.

Lemma 79 (new arc)**.** *Let $\vec{y} = \sum_k \lambda^{(k)} \vec{y}^{(k)}$ be a non-degenerate convex combination of $O(n)$ base polyhedron BFS's $\vec{y}^{(k)}$ which are consistent with every arc $(i, j) \in A$. If some element p satisfies $\texttt{upper}(p) > n^4 \max y_j$, then we can find, using $O(n \cdot EO)$ oracle calls and $O(n^2)$ time, some $q \notin R(p)$ such that the arc (p, q) is valid, i.e. if p is in a minimizer, then so is q.*

Proof. If $\max y_j < 0$ then we are immediately done by Lemma 65. We assume $\max y_j \geq 0$ in the proof. For all k let $\vec{y}'^{(k)}$ be the BFS obtained by taking the defining permutation of $\vec{y}^{(k)}$ and moving $R(p)$ to the front while preserving the relative ordering of $R(p)$ within each permutation). Furthermore, let $\vec{y}' \stackrel{\text{def}}{=} \sum_k \lambda^{(k)} \vec{y}'^{(k)}$. Then since $y'^{(k)}_p = f(R(p)) - f(R(p) - p) = \texttt{upper}(p)$ we have $\texttt{upper}(p) = y'_p = f(R(p)) - f(R(p) - p)$. Moreover,

$$y'_j \geq y_j \quad \forall j \in R(p) \text{ and } y'_j \leq y_j \quad \forall j \notin R(p) \tag{15.2}$$

by diminishing marginal return.

Now, suppose p is in a minimizer $S_{\min}$. Then $R(p) \subseteq S_{\min}$ by definition. We then define $f'(S) = f(S \cup R(p))$ for $S \subseteq V \backslash R(p)$. It can be checked readily that f' is submodular and $S_{\min} \backslash R(p)$ is a minimizer of f' (over the corresponding ring family). Note that now $\vec{y}'_{V \backslash R(p)}$ (the restriction of $\vec{y}'$ to $V \backslash R(p)$) is a convex combination of the BFS's of the base polyhedron $\mathcal{B}(f')$ of f'. We shall show that $\vec{y}'_{V \backslash R(p)}$ has the desired property in Lemma 77.

Note that $y'(V \backslash R(p) + p) \leq y(V \backslash R(p) + p)$ since

$$y'(V \backslash R(p) + p) = y'(V) - y'(R(p) - p) = y(V) - y'(R(p) - p) \leq y(V) - y(R(p) - p) = y(V \backslash R(p) + p).$$

But now since $\vec{y}$ is non-degenerate $\max_j y_j \geq 0$ and therefore

$$\begin{aligned} y'(V \backslash R(p)) &\leq y(V \backslash R(p) + p) - y'_p \\ &= y(V \backslash R(p) + p) - (f(R(p)) - f(R(p) - p)) \\ &\leq n \max y_j - (f(R(p)) - f(R(p) - p)) \\ &< (n - n^4) \max y_j \end{aligned} \tag{15.3}$$

Therefore by the Pigeonhole Principle some $q \notin R(p)$ must satisfy

$$\begin{aligned} y'_q &< \left((n - n^4) \max y_j\right) / (n - 1) \\ &= -(n^3 + n^2 + n) \max y_j \\ &\leq -(n^3 + n^2 + n) \max_{j \notin R(p)} y_j \\ &\leq -(n^3 + n^2 + n) \max_{j \notin R(p)} y'_j \quad \text{by (15.2)} \end{aligned}$$

By Lemma 77, this q must be in any minimizer of f'. In other words, whenever p is in a minimizer of f, then so is q.

Note however that computing all $\vec{y}'$ would take $O(n^2)$ oracle calls in the worst case as there are $O(n)$ $\vec{y}'^{(k)}$'s. We use the following trick to identify some q with $y'_q < -(n-1) \max y_j$ using just $O(n)$ calls. The idea is that we actually only want to have sufficient decreases in $y'(V \backslash R(p))$ which can be accomplished by having a large corresponding decrease in some $\vec{y}'^{(k)}$.

For each k, by the same argument above (see (15.3))

$$y'^{(k)}(V \backslash R(p)) - y^{(k)}(V \backslash R(p)) \leq y^{(k)}_p - (f(R(p)) - f(R(p) - p)) \tag{15.4}$$

The "weighted decrease" $\lambda^{(k)} \left(y^{(k)}_p - (f(R(p)) - f(R(p) - p))\right)$ for $\vec{y}'^{(k)}$ sum up to

$$\sum \lambda^{(k)} \left(y^{(k)}_p - (f(R(p)) - f(R(p) - p))\right) = y_p - (f(R(p)) - f(R(p) - p)) < (1 - n^4) \max y_j$$

Thus by the Pigeonhole Principle, some l will have

$$\lambda^{(l)} \left(y^{(l)}_p - (f(R(p)) - f(R(p) - p))\right) < \left((1 - n^4) \max y_j\right) / O(n) < -n^2 \max y_j.$$

For this $\vec{y}^{(l)}$ we compute $\vec{y}'^{(l)}$. We show that $\vec{y}'' = \lambda^{(l)}\vec{y}'^{(l)} + \sum_{k\neq l}\lambda^{(k)}\vec{y}^{(k)}$ has the same property as $\vec{y}'$ above.

$$\begin{aligned}
y''(V\backslash R(p)) &= \lambda^{(l)}y'^{(l)}(V\backslash R(p)) + \sum_{k\neq l}\lambda^{(k)}y^{(k)}(V\backslash R(p)) \\
&= y(V\backslash R(p)) + \lambda^{(l)}\left(y'^{(l)}(V\backslash R(p)) - y^{(l)}(V\backslash R(p))\right) \\
&\leq y(V\backslash R(p)) + \lambda^{(l)}\left(y_p^{(l)} - (f(R(p)) - f(R(p)-p))\right) \quad \text{by (15.4)} \\
&< (n-1)\max y_j - n^2 \max y_j \\
&< (n-n^2)\max y_j
\end{aligned}$$

Then some $q \in V\backslash R(p)$ must satisfy

$$y''_q < \frac{n-n^2}{n-1}\max y_j = -n\max y_j$$

That is, the arc (p,q) is valid. This takes $O(n)$ oracle calls as given $\vec{y} = \sum_k \lambda^{(k)}\vec{y}^{(k)}$, computing $\vec{y}''$ requires knowing only $f(R(p))$, $f(R(p)-p)$, and $\vec{y}'^{(l)}$ which can be computed from $\vec{y}^{(l)}$ with n oracle calls. The runtime is $O(n^2)$ which is needed for computing $\vec{y}''$. □

Lemma 80. *Let $\vec{y} = \sum_k \lambda^{(k)}\vec{y}^{(k)}$ be a non-degenerate convex combination of base polyhedron BFS $\vec{y}^{(k)}$ which is consistent with every arc $(i,j) \in A$. If $\texttt{lower}(p) < n^4 \min y_j$, then we can find, using $O(n \cdot EO)$ oracle calls and $O(n^2)$ time, some $q \notin Q(p)$ such that the arc (q,p) is valid, i.e. if p is not in a minimizer, then q is not either.*

Proof. It is possible to follow the same recipe in the proof of Lemma 79 but using Lemma 76 instead of Lemma 77. Here we offer a proof which directly invokes Lemma 77 on a different submodular function.

Let g be defined by $g(S) \stackrel{\text{def}}{=} f(V\backslash S)$ for any S, and A_g be the set of arcs obtained by reversing the directions of the arcs of A. Consider the problem of minimizing g over the ring family A_g. Using subscripts to avoid confusion with f and g, e.g. $R_g(i)$ is the set of descendants of i w.r.t. A_g, it is not hard to verify the following:

- g is submodular
- $R_g(i) = Q_f(i)$
- $g(R_g(p)) - g(R_g(p)-p) = -\left(f(V\backslash Q_f(p)+p) - f(V\backslash Q_f(p))\right)$
- $-\vec{y}^{(k)}$ is a BFS of $\mathcal{B}(g)$ if and only if $\vec{y}^{(k)}$ is a BFS of $\mathcal{B}(f)$
- $\max(-y_j) = -\min y_j$

By using the above correspondence and applying Lemma 79 to g and A_g, we can find, using $O(n)$ oracle calls and $O(n^2)$ time, some $q \notin R_g(p) = Q(p)$ such that the arc (p,q) is valid for g and A_g. In other words, the reverse (q,p) will be valid for f and A. □

These lemmas lay the foundation of our algorithm. They suggests that if the positive entries of a point in the base polyhedron are small relative to some $\texttt{upper}(p) = f(R(p)) - f(R(p)-p)$, a new arc (p,q) can be added to A. This can be seen as a robust version of Lemma 65.

Finally, we end the section with a technical lemma that will be used crucially for both of our algorithms. The importance of it would become obvious when it is invoked in our analyses.

Lemma 81. *Let $\vec{h}''$ denote a convex combination of two vectors $\vec{h}$ and $\vec{h}'$ in the base polyhedron, i.e. $\vec{h}'' = \lambda\vec{h} + (1-\lambda)\vec{h}'$ for some $\lambda \in [0,1]$. Further suppose that*

$$\left\|\vec{h}''\right\|_2 \leq \alpha \min\left\{\lambda\left\|\vec{h}\right\|_2, (1-\lambda)\left\|\vec{h}'\right\|_2\right\}$$

for some $\alpha \leq \frac{1}{2\sqrt{n}}$. Then for $p = \arg\max_j(\max\{\lambda|h_j|, (1-\lambda)|h'_j|\})$ we have

$$\texttt{lower}(p) \leq -\frac{1}{2\alpha\sqrt{n}} \cdot \left\|\vec{h}''\right\|_\infty \quad \textit{and} \quad \texttt{upper}(p) \geq \frac{1}{2\alpha\sqrt{n}} \cdot \left\|\vec{h}''\right\|_\infty \quad .$$

Proof. Suppose without loss of generality that $\lambda|h_p| \geq (1-\lambda)|h'_p|$. Then by assumptions we have

$$\left\|\vec{h}''\right\|_\infty \leq \left\|\vec{h}''\right\|_2 \leq \alpha \cdot \min\left\{\lambda\left\|\vec{h}\right\|_2, (1-\lambda)\left\|\vec{h}'\right\|_2\right\} \leq \alpha\sqrt{n}\,|\lambda h_p| \quad .$$

However, since $\alpha \leq \frac{1}{2\sqrt{n}}$ we see that

$$\left|\lambda h_p + (1-\lambda)h'_p\right| \leq \left\|h''\right\|_\infty \leq \alpha\sqrt{n}\,|\lambda h_p| \leq \frac{1}{2}\,|\lambda h_p| \quad .$$

Consequently, λh_p and $(1-\lambda)h'_p$ have opposite signs and $\left|(1-\lambda)h'_p\right| \geq \frac{1}{2}\left|\lambda h'_p\right|$. We then have,

$$\texttt{lower}(p) \leq \min\left\{h_p, h'_p\right\} \leq \min\left\{\lambda h_p, (1-\lambda)h'_p\right\} \leq -\frac{1}{2}\,|\lambda h_p| \leq -\frac{1}{2\alpha\sqrt{n}}\left\|h''\right\|_\infty$$

and

$$\texttt{upper}(p) \geq \max\left\{h_p, h'_p\right\} \geq \max\left\{\lambda h_p, (1-\lambda)h'_p\right\} \geq \frac{1}{2}\,|\lambda h_p| \geq \frac{1}{2\alpha\sqrt{n}}\left\|h''\right\|_\infty .$$

□

15.3 $\widetilde{O}(n^4 \cdot \mathbf{EO} + n^5)$ Time Algorithm

Here we present a $\widetilde{O}(n^4 \cdot \mathrm{EO} + n^5)$ time, i.e. strongly polynomial time algorithm, for SFM. We build upon the algorithm achieved in the section to achieve a faster running time in Section 15.4.

Our new algorithm combines the existing tools for SFM developed over the last decade with our cutting plane method. While there are certain similarities with previous algorithms (especially [54, 60, 56]), our approach significantly departs from all the old approaches in one important aspect.

All of the previous algorithms actively maintain a point in the base polyhedron and represent it as a *convex combination* of BFS's. At each step, a new BFS may enter the convex combination and an old BFS may exit. Our algorithm, on the other hand, maintains only a *collection* of BFS's (corresponding to our separating hyperplanes), rather than an explicit convex combination. A "good" convex combination is computed from the *collection* of BFS's only after running Cutting Plane for enough iterations. We believe that this crucial difference is the fundamental reason which offers the speedup. This is achieved by the Cutting Plane method which considers the *geometry* of the collection of BFS's. On the other hand, considering only a convex combination of BFS's effectively narrows our sight to only one *point* in the base polyhedron.

Overview

Now we are ready to describe our strongly polynomial time algorithm. Similar to the weakly polynomial algorithm, we first run our cutting plane for enough iterations on the initial feasible region $\{\vec{x} \in [0,1]^n : x_i \leq x_j \; \forall (i,j) \in A\}$, after which a pair of approximately parallel supporting hyperplanes F_1, F_2 of width $1/n^{\Theta(1)}$ can be found. Our strategy is to write F_1 and F_2 as a nonnegative combination of the facets of remaining feasible region P. This combination is made up of newly added separating hyperplanes as well as the inequalities $x_i \geq 0$, $x_j \leq 1$ and $x_i \leq x_j$. We then argue that one of the following updates can be done:

- Collapsing: $x_i = 0$, $x_j = 1$ or $x_i = x_j$
- Adding a new arc (i,j): $x_i \leq x_j$ for some $(i,j) \notin A$

The former case is easy to handle by elimination or contraction. If $x_i = 0$, we simply eliminate i from the ground set V; and if $x_i = 1$, we redefine f so that $f(S) = f(S+i)$ for any $S \subseteq V - i$. $x_i = x_j$ can be handled in a similar fashion. In the latter case, we simply add the arc (i,j) to A. We then repeat the same procedure on the new problem.

Roughly speaking, our strongly polynomial time guarantee follows as eliminations and contractions can happen at most n times and at most $2 \cdot \binom{n}{2}$ new arcs can be added. While the whole picture is simple, numerous technical details come into play in the execution. We advise readers to keep this overview in mind when reading the subsequent sections.

Algorithm

Our algorithm is summarized below. Again, we remark that our algorithm simply uses Theorem 82 regarding our cutting plane and is agnostic as to how the cutting plane works, thus it could be replaced with other methods, albeit at the expense of slower runtime.

1. Run cutting plane on (15.1) (Theorem 82 with $\tau = \Theta(1)$) using our modified separation oracle (Section 15.2.1).
2. Identify a pair of "narrow" approximately parallel supporting hyperplanes or get some BFS $\vec{h} = 0$ (in which case both $\emptyset$ and V are minimizers).
3. Deduce from the hyperplanes some new constraint of the forms $x_i = 0, x_j = 1, x_i = x_j$ or $x_i \leq x_j$ (Section 15.3.2).
4. Consolidate A and f (Section 15.3.1).
5. Repeat by running our cutting plane method on (15.1) with updated A and f. (Note that Any previously found separating hyperplanes are discarded.)

We call step (1) a *phase* of cutting plane. The minimizer can be constructed by unraveling the recursion.

15.3.1 Consolidating A and f

Here we detail how the set of valid arcs A and submodular function f should be updated once we deduce new information $x_i = 0, x_i = 1, x_i = x_j$ or $x_i \leq x_j$. Recall that $R(i)$ and $Q(i)$ are the sets of descendants and ancestors of i respectively (including i itself). The changes below are somewhat self-evident, and are actually used in some of the previous algorithms so we only sketch how they are done without a detailed justification.

Changes to the digraph representation D of our ring family include:

- $x_i = 0$: remove $Q(i)$ from the ground set and all the arcs incident to $Q(i)$
- $x_i = 1$: remove $R(i)$ from the ground set and all the arcs incident to $R(i)$
- $x_i = x_j$: contract i and j in D and remove any duplicate arcs
- $x_i \leq x_j$: insert the arc (i, j) to A
- For the last two cases, we also contract the vertices on a directed cycle of A until there is no more. Remove any duplicate arcs.

Here we can contract any cycle $(i_1, \ldots, i_k)$ because the inequalities $x_{i_1} \leq x_{i_2}, \ldots, x_{i_{k-1}} \leq x_{i_k}, x_{i_k} \leq x_{i_1}$ imply $x_{i_1} = \ldots = x_{i_k}$.

Changes to f:

- $x_i = 0$: replace f by $f' : 2^{V \setminus Q(i)} \longrightarrow \mathbb{R}$, $f'(S) = f(S)$ for $S \subseteq V \setminus Q(i)$
- $x_i = 1$: replace f by $f' : 2^{V \setminus R(i)} \longrightarrow \mathbb{R}$, $f'(S) = f(S \cup R(i))$ for $S \subseteq V \setminus R(i)$
- $x_i = x_j$: see below
- $x_i \leq x_j$: no changes to f needed if it does not create a cycle in A; otherwise see below
- Contraction of $C = \{i_1, \ldots, i_k\}$: replace f by $f' : 2^{V \setminus C + l} \longrightarrow \mathbb{R}$, $f'(S) = f(S)$ for $S \subseteq V \setminus C$ and $f'(S) = f((S - l) \cup C)$ for $S \ni l$

Strictly speaking, these changes are in fact *not* needed as they will automatically be taken care of by our cutting plane method. Nevertheless, performing them lends a more natural formulation of the algorithm and simplifies its description.

15.3.2 Deducing New Constraints $x_i = 0$, $x_j = 1$, $x_i = x_j$ or $x_i \leq x_j$

Here we show how to deduce new constraints through the result of our cutting plane method. This is the most important ingredient of our algorithm. As mentioned before, similar arguments were used first by IFF [56] and later in [54, 60]. There are however two important differences for our method:

- We maintain a collection of BFS's rather a convex combination; a convex combination is computed and needed only after each phase of cutting plane.
- As a result, our results are proved mostly *geometrically* whereas the previous ones were proved mostly *combinatorially*.

Our ability to deduce such information hinges on the power of the cutting plane method in Part I. We re-state our main result Theorem 31 in the language of SFM. Note that Theorem 82 is formulated in a fairly general manner in order to accommodate for the next section. Readers may wish to think $\tau = \Theta(1)$ for now.

Theorem 82 (Theorem 31 restated for SFM)**.** *For any $\tau \geq 100$, applying our cutting plane method, Theorem 82, to (15.1) with our modified separation oracle (or its variant in Section 15.4) with high probability in n either*

1. *Finds a degenerate BFS $\vec{h} \geq \vec{0}$ or $\vec{h} \leq \vec{0}$.*

2. *Finds a polytope P consisting of $O(n)$ constraints which are our separating hyperplanes or the constraints in (15.1). Moreover, P satisfies the following inequalities*

$$\vec{c}^T\vec{x} \le M \quad \text{and} \quad \vec{c}'^T\vec{x} \le M',$$

both of which are nonnegative combinations of the constraints of P, where $||\vec{c}+\vec{c}'||_2 \le \min\{||\vec{c}||_2, ||\vec{c}'||_2\}/n^{\Theta(\tau)}$ and $|M+M'| \le \min\{||\vec{c}||_2, ||\vec{c}'||_2\}/n^{\Theta(\tau)}$.

Furthermore, the algorithm runs in expected time $O(n^2\tau \log n \cdot EO + n^3\tau^{O(1)}\log^{O(1)} n)$.

Proof. In applying Theorem 82 we let K be the set of minimizers of f over the ring family and the box is the hypercube with $R = 1$. We run cutting plane with our modified separation oracle (Lemma 75). The initial polytope $P^{(0)}$ can be chosen to be, say, the hypercube. If some separating hyperplane is degenerate, then we have the desired result (and know that either $\emptyset$ or V is optimal). Otherwise let P be the current feasible region. Note that $P \neq \emptyset$, because our minimizers of $\hat{f}$ are all in $P^{(0)}$ and $P^{(k)}$ as they are never cut away by the separating hyperplanes.

Let $\mathcal{S}$ be the collection of inequalities (15.1) as well as the separating hyperplanes $\vec{h}^T\vec{x} \le \hat{f}(\bar{x}_h) = \vec{h}^T\bar{x}_h$ used. By Theorem 31, all of our minimizers will be contained in P, consisting of $O(n)$ constraints $\mathbf{A}\vec{x} \ge \vec{b}$. Each such constraint $\vec{a}_i^T\vec{x} \ge b_i$ is a scaling and shifting of some inequality $\vec{p}_i^T\vec{x} \ge q_i$ in $\mathcal{S}$, i.e. $\vec{a}_i = \vec{p}_i/||\vec{p}_i||_2$ and $b_i \le q_i/||\vec{p}_i||_2$.

By taking $\epsilon = 1/n^{\Theta(\tau)}$ with sufficiently large constant in Θ, our theorem certifies that P has a narrow width by $\vec{a}_1$, some nonnegative combination $\sum_{i=2}^{O(n)} t_i\vec{a}_i$ and point $\vec{x}_o \in P$ with $||\vec{x}_o||_\infty \le 3\sqrt{n}R = 3\sqrt{n}$ satisying the following:

$$\left\|\vec{a}_1 + \sum_{i=2}^{O(n)} t_i\vec{a}_i\right\|_2 \le 1/n^{\Theta(\tau)}$$

$$0 \le \vec{a}_1^T\vec{x}_o - \vec{b}_1 \le 1/n^{\Theta(\tau)}$$

$$0 \le \left(\sum_{i=2}^{O(n)} t_i a_i\right)^T \vec{x}_o - \sum_{i=2}^{O(n)} t_i b_i \le 1/n^{\Theta(\tau)}$$

We convert these inequalities to $\vec{p}$ and q. Let $t_i' \stackrel{\text{def}}{=} t_i \cdot ||\vec{p}_1||_2/||\vec{p}_i||_2 \ge 0$.

$$\left\|\vec{p}_1 + \sum_{i=2}^{O(n)} t_i'\vec{p}_i\right\|_2 \le ||\vec{p}_1||_2/n^{\Theta(\tau)}$$

$$0 \le \vec{p}_1^T\vec{x}_o - q_1 \le ||\vec{p}_1||_2/n^{\Theta(\tau)}$$

$$0 \le \left(\sum_{i=2}^{O(n)} t_i'\vec{p}_i\right)^T \vec{x}_o - \sum_{i=2}^{O(n)} t_i' q_i \le ||\vec{p}_1||_2/n^{\Theta(\tau)}$$

We claim that[6] $\vec{c} = -\vec{p}_1$, $M = -q_1$, $\vec{c}' = -\sum_{i=2}^{O(n)} t_i'\vec{p}_i$, $M' = -\sum_{i=2}^{O(n)} t_i' q_i$ satisfy our requirement.

[6]Minus signs is needed because we express our inequalities as e.g. $\vec{h}^T\vec{x} \le \vec{h}^T\bar{x}_h$ whereas in Theorem 31, $\vec{a}_i^T\vec{x} \ge b_i$ is used. We apologize for the inconvenience.

We first show that $||\vec{c}+\vec{c}'||_2 \le \min\{||\vec{c}||_2, ||\vec{c}'||_2\}/n^{\Theta(\tau)}$. We have $||\vec{c}+\vec{c}'||_2 \le ||\vec{c}||_2/n^{\Theta(\tau)}$ from the first inequality. If $||\vec{c}||_2 \le ||\vec{c}'||_2$ we are done. Otherwise, by triangle inequality

$$||\vec{c}'||_2 - ||\vec{c}||_2 \le ||\vec{c}+\vec{c}'||_2 \le ||\vec{c}||_2/n^{\Theta(\tau)} \implies 2||\vec{c}||_2 \ge ||\vec{c}'||_2$$

and hence $||\vec{c}+\vec{c}'||_2 \le ||\vec{c}||_2/n^{\Theta(\tau)} \le ||\vec{c}'||_2/2n^{\Theta(\tau)} = ||\vec{c}'||_2/n^{\Theta(\tau)}$.

We also need to prove $|M + M'| \le \min\{||\vec{c}||_2, ||\vec{c}'||_2\}/n^{\Theta(\tau)}$. Summing the second and third inequalities,

$$-||\vec{c}||_2/n^{\Theta(\tau)} \le (\vec{c}+\vec{c}')^T\vec{x}_o - (M + M') \le 0$$

Recall that we have $||\vec{x}_o||_\infty \le 3\sqrt{n}$. Then

$$\begin{aligned} |M + M'| &\le |(\vec{c}+\vec{c}')^T\vec{x}_o - (M + M')| + |(\vec{c}+\vec{c}')^T\vec{x}_o| \\ &\le ||\vec{c}||_2/n^{\Theta(\tau)} + 3\sqrt{n}||\vec{c}+\vec{c}'||_2 \\ &\le ||\vec{c}||_2/n^{\Theta(\tau)} + 3\sqrt{n}||\vec{c}||_2/n^{\Theta(\tau)} \\ &= ||\vec{c}||_2/n^{\Theta(\tau)} \end{aligned}$$

as desired. Our result then follows as we proved $2||\vec{c}'||_2 \ge ||\vec{c}||_2$.

Finally, we have the desired runtime as our modified separation oracle runs in time $O(n \cdot \mathrm{EO} + n^2 \log^{O(1)} n)$. □

Informally, the theorem above simply states that after $O(n\tau \log n)$ iterations of cutting plane, the remaining feasible region P can be sandwiched between two approximately parallel supporting hyperplanes of width $1/n^{O(\tau)}$. A good intuition to keep in mind is that every $O(n)$ iterations of cutting plane reduces the minimum width by a constant factor.

Remark 83. As shown in the proof of Theorem 82, one of the two approximately parallel hyperplanes can actually be chosen to be a constraint of our feasible region P. However we do not exploit this property as it does not seem to help us and would break the notational symmetry in $\vec{c}$ and $\vec{c}'$.

Setup

In each phase, we run cutting plane using Theorem 82 with $\tau = \Theta(1)$. If some separating hyperplane used is degenerate, we have found the minimizer by Lemma 65.

Now assume none of the separating hyperplanes is degenerate. By Theorem 82, P is sandwiched by a pair of approximately parallel supporting hyperplanes F, F' which are of width $1/10n^{10}$ apart. The width here can actually be $1/n^c$ for any constant c by taking a sufficiently large constant in Theta.

Here, we show how to deduce from F and F' some $x_i = 0,, x_j = 1, x_i = x_j$, or $x_i \le x_j$ constraint on the minimizers of f over the ring family. Let

$$\vec{c}^T\vec{x} = \sum c_i x_i \le M \quad \text{and} \quad \vec{c}'^T\vec{x} = \sum c'_i x_i \le M'$$

be the inequality for F and F' such that

$$|M + M'|,\ ||\vec{c}+\vec{c}'||_2 \le \mathtt{gap}, \quad \text{where } \mathtt{gap} \stackrel{\text{def}}{=} \frac{1}{10n^{10}} \min\{||\vec{c}||_2, ||\vec{c}'||_2\}.$$

By the same theorem we can write $\vec{c}^T\vec{x} \le M$ as a nonnegative combination of the constraints for P. Recall that the constraints for P take on four different forms: (1) $-x_i \le 0$; (2) $x_j \le 1$; (3) $-(x_j - x_i) \le 0$; (4) $\vec{h}^T\vec{x} = \sum h_i x_i \le \hat{f}(\bar{x}_h)$. Here the first three types are present initially whereas

the last type is the separating hyperplane added. As alleged previously, the coefficient vector $\vec{h}$ corresponds to a BFS of the base polyhedron for f. Our analysis crucially exploits this property.

Thus suppose $\vec{c}^T\vec{x} = \sum_i c_i x_i \leq M$ is a nonnegative combination of our constraints with weights $\alpha_i, \beta_j, \gamma_{ij}, \lambda_h \geq 0$. The number of (positive) $\alpha_i, \beta_j, \gamma_{ij}, \lambda_h$ is at most $O(n)$. Here we denote separating hyperplanes by $\vec{h}^T\vec{x} \leq \hat{f}(\bar{x}_h)$. Let H be the set of BFS's used to construct separating hyperplanes.

$$\vec{c}^T\vec{x} = -\sum_i \alpha_i x_i + \sum_j \beta_j x_j + \sum_{(i,j)\in A} \gamma_{ij}(x_i - x_j) + \sum_{h\in H} \lambda_h \vec{h}^T\vec{x} \quad \text{and} \quad M = \sum_j \beta_j + \sum_{h\in H} \lambda_h \hat{f}(\bar{x}_h). \tag{15.5}$$

Similarly, we write the inequality for F' as a nonnegative combination of the constraints for P and the number of (positive) $\alpha'_i, \beta'_j, \gamma'_{ij}, \lambda'_h$ is $O(n)$:

$$\vec{c}'^T\vec{x} = -\sum \alpha'_i x_i + \sum \beta'_j x_j + \sum_{(i,j)\in A} \gamma'_{ij}(x_i - x_j) + \sum_{h\in H} \lambda'_h \vec{h}^T\vec{x} \quad \text{and} \quad M' = \sum \beta'_j + \sum_{h\in H} \lambda'_h \hat{f}(\bar{x}_h). \tag{15.6}$$

We also scale $\vec{c}, \vec{c}', \alpha, \alpha', \beta, \beta', \gamma, \gamma', \lambda, \lambda'$ so that

$$\sum_{h\in H} (\lambda_h + \lambda'_h) = 1$$

as this does not change any of our preceding inequalities regarding F and F'.

Now that F, F' have been written as combinations of our constraints, we have gathered the necessary ingredients to derive our new arc. We first give a geometric intuition why we would expect to be able to derive a new constraint. Consider the nonnegative combination making up F. We think of the coefficient β_j as the contribution of $x_j \leq 1$ to F. Now if β_j is very large, F is "very parallel" to $x_j \leq 1$ and consequently F' would miss $x_j = 0$ as the gap between F and F' is small. P would then miss $x_j = 0$ too as it is sandwiched between F and F'. Similarly, a large α_i and a large γ_{ij} would respectively imply that $x_i = 1$ and $(x_i = 0, x_j = 1)$ would be missed. The same argument works for F' as well.

But on the other hand, if the contributions from $x_i \geq 0, x_j \leq 1, x_i \leq x_j$ to both F and F' are small, then the supporting hyperplanes $\vec{c}^T\vec{x} \leq \ldots$ and $\vec{c}'^T\vec{x} \leq \ldots$ would be mostly made up of separating hyperplanes $\vec{h}^T\vec{x} \leq \hat{f}(\bar{x}_h)$. By summing up these separating hyperplanes (whose coefficients form BFS's), we would then get a point in the base polyhedron which is very close to the origin 0. Moreover, by Lemma 81 and Lemma 79 we should then be able to deduce some interesting information about the minimizer of f over D.

The rest of this section is devoted to realizing the vision sketched above. We stress that while the algebraic manipulations may be long, they are simply the execution of this elegant geometric picture.

Now, consider the following weighted sum of $\vec{h}^T\vec{x} \leq \hat{f}(\bar{x}_h)$:

$$\left(\sum_{h\in H} \lambda_h \vec{h}^T + \sum_{h\in H} \lambda'_h \vec{h}^T\right)\vec{x} = \sum_{h\in H} \lambda_h \vec{h}^T\vec{x} + \sum_{h\in H} \lambda'_h \vec{h}^T\vec{x} \leq \sum_{h\in H} \lambda_h \hat{f}(\bar{x}_h) + \sum_{h\in H} \lambda'_h \hat{f}(\bar{x}_h).$$

Observe that $\sum_{h\in H}\lambda_h\vec{h}^T+\sum_{h\in H}\lambda'_h\vec{h}^T$ is in the base polyhedron since it is a convex combination of BFS $\vec{h}$. Furthermore, using (15.5) and (15.6) this can also be written as

$$\begin{aligned}\left(\sum_{h\in H}\lambda_h\vec{h}^T+\sum_{h\in H}\lambda'_h\vec{h}^T\right)\vec{x} = &\left(\vec{c}^T\vec{x}+\sum\alpha_i x_i-\sum\beta_j x_j+\sum_{(i,j)\in A}\gamma_{ij}(x_j-x_i)\right)\\ &+\left(\vec{c}'^T\vec{x}+\sum\alpha'_i x_i-\sum\beta'_j x_j+\sum_{(i,j)\in A}\gamma'_{ij}(x_j-x_i)\right)\end{aligned} \tag{15.7}$$

and

$$\begin{aligned}\sum_{h\in H}\lambda_h\hat{f}(\bar{x}_h)+\sum_{h\in H}\lambda'_h\hat{f}(\bar{x}_h) &= \left(M-\sum\beta_j\right)+\left(M'-\sum\beta'_j\right)\\ &= (M+M')-\sum\beta_j-\sum\beta'_j\end{aligned}$$

Furthermore, we can bound $\vec{c}^T\vec{x}+\vec{c}'^T\vec{x}$ by $\vec{c}^T\vec{x}+\vec{c}'^T\vec{x}\geq-||\vec{c}+\vec{c}'||_1\geq-\sqrt{n}||\vec{c}+\vec{c}'||_2\geq-\sqrt{n}\mathtt{gap}$ as $\vec{x}\leq 1$. Since $M+M'\leq\mathtt{gap}$, we obtain

$$\begin{aligned}LHS\overset{\text{def}}{=}&\sum\alpha_i x_i+\sum\alpha'_i x_i-\sum\beta_j x_j-\sum\beta'_j x_j+\sum_{(i,j)\in A}\gamma_{ij}(x_j-x_i)+\sum_{(i,j)\in A}\gamma'_{ij}(x_j-x_i)\\ &\leq 2\sqrt{n}\mathtt{gap}-\sum\beta_j-\sum\beta'_j\end{aligned}$$

Geometrically, the next lemma states that if the contribution from, say $x_i\geq 0$, to F is too large, then F' would be forced to miss $x_i=1$ because they are close to one another.

Lemma 84. *Suppose $\vec{x}$ satisfies (15.1) and $LHS\leq 2\sqrt{n}\mathtt{gap}-\sum\beta_j-\sum\beta'_j$ with $\alpha_i,\beta_j,\gamma_{ij},\alpha'_i,\beta'_j,\gamma'_{ij}\geq 0$.*

1. *If $\alpha_i>2\sqrt{n}\mathtt{gap}$ or $\alpha'_i>2\sqrt{n}\mathtt{gap}$, then $x_i<1$.*
2. *If $\beta_j>2\sqrt{n}\mathtt{gap}$ or $\beta'_j>2\sqrt{n}\mathtt{gap}$, then $x_j>0$.*
3. *If $\gamma_{ij}>2\sqrt{n}\mathtt{gap}$ or $\gamma'_{ij}>2\sqrt{n}\mathtt{gap}$, then $0\leq x_j-x_i<1$.*

Proof. We only prove it for $\alpha_i,\beta_j,\gamma_{ij}$ as the other case follows by symmetry.

Using $0\leq x\leq 1$ and $x_i\leq x_j$ for $(i,j)\in A$, we have $LHS\geq\alpha_i x_i-\sum\beta_j-\sum\beta'_j$. Hence $\alpha_i x_i\leq 2\sqrt{n}\mathtt{gap}$ and we get $x_i<1$ if $\alpha_i>2\sqrt{n}\mathtt{gap}$.

Similarly, $LHS\geq-\beta_k x_k-\sum_{j\neq k}\beta_j-\sum\beta'_j$ which gives $-\beta_k x_k\leq 2\sqrt{n}\mathtt{gap}-\beta_k$. Then $x_k>0$ if $\beta_k>2\sqrt{n}\mathtt{gap}$.

Finally, $LHS\geq\gamma_{ij}(x_j-x_i)-\sum\beta_j-\sum\beta'_j$ which gives $\gamma_{ij}(x_j-x_i)\leq 2\sqrt{n}\mathtt{gap}$. Then $x_j-x_i<1$ if $\gamma_{ij}>2\sqrt{n}\mathtt{gap}$. We have $x_i\leq x_j$ since $(i,j)\in A$. □

So if either condition of Lemma 84 holds, we can set $x_i=0$ or $x_j=1$ or $x_i=x_j$ since our problem (15.1) has an integral minimizer and any minimizer of $\hat{f}$ is never cut away by Lemma 75. Consequently, in this case we can reduce the dimension by at least 1. From now on we may assume that

$$\max\{\alpha_i,\alpha'_i,\beta_j,\beta'_j,\gamma_{ij},\gamma'_{ij}\}\leq 2\sqrt{n}\mathtt{gap}. \tag{15.8}$$

Geometrically, (15.8) says that if the supporting hyperplanes are both mostly made up of the separating hyperplanes, then their aggregate contributions to F and F' should be small in absolute value.

The next lemma identifies some $p\in V$ for which $f(R(p))-f(R(p)-p)$ is "big". This prepares for the final step of our approach which invokes Lemma 79.

Lemma 85. *Let* $\vec{y} \stackrel{\text{def}}{=} \sum_{h\in H} \lambda_h \vec{h}$ *and* $\vec{y}' \stackrel{\text{def}}{=} \sum_{h\in H} \lambda'_h \vec{h}$ *and let* $p \in \arg\max_l \{\max\{|y_l|, |y'_l|\}\}$ *then*

$$\texttt{upper}(p) \geq n^7 \left\|\vec{y} + \vec{y}'\right\|_\infty$$

assuming (15.8).

Proof. Recall that $\left\|\vec{c} + \vec{c}'\right\|_2 \leq \texttt{gap}$ where $\texttt{gap} = \frac{1}{10n^{10}} \min\{||\vec{c}||_2, ||\vec{c}'||_2\}$,

$$\vec{c} = \vec{y} - \sum_i \alpha_i \vec{\mathbb{1}}_i + \sum_j \beta_j \vec{\mathbb{1}}_j + \sum_{(i,j)} \gamma_{ij}(\vec{\mathbb{1}}_i - \vec{\mathbb{1}}_j) \quad \text{and } \vec{c}' = \vec{y}' - \sum_i \alpha'_i \vec{\mathbb{1}}_i + \sum_j \beta'_j \vec{\mathbb{1}}_j + \sum_{(i,j)} \gamma'_{ij}(\vec{\mathbb{1}}_i - \vec{\mathbb{1}}_j)\,.$$

By (15.8) we know that $\left\|\vec{c} - \vec{y}\right\|_2 \leq 4n^2\texttt{gap} \leq \frac{4}{10n^8}\left\|\vec{c}\right\|_2$ and $\left\|\vec{c}' - \vec{y}'\right\|_2 \leq 4n^2\texttt{gap} \leq \frac{4}{10n^8}\left\|\vec{c}'\right\|_2$. Consequently, by the triangle inequality we have that

$$\left\|\vec{y} + \vec{y}'\right\|_2 \leq \left\|\vec{c} + \vec{c}'\right\|_2 + \left\|\vec{c} - \vec{y}\right\|_2 + \left\|\vec{c}' - \vec{y}'\right\|_2 \leq 9n^2\texttt{gap}$$

and

$$\left\|\vec{c}\right\|_2 \leq \left\|\vec{c} - \vec{y}\right\|_2 + \left\|\vec{y}\right\|_2 \leq \frac{4}{10n^8}\left\|\vec{c}\right\|_2 + \left\|\vec{y}\right\|_2 \quad \Rightarrow \quad \left\|\vec{c}\right\|_2 \leq 2\left\|\vec{y}\right\|_2$$

Similarly, we have that $\left\|\vec{c}'\right\|_2 \leq 2\left\|\vec{y}'\right\|_2$. Consequently since $\texttt{gap} \leq \frac{1}{10n^{10}} \min\{||\vec{c}||_2, ||\vec{c}'||_2\}$, we have that

$$\left\|\vec{y} + \vec{y}'\right\|_2 \leq \frac{2}{n^8} \min\left\{\left\|\vec{y}\right\|_2, \left\|\vec{y}'\right\|_2\right\}$$

and thus, invoking Lemma 81 yields the result. □

We summarize the results in the lemma below.

Corollary 86. *Let* P *be the feasible region after running cutting plane on* (15.1)*. Then one of the following holds:*

1. *We found a degenerate BFS and hence either* $\emptyset$ *or* V *is a minimizer.*
2. *The integral points of* P *all lie on some hyperplane* $x_i = 0$, $x_j = 1$ *or* $x_i = x_j$ *which we can find.*
3. *Let* H *be the collection of BFS's* $\vec{h}$ *used to construct our separating hyperplanes for* P*. Then there is a convex combination* $\vec{y}$ *of* H *such that* $n^4|y_i| < \max_p \texttt{upper}(p)$ *for all* i*.*

Proof. As mentioned before, (1) happens if some separating hyperplane is degenerate. We have (2) if one of the conditions in Lemma 84 holds. Otherwise, $y = \sum_{h\in H} \lambda_h \vec{h} + \sum_{h\in H} \lambda'_h \vec{h}$ is a candidate for Case 3 by Lemma 85. □

Let us revisit the conditions of Lemma 79 and explain that they are satisfied by Case 3 of the last lemma.

- $\vec{y}$ is a convex combination of at most $O(n)$ BFS's. This holds in Case 3 since our current feasible region consists of only $O(n)$ constraints thanks to the Cutting Plane method.
- Those BFS's must be consistent with every arc of A. This holds because Case 3 uses the BFS's for constructing our separating hyperplane. Our modified separation oracle guarantees that they are consistent with A.

Thus in Case 3 of the last corollary, Lemma 79 allows us to deduce a new constraint $x_p \leq x_q$ for some $q \notin R(p)$.

15.3.3 Running Time

Here we bound the total running time of our algorithm and prove the following.

Theorem 87. *Our algorithm runs in time* $O(n^4 \log n \cdot EO + n^5 \log^{O(1)} n)$.

Proof. To avoid being repetitive, we appeal to Corollary 86. Each phase of cutting plane takes time $O(n^2 \log n \cdot \text{EO} + n^3 \log^{O(1)} n)$ (Theorem 82 with τ being a big constant. Given F and F' represented as a nonnegative combination of facets, we can check for the conditions in Lemma 84 in $O(n)$ time as there are only this many facets of P. This settles Case 2 of Corollary 86. Finally, Lemma 79 tells us that we can find a new arc in $O(n \cdot \text{EO} + n^2)$ time for Case 3 of Corollary 86. Our conclusion follows from the fact that we can get $x_i = 0$, $x_i = 1$, $x_i = x_j$ at most n times and $x_i \leq x_j$ at most $O(n^2)$ times. □

15.4 $\widetilde{O}(n^3 \cdot \text{EO} + n^4)$ Time Algorithm

Here we show how to improve our running time for strongly polynomial SFM to $\widetilde{O}(n^3 \cdot \text{EO} + n^4)$. Our algorithm can be viewed as an extension of the algorithm we presented in the previous Section 15.3. The main bottleneck of our previous algorithm was the time needed to identify a new arc, which cost us $\widetilde{O}(n^2 \cdot \text{EO} + n^3)$. Here we show how to reduce our amortized cost for identifying a valid arc down to $\widetilde{O}(n \cdot \text{EO} + n^2)$ and thereby achieve our result.

The key observation we make to improve this running time is that our choice of p for adding an arc in the previous lemma can be relaxed. p actually need not be $\arg\max_i \texttt{upper}(i)$; instead it is enough to have $\texttt{upper}(p) > n^4 \max\{\alpha_i, \alpha'_i, \beta_j, \beta'_j, \gamma_{ij}, \gamma'_{ij}\}$. For each such p a new constraint $x_p \leq x_q$ can be identified via Lemma 79. So if there are many p's satisfying this we will be able to obtain many new constraints and hence new valid arcs (p, q).

On the other hand, the bound in Lemma 85 says that our point in the base polyhedron is small in *absolute* value. This is actually stronger than what we need in Lemma 79 which requires only its positive entries to be "small". However as we saw in Lemma 80 we can generate a constraint of the form $x_q \leq x_p$ whenever $\texttt{lower}(p)$ is sufficiently negative.

Using this idea, we divide V into different buckets according to $\texttt{upper}(p)$ and $\texttt{lower}(p)$. This will allow us to get a speedup for two reasons.

First, bucketing allows us to disregard unimportant elements of V during certain executions of our cutting plane method. If both $\texttt{upper}(i)$ and $\texttt{lower}(i)$ are small in absolute value, then i is essentially negligible because for a separating hyperplane $\vec{h}^T \vec{x} \leq \hat{f}(\bar{x})$, any $h_i \in [\texttt{lower}(i), \texttt{upper}(i)]$ small in absolute value would not really make a difference. We can then run our cutting plane algorithm only on those non-negligible i's, thereby reducing our time complexity. Of course, whether h_i is small is something relative. This suggests that partitioning the ground set by the relative size of $\texttt{upper}(i)$ and $\texttt{lower}(i)$ is a good idea.

Second, bucketing allows us to ensure that we can always add an arc for many edges simultaneously. Recall that we remarked that all we want is $n^{O(1)}|y_i| \leq \texttt{upper}(p)$ for some $\vec{y}$ in the base polyhedron. This would be sufficient to identify a new valid arc (p, q). Now if the marginal differences $\texttt{upper}(p)$ and $\texttt{upper}(p')$ are close in value, knowing $n^{O(1)}|y_i| \leq \texttt{upper}(p)$ would effectively give us the same for p' for free. This suggests that elements with similar marginal differences should be grouped together.

The remainder of this section simply formalizes these ideas. In Section 15.4.1 we discuss how we partition the ground set V. In Section 15.4.2, we present our cutting plane method on a subset of the coordinates. Then in Section 15.4.3 we show how we find new arcs. Finally, in Section 15.4.4 we put all of this together to achieve our desired running time.

15.4.1 Partitioning Ground Set into Buckets

We partition the ground set V into different buckets according to the values of $\texttt{upper}(i)$ and $\texttt{lower}(i)$. This is reminiscent to Iwata-Orlin's algorithm [60] which considers elements with big $\texttt{upper}(i)$. However they did not need to do bucketing by size or to consider $\texttt{lower}(i)$, whereas these seem necessary for our algorithm.

Let $N = \max_i\{\max\{\texttt{upper}(i), -\texttt{lower}(i)\}\}$ be the largest marginal difference in absolute value. By Lemma (78), $N \geq 0$. We partition our ground set V as follows:

$$B_1 = \{i : \texttt{upper}(i) \geq N/n^{10} \text{ or } \texttt{lower}(i) \leq -N/n^{10}\}$$

$$\begin{aligned} B_k \quad = \{i \notin B_1 \cup \ldots \cup B_{k-1} : \quad & N/n^{10k} \leq \texttt{upper}(i) < N/n^{10(k-1)} \\ & \text{or } \; -N/n^{10(k-1)} < \texttt{lower}(i) \leq -N/n^{10k}\}, \quad k \geq 2 \end{aligned}$$

We call B_k *buckets*. Our buckets group elements by the values of $\texttt{upper}(i)$ and $\texttt{lower}(i)$ at $1/n^{10}$ "precision". There are two cases.

- Case 1: the number of buckets is at most $\log n$[7], in which case $\texttt{upper}(i) > N/n^{O(\log n)}$ or $\texttt{lower}(i) < -N/n^{O(\log n)}$ for all i.
- Case 2: there is some k for which $|B_1 \cup \ldots \cup B_k| \geq |B_{k+1}|$.

This is because if there is no such k in Case 2, then by induction each bucket B_{k+1} has at least $2^k|B_1| \geq 2^k$ elements and hence $k \leq \log n$.

Case 1 is easier to handle, and is in fact a special case of Case 2. We first informally sketch the treatment for Case 1 which should shed some light into how we deal with Case 2.

We run Cutting Plane for $O(n \log^2 n)$ iterations (i.e. $\tau = \Theta(\log n)$). By Theorem 82, our feasible region P would be sandwiched by a pair of approximately parallel supporting hyperplanes of width at most $1/n^{\Theta(\log n)}$. Now proceeding as in the last section, we would be able to find some $\vec{y}$ in the base polyhedron and some element p such that $n^{\Theta(\log n)}|y_i| \leq \texttt{upper}(p)$. This gives

$$n^{\Theta(\log n)}|y_i| \leq \frac{\texttt{upper}(p)}{n^{\Theta(\log n)}} \leq \frac{N}{n^{\Theta(\log n)}}.$$

Since $\texttt{upper}(i) > N/n^{\Theta(\log n)}$ or $\texttt{lower}(i) < -N/n^{\Theta(\log n)}$ for all i in Case 1, we can then conclude that some valid arc (i, q) or (q, i) can be added for every i. Thus we add $n/2$ arcs simultaneously in one phase of the algorithm at the expense of blowing up the runtime by $O(\log n)$. This saves a factor of $n/\log n$ from our runtime in the last section, and the amortized cost for an arc would then be $\widetilde{O}(n \cdot \text{EO} + n^2)$.

On the other hand, in Case 2 we have a "trough" at B_{k+1}. Roughly speaking, this trough is useful for acting as a soft boundary between $B_1 \cup \ldots \cup B_k$ and $\bigcup_{l \geq k+2} B_l$. Recall that we are able to "ignore" $\bigcup_{l \geq k+2} B_l$ because their h_i is relatively small in absolute value. In particular, we know that for any $p \in B_1 \cup \ldots \cup B_k$ and $i \in B_l$, where $l \geq k + 2$,

$$\max\{\texttt{upper}(p), -\texttt{lower}(p)\} \geq n^{10} \max\{\texttt{upper}(i), -\texttt{lower}(i)\}.$$

This is possible because B_{k+1}, which is sandwiched in between, acts like a shield preventing B_l to "mess with" $B_1 \cup \ldots \cup B_k$. This property comes at the expense of sacrificing B_{k+1} which must confront B_l.

Furthermore, we require that $|B_1 \cup \ldots \cup B_k| \geq |B_{k+1}|$, and run Cutting Plane on $B = (B_1 \cup \ldots \cup B_k) \cup B_{k+1}$. If $|B_{k+1}| \gg |B_1 \cup \ldots \cup B_k|$, our effort would mostly be wasted on B_{k+1} which is sacrificed, and the amortized time complexity for $B_1 \cup \ldots \cup B_k$ would then be large.

Before discussing the algorithm for Case 2, we need some preparatory work.

[7] More precisely, $B_k = \emptyset$ for $k > \lceil \log n \rceil$.

15.4.2 Separating Hyperplane: Project and Lift

Our speedup is achieved by running our cutting plane method on the projection of our feasible region onto $B := (B_1 \cup \cdots \cup B_k) \cup B_{k+1}$. More precisely, we start by running our cutting plane on $P^B = \{\vec{x} \in \mathbb{R}^B : \exists \vec{x}' \in \mathbb{R}^{\bar{B}} \text{ s.t. } (\vec{x}, \vec{x}') \text{ satisfies (15.1)}\}$, which has a lower dimension. However, to do this, we need to specify a separation oracle for P^B. Here we make one of the most natural choices.

We begin by making an apparently immaterial change to our set of arcs A. Let us take the *transitive closure* of A by adding the arc (i, j) whenever there is a path from i to j. Clearly this would not change our ring family as a path from i to j implies $j \in R(i)$. Roughly speaking, we do this to handle pathological cases such as $(i, k), (k, j) \in A, (i, j) \notin A$ and $i, j \in B, k \notin B$. Without introducing the arc (i, j), we risk confusing a solution containing i but not j as feasible since we are restricting our attention to B and ignoring $k \notin B$.

Definition 88. Given a digraph $D = (V, A)$, the transitive closure of A is the set of arcs (i, j) for which there is a directed path from i to j. We say that A is *complete* if it is equal to its transitive closure.

Given $\bar{x} \in [0, 1]^B$, we define the completion of $\bar{x}$ with respect to A as follows.

Definition 89. Given $\bar{x} \in [0, 1]^B$ and a set of arcs A, $x^{\mathcal{C}} \in [0, 1]^n$ is a completion of $\bar{x}$ if $x^{\mathcal{C}}_B = \bar{x}$ and $x^{\mathcal{C}}_i \le x^{\mathcal{C}}_j$ for every $(i, j) \in A$. Here $x^{\mathcal{C}}_B$ denotes the restriction of $x^{\mathcal{C}}$ to B.

Lemma 90. *Given $\bar{x} \in [0, 1]^B$ and a complete set of arcs A, there is a completion of $\bar{x}$ if $\bar{x}_i \le \bar{x}_j$ for every $(i, j) \in A \cap (B \times B)$. Moreover, it can be computed in $O(n^2)$ time.*

Proof. We set $x^{\mathcal{C}}_B = \bar{x}$. For $i \notin B$, we set

$$x^{\mathcal{C}}_i = \begin{cases} 1 & \text{if } \nexists j \in B \text{ s.t. } (i, j) \in A \\ \min_{(i,j)\in A, j\in B} x^{\mathcal{C}}_j & \text{otherwise} \end{cases}$$

One may verify that $x^{\mathcal{C}}$ satisfies our requirement as A is complete. Computing each $x^{\mathcal{C}}_i$ takes $O(n)$ time. Since $|V \backslash B| = |\bar{B}| \le n$, computing the whole $x^{\mathcal{C}}$ takes $O(n^2)$ time. □

This notion of completion is needed since our original separation oracle requires a full dimensional input $\bar{x}$. Now that $\bar{x} \in \mathbb{R}^B$, we need a way of extending it to $\mathbb{R}^n$ while retaining the crucial property that $\vec{h}$ is consistent with every arc in A.

Note that the runtime is still $O(n \cdot \mathrm{EO} + n^2 \log^{O(1)} n)$ as $x^{\mathcal{C}}$ can be computed in $O(n^2)$ time by the last lemma.

We reckon that the hyperplane $\vec{h}^T_B \vec{x}_B \le \sum_{i\in B} h_i \bar{x}_i$ returned by the oracle is *not* a valid separating hyperplane (i.e. it may cut out the minimizers). Nevertheless, we will show that it is a decent "proxy" to the true separating hyperplane $\vec{h}^T \vec{x} \le \hat{f}(x^{\mathcal{C}}) = \sum_{i\in V} h_i x^{\mathcal{C}}_i$ and is good enough to serve our purpose of sandwiching the remaining feasible region in a small strip. To get a glimpse, note that the terms missing $\vec{h}^T_B \vec{x}_B \le \sum_{i\in B} h_i \bar{x}_i$ all involve h_i for $i \notin B$, which is "negligible" compared to $B_1 \cup \cdots \cup B_k$.

One may try to make $\vec{h}^T_B \vec{x}_B \le \sum_{i\in B} h_i \bar{x}_i$ valid, say, by $\vec{h}^T_B \vec{x}_B \le \sum_{i\in B} h_i \bar{x}_i + \sum_{i\notin B} |h_i|$. The problem is that such hyperplanes would not be separating for $\bar{x}$ anymore as $\vec{h}^T_B \bar{x} = \sum_{i\in B} h_i \bar{x}_i < \sum_{i\in B} h_i \bar{x}_i + \sum_{i\notin B} |h_i|$. Consequently, we lose the width (or volume) guarantee of our cutting plane algorithm. Although this seems problematic, it is actually still possible to show a guarantee sufficient for our purpose as $\sum_{i\notin B} |h_i|$ is relatively small. We leave it as a nontrivial exercise to interested readers.

Algorithm 6: Projected Separation Oracle

Input: $\bar{x} \in \mathbb{R}^B$ and a complete set of arcs A
if $\bar{x}_i < 0$ *for some* $i \in B$ **then**
  Output: $x_i \geq 0$
else if $\bar{x}_j > 1$ *for some* $j \in B$ **then**
  Output: $x_j \leq 1$
else if $\bar{x}_i > \bar{x}_j$ *for some* $(i,j) \in A \cap B^2$ **then**
  Output: $x_i \leq x_j$
else
  Let $x^{\mathcal{C}} \in \mathbb{R}^n$ be a completion of $\bar{x}$
  Let $i_1, \ldots, i_n$ be a permutation of V such that $x^{\mathcal{C}}_{i_1} \geq \ldots \geq x^{\mathcal{C}}_{i_n}$ and for all $(i,j) \in A$, j precedes i in $i_1, \ldots, i_n$.
  Output: $\vec{h}_B^T \vec{x}_B = \sum_{i \in B} h_i x_i \leq \sum_{i \in B} h_i \bar{x}_i$, where $\vec{h}$ is the BFS defined by the permutation $i_1, \ldots, i_n$.

In conclusion, it seems that one cannot have the best of both worlds: the hyperplane returned by the oracle cannot be simultaneously valid and separating.

Algorithm

We take k to be the first for which $|B_1 \cup \ldots \cup B_k| \geq |B_{k+1}|$, i.e. $|B_1 \cup \ldots \cup B_l| < |B_{l+1}|$ for $l \leq k-1$. Thus $k \leq \log n$. Let $b = |B|$, and so $|B_1 \cup \cdots \cup B_k| \geq b/2$. Case 1 is a special case by taking $B = V$.

Our algorithm is summarized below. Here A is always complete as A is replaced its transitive closure whenever a new valid arc is added.

1. Run Cutting Plane on $P^B = \{x \in \mathbb{R}^B : \exists x' \in \mathbb{R}^{\bar{B}} \text{ s.t. } (x, x') \text{ satisfies (15.1)}\}$ with the new projected separation oracle.
2. Identify a pair of "narrow" approximately parallel supporting hyperplanes.
3. Deduce from the hyperplanes certain new constraints of the forms $x_i = 0, x_j = 1, x_i = x_j$ or $x_i \leq x_j$ by lifting separating hyperplanes back to $\mathbb{R}^n$
4. Consolidate A and f. If some $x_i \leq x_j$ added, replace A by its transitive closure.
5. Repeat Step 1 with updated A and f. (Any previously found separating hyperplanes are discarded.)

The minimizer can be constructed by unraveling the recursion.

First of all, to be able to run Cutting Plane on P^B we must come up with a polyhedral description of P^B which consists of just the constraints involving B. This is shown in the next lemma.

Lemma 91. *Let* $P^B = \{\vec{x} \in \mathbb{R}^B : \exists \vec{x}' \in \mathbb{R}^{\bar{B}} \text{ s.t. } (\vec{x}, \vec{x}') \text{ satisfies (15.1)}\}$. *Then*

$$P^B = \{\vec{x} \in \mathbb{R}^B : 0 \leq \vec{x} \leq 1, x_i \leq x_j \forall (i,j) \in A \cap (B \times B)\}$$

Proof. It is clear that $P^B \subseteq \{\vec{x} \in \mathbb{R}^B : 0 \leq \vec{x} \leq 1, x_i \leq x_j \forall (i,j) \in A \cap (B \times B)\}$ as the constraints $0 \leq x \leq 1, x_i \leq x_j \forall (i,j) \in A \cap (B \times B)$ all appear in (15.1).

Conversely, for any $\vec{x} \in \mathbb{R}^B$ satisfying $0 \leq \vec{x} \leq 1, x_i \leq x_j \forall (i,j) \in A \cap (B \times B)$, we know there is some completion $x^{\mathcal{C}}$ of $\vec{x}$ by Lemma 90 as A is complete. Now $x^{\mathcal{C}}$ satisfies (15.1) by definition, and hence $\vec{x} \in P^B$. □

The only place where we have really changed the algorithm is Step (3).

15.4.3 Deducing New Constraints $x_i = 0$, $x_j = 1$, $x_i = x_j$ or $x_i \le x_j$

Our method will deduce one of the following:

- $x_i = 0$, $x_j = 1$ or $x_i = x_j$
- for each $p \in B_1 \cup \cdots \cup B_k$, $x_p \le x_q$ for some $q \notin R(p)$ or $x_p \ge x_q$ for some $q \notin Q(p)$

Our argument is very similar to the last section's. Roughly speaking, it is the same argument but with "noise" introduced by $i \notin B$. We use extensively the notations from the last section.

Our main tool is again Theorem 82. Note that n should be replaced by b in the Theorem statement. We invoke it with $\tau = k \log_b n = O(\log^2 n)$ (using $k \le \log n$) to get a width of $1/b^{\Theta(\tau)} = 1/n^{\Theta(k)}$. This takes time at most $O(bn \log^2 n \cdot \mathrm{EO} + bn^2 \log^{O(1)} n)$. Again, this is intuitively clear as we run it for $O(kb \log n)$ iterations, each of which takes time $O(n \cdot \mathrm{EO} + n^2 \log^{O(1)} n)$.

After each phase of (roughly $O(kb \log n)$ iterations) of Cutting Plane, P^B is sandwiched between a pair of approximately parallel supporting hyperplanes F and F' which have width $1/n^{20k}$. Let F and F' be

$$\vec{c}^T \vec{x}_B = \sum_{i \in B} c_i x_i \le M, \quad \vec{c}'^T \vec{x}_B = \sum_{i \in B} c'_i x_i \le M',$$

such that

$$|M + M'|,\ ||\vec{c} + \vec{c}'||_2 \le \mathtt{gap}, \quad \text{where } \mathtt{gap} = \frac{1}{n^{20k}} \min\{||\vec{c}||_2, ||\vec{c}'||_2\}.$$

The rest of this section presents an execution of the ideas discussed above. All of our work is basically geared towards bringing the amortized cost for identifying a valid arc down to $\widetilde{O}(n \cdot \mathrm{EO} + n^2)$. Again, we can write these two constraints as a nonnegative combination. Here $\bar{x}_h^{\mathcal{C}}$ is the completion of the point $\bar{x}_h$ used to construct $\vec{h}_B^T \vec{x}_B \le \vec{h}_B^T \left(\bar{x}_h^{\mathcal{C}}\right)_B$. (Recall that $\left(\bar{x}_h^{\mathcal{C}}\right)_B$ is the restriction of $\bar{x}_h^{\mathcal{C}}$ to B.)

$$\vec{c}^T \vec{x}_B = -\sum_{i \in B} \alpha_i x_i + \sum_{j \in B} \beta_j x_j + \sum_{(i,j) \in A \cap B^2} \gamma_{ij}(x_i - x_j) + \sum_{h \in H} \lambda_h \vec{h}_B^T \vec{x}_B \quad \text{and} \quad M = \sum_{j \in B} \beta_j + \sum_{h \in H} \lambda_h \vec{h}_B^T \left(\bar{x}_h^{\mathcal{C}}\right)_B.$$

$$\vec{c}'^T \vec{x}_B = -\sum_{i \in B} \alpha'_i x_i + \sum_{j \in B} \beta'_j x_j + \sum_{(i,j) \in A \cap B^2} \gamma'_{ij}(x_i - x_j) + \sum_{h \in H} \lambda'_h \vec{h}_B^T \vec{x}_B \quad \text{and} \quad M' = \sum_{j \in B} \beta'_j + \sum_{h \in H} \lambda'_h \vec{h}_B^T \left(\bar{x}_h^{\mathcal{C}}\right)_B.$$

As we have discussed, the problem is that the separating hyperplanes $\vec{h}_B^T \vec{x}_B \le \vec{h}_B^T \left(\bar{x}_h^{\mathcal{C}}\right)_B$ are not actually valid. We can, however, recover their valid counterpart by lifting them back to $\vec{h}^T \vec{x} \le \vec{h}^T \bar{x}_h^{\mathcal{C}}$. The hope is that $\vec{h}_B^T \vec{x}_B \le \vec{h}_B^T \left(\bar{x}_h^{\mathcal{C}}\right)_B$ and $\vec{h}^T \vec{x} \le \vec{h}^T \bar{x}_h^{\mathcal{C}}$ are not too different so that the arguments will still go through. We show that this is indeed the case.

Again, we scale $c, c', \alpha, \alpha', \beta, \beta', \gamma, \gamma', \lambda, \lambda'$ so that

$$\sum_{h \in H} (\lambda_h + \lambda'_h) = 1.$$

By adding all the constituent separating hyperplane inequalities, we get

$$\sum_{h\in H}\lambda_h\vec{h}^T\vec{x}+\sum_{h\in H}\lambda'_h\vec{h}^T\vec{x}\le\sum_{h\in H}\lambda_h\vec{h}^T\bar{x}_h^{\mathcal{C}}+\sum_{h\in H}\lambda'_h\vec{h}^T\bar{x}_h^{\mathcal{C}}$$

Let

$$LHS\stackrel{\text{def}}{=}\sum\alpha_ix_i+\sum\alpha'_ix_i-\sum\beta_jx_j-\sum\beta'_jx_j+\sum\gamma_{ij}(x_j-x_i)+\sum\gamma'_{ij}(x_j-x_i).$$

Here we know that

$$\sum_{h\in H}\lambda_h\vec{h}^T\vec{x}+\sum_{h\in H}\lambda'_h\vec{h}^T\vec{x}=LHS+(\vec{c}+\vec{c}')^T\vec{x}_B+\sum_{h\in H}\lambda_h\vec{h}_{\bar{B}}^T\vec{x}_{\bar{B}}+\sum_{h\in H}\lambda'_h\vec{h}_{\bar{B}}^T\vec{x}_{\bar{B}}$$

$$\sum_{h\in H}\lambda_h\vec{h}^T\bar{x}_h^{\mathcal{C}}+\sum_{h\in H}\lambda'_h\vec{h}^T\bar{x}_h^{\mathcal{C}}=(M+M')+\sum_{h\in H}\lambda_h\vec{h}_{\bar{B}}^T\left(\bar{x}_h^{\mathcal{C}}\right)_{\bar{B}}+\sum_{h\in H}\lambda'_h\vec{h}_{\bar{B}}^T\left(\bar{x}_h^{\mathcal{C}}\right)_{\bar{B}}-\sum\beta_j-\sum\beta'_j$$

Combining all yields

$$LHS+(\vec{c}+\vec{c}')^T\vec{x}_B+\sum_{h\in H}\lambda_h\vec{h}_{\bar{B}}^T\vec{x}_{\bar{B}}+\sum_{h\in H}\lambda'_h\vec{h}_{\bar{B}}^T\vec{x}_{\bar{B}}\le(M+M')+\sum_{h\in H}\lambda_h\vec{h}_{\bar{B}}^T\left(\bar{x}_h^{\mathcal{C}}\right)_{\bar{B}}+\sum_{h\in H}\lambda'_h\vec{h}_{\bar{B}}^T\left(\bar{x}_h^{\mathcal{C}}\right)_{\bar{B}}-\sum\beta_j-\sum\beta'_j$$

Here $(\vec{c}+\vec{c}')^T\vec{x}_B$ can be bounded as before: $(\vec{c}+\vec{c}')^T\vec{x}_B\ge-\sqrt{n}||\vec{c}+\vec{c}'||_2\ge-\sqrt{n}\mathtt{gap}$. Since $M+M'\le\mathtt{gap}$, We then obtain

$$LHS+\sum_{h\in H}\lambda_h\vec{h}_{\bar{B}}^T\vec{x}_{\bar{B}}+\sum_{h\in H}\lambda'_h\vec{h}_{\bar{B}}^T\vec{x}_{\bar{B}}\le2\sqrt{n}\mathtt{gap}+\sum_{h\in H}\lambda_h\vec{h}_{\bar{B}}^T\left(\bar{x}_h^{\mathcal{C}}\right)_{\bar{B}}+\sum_{h\in H}\lambda'_h\vec{h}_{\bar{B}}^T\left(\bar{x}_h^{\mathcal{C}}\right)_{\bar{B}}-\sum\beta_j-\sum\beta'_j$$

We should expect the contribution from $\vec{h}_{\bar{B}}$ to be small as h_i for $i\notin B$ is small compared to $B_1\cup\ldots\cup B_k$. We formalize our argument in the next two lemmas.

Lemma 92. *We have* $\sum_{h\in H}\lambda_h\vec{h}_{\bar{B}}^T\left(\bar{x}_h^{\mathcal{C}}\right)_{\bar{B}}+\sum_{h\in H}\lambda'_h\vec{h}_{\bar{B}}^T\left(\bar{x}_h^{\mathcal{C}}\right)_{\bar{B}}\le N/n^{10(k+1)-1}$.

Proof. We bound each component of $\sum_{h\in H}\lambda_h\vec{h}_{\bar{B}}^T\left(\bar{x}_h^{\mathcal{C}}\right)_{\bar{B}}+\sum_{h\in H}\lambda'_h\vec{h}_{\bar{B}}^T\left(\bar{x}_h^{\mathcal{C}}\right)_{\bar{B}}$. For $i\in\bar{B}$, we have $\mathtt{upper}(i)\le N/n^{10(k+1)}$. By Lemma 78 $h_i\le\mathtt{upper}(i)$. Therefore,

$$\sum_{h\in H}\lambda_h\vec{h}_i^T\left(\bar{x}_h^{\mathcal{C}}\right)_i+\sum_{h\in H}\lambda'_h\vec{h}_i^T\left(\bar{x}_h^{\mathcal{C}}\right)_i\quad\le\quad\left(\sum_{h\in H}\lambda_h+\sum_{h\in H}\lambda'_h\right)N/n^{10(k+1)}=N/n^{10(k+1)}.$$

Our result then follows since

$$\sum_{h\in H}\lambda_h\vec{h}_{\bar{B}}^T\left(\bar{x}_h^{\mathcal{C}}\right)_{\bar{B}}+\sum_{h\in H}\lambda'_h\vec{h}_{\bar{B}}^T\left(\bar{x}_h^{\mathcal{C}}\right)_{\bar{B}}=\sum_{i\in\bar{B}}\left(\sum_{h\in H}\lambda_h\vec{h}_i^T\left(\bar{x}_h^{\mathcal{C}}\right)_i+\sum_{h\in H}\lambda'_h\vec{h}_i^T\left(\bar{x}_h^{\mathcal{C}}\right)_i\right).$$

□

Lemma 93. *We have* $\sum_{h\in H}\lambda_h\vec{h}_{\bar{B}}^T\vec{x}_{\bar{B}}+\sum_{h\in H}\lambda'_h\vec{h}_{\bar{B}}^T\vec{x}_{\bar{B}}\ge-N/n^{10(k+1)-1}$.

Proof. The proof is almost identical to the last lemma except that we use $h_i\ge\mathtt{lower}(i)$ instead of $h_i\le\mathtt{upper}(i)$, and $\mathtt{lower}(i)\ge-N/n^{10(k+1)}$. □

The two lemmas above imply that

$$LHS\le2\sqrt{n}\mathtt{gap}-\sum\beta_j-\sum\beta_j+2N/n^{10(k+1)-1}=\mathtt{gap}'-\sum\beta_j-\sum\beta_j$$

where $\mathtt{gap}'=2\sqrt{n}\mathtt{gap}+2N/n^{10(k+1)-1}$.

Lemma 94. *Suppose x satisfies (15.1) and LHS $\le \mathtt{gap}' - \sum \beta_j - \sum \beta'_j$ with $\alpha_i, \beta_j, \gamma_{ij}, \alpha'_i, \beta'_j, \gamma'_{ij} \ge 0$.*

1. *If $\alpha_i > \mathtt{gap}'$ or $\alpha'_i > \mathtt{gap}'$, then $x_i < 1$.*
2. *If $\beta_j > \mathtt{gap}'$ or $\beta'_j > \mathtt{gap}'$, then $x_j > 0$.*
3. *If $\gamma_{ij} > \mathtt{gap}'$ or $\gamma'_{ij} > \mathtt{gap}'$, then $0 \le x_j - x_i < 1$.*

Proof. The proof is exactly the same as Lemma 84 with $2\sqrt{n}\mathtt{gap}$ replaced by $\mathtt{gap}'$. □

From now on we may assume that

$$\max\{\alpha_i, \alpha'_i, \beta_j, \beta'_j, \gamma_{ij}, \gamma'_{ij}\} \le \mathtt{gap}'. \tag{15.9}$$

Lemma 95. *Let $\vec{y} \stackrel{\text{def}}{=} \sum_{h\in H} \lambda_h \vec{h}$ and $\vec{y}' \stackrel{\text{def}}{=} \sum_{h\in H} \lambda'_h \vec{h}$ and let $p \in \arg\max_{l\in B}\{\max\{|y_l|, |y'_l|\}$ then*

$$N \ge n^{10k+6} \left\|\vec{y}_B + \vec{y}'_B\right\|_\infty$$

assuming (15.9).

Proof. Recall that $\left\|\vec{c}+\vec{c}'\right\|_2 \le \mathtt{gap} < \mathtt{gap}'$ where $\mathtt{gap} = \frac{1}{n^{20k}} \min\{||\vec{c}||_2, ||\vec{c}'||_2\}$ and $\mathtt{gap}' = 2\sqrt{n}\mathtt{gap} + 2N/n^{10(k+1)-1}$. Now there are two cases.

Case 1: $2\sqrt{n}\mathtt{gap} \ge 2N/n^{10(k+1)-1}$. Then $\mathtt{gap}' \le 4\sqrt{n}\mathtt{gap}$ and we follow the same proof of Lemma 85. We have

$$\vec{c} = \vec{y}_B - \sum_i \alpha_i \vec{\mathbb{1}}_i + \sum_j \beta_j \vec{\mathbb{1}}_j + \sum_{(i,j)} \gamma_{ij}(\vec{\mathbb{1}}_i - \vec{\mathbb{1}}_j) \quad \text{and } \vec{c}' = \vec{y}'_B - \sum_i \alpha'_i \vec{\mathbb{1}}_i + \sum_j \beta'_j \vec{\mathbb{1}}_j + \sum_{(i,j)} \gamma'_{ij}(\vec{\mathbb{1}}_i - \vec{\mathbb{1}}_j)\,.$$

By (15.9) we know that $\left\|\vec{c} - \vec{y}_B\right\|_2 \le 4n^2\mathtt{gap}' \le \frac{1}{n^{17k}}\left\|\vec{c}\right\|_2$ and $\left\|\vec{c}' - \vec{y}'_B\right\|_2 \le 4n^2\mathtt{gap}' \le \frac{1}{n^{17k}}\left\|\vec{c}\right\|_2$. Consequently, by the triangle inequality we have that

$$\left\|\vec{y}_B + \vec{y}'_B\right\|_2 \le \left\|\vec{c}+\vec{c}'\right\|_2 + \left\|\vec{c} - \vec{y}_B\right\|_2 + \left\|\vec{c}' - \vec{y}'_B\right\|_2 \le 9n^2\mathtt{gap}'$$

and

$$\left\|\vec{c}\right\|_2 \le \left\|\vec{c} - \vec{y}_B\right\|_2 + \left\|\vec{y}_B\right\|_2 \le \frac{1}{n^{17k}}\left\|\vec{c}\right\|_2 + \left\|\vec{y}_B\right\|_2 \quad \Rightarrow \quad \left\|\vec{c}\right\|_2 \le 2\left\|\vec{y}_B\right\|_2$$

Similarly, we have that $\left\|\vec{c}'\right\|_2 \le 2\left\|\vec{y}'_B\right\|_2$. Consequently since $\mathtt{gap}' \le \frac{1}{n^{19k}} \min\{||\vec{c}||_2, ||\vec{c}'||_2\}$, we have that

$$\left\|\vec{y}_B + \vec{y}'_B\right\|_2 \le \frac{18}{n^{17k}} \min\left\{\left\|\vec{y}_B\right\|_2, \left\|\vec{y}'_B\right\|_2\right\}$$

and thus, invoking Lemma 81 yields $N \ge \mathtt{upper}(p) \ge n^{16k}\left\|\vec{y}_B + \vec{y}'_B\right\|_\infty$, as desired.

Case 2: $2\sqrt{n}\mathtt{gap} < 2N/n^{10(k+1)-1}$. Then for any $i \in B$, $|c_i + c'_i| \le ||\vec{c} + \vec{c}'||_2 \le \mathtt{gap} < 2N/n^{10(k+1)-1}$. Since

$$\vec{y}_B + \vec{y}'_B = (\vec{c}+\vec{c}') + \sum_i \alpha_i \vec{\mathbb{1}}_i - \sum_j \beta_j \vec{\mathbb{1}}_j - \sum_{(i,j)} \gamma_{ij}(\vec{\mathbb{1}}_i - \vec{\mathbb{1}}_j) + \sum_i \alpha'_i \vec{\mathbb{1}}_i - \sum_j \beta'_j \vec{\mathbb{1}}_j - \sum_{(i,j)} \gamma'_{ij}(\vec{\mathbb{1}}_i - \vec{\mathbb{1}}_j)$$

we have

$$\left\|\vec{y}_B + \vec{y}'_B\right\|_\infty \le 2N/n^{10(k+1)-1} + 2n^{1.5}\mathtt{gap}' \le N/n^{10k+7}.$$

□

Corollary 96. *Let P be the feasible region after running Cutting Plane on (15.1) with the projected separation oracle. Then one of the following holds:*

1. *We found a BFS $\vec{h}$ with $\vec{h}_B = 0$.*
2. *The integral points of P all lie on some hyperplane $x_i = 0, x_j = 1$ or $x_i = x_j$.*
3. *Let H be the collection of BFS's $\vec{h}$ used to construct our separating hyperplanes for P. Then there is a convex combination $\vec{y}$ of H such that for $p \in B_1 \cup \cdots \cup B_k$, we have $n^4|y_i| < \texttt{upper}(p)$ or $\texttt{lower}(p) < -n^4|y_i|$ for all i.*

Proof. As mentioned before, (1) happens if some separating hyperplane satisfies $\vec{h}_B = 0$ when running cutting plane on the non-negligible coordinates. We have (2) if some condition in Lemma 94 holds. Otherwise, we claim $y = \sum_{\boldsymbol{h}} \lambda_{\boldsymbol{h}} \boldsymbol{h} + \sum_{\boldsymbol{h}} \lambda'_{\boldsymbol{h}} \boldsymbol{h}$ is a candidate for Case 3. y is a convex combination of BFS and by Lemma 95, for the big elements $i \in B$ we have

$$|y_i| \le N/n^{10k+6} \le \frac{1}{n^4} \max\{\texttt{upper}(p), -\texttt{lower}(p)\}.$$

where the last inequality holds since for $p \in B_1 \cup \cdots \cup B_k$, $\max\{\texttt{upper}(p), -\texttt{lower}(p)\} \ge N/n^{10k}$.

On the other hand, for the small elements $i \notin B$, $|y_i| \le N/n^{10(k+1)} \le \frac{1}{n^4} \max\{\texttt{upper}(p), -\texttt{lower}(p)\}$ as desired. □

The gap is then smaller enough to add an arc for each $p \in B_1 \cup \cdots \cup B_k$ by Lemmas 79 and 80. Therefore we can add a total of $|B_1 \cup \cdots \cup B_k|/2 \ge b/4$ arcs with roughly $O(kb \log n) = \widetilde{O}(b)$ iterations of Cutting Plane, each of which takes $\widetilde{O}(n \cdot \mathrm{EO} + n^2)$. That is, the amortized cost for each arc is $\widetilde{O}(n \cdot \mathrm{EO} + n^2)$. We give a more formal time analysis in below but it should be somewhat clear why we have the desired time complexity.

Lemma 97. *Suppose there is a convex combination $\vec{y}$ of H such that for $p \in B_1 \cup \cdots \cup B_k$, we have $n^4|y_i| < \texttt{upper}(p)$ or $\texttt{lower}(p) < -n^4|y_i|$ for all i. Then we can identify at least $b/4$ new valid arcs.*

Proof. We have $|H| = O(n)$ since H is the set of BFS's used for the constraints of P which has $O(n)$ constraints. By Lemmas 79 and 80, for $p \in B_1 \cup \cdots \cup B_k$ we can add a new valid arc (p, q) or (q, p). However note that a new arc (p_1, p_2) may added twice by both p_1 and p_2. Therefore the total number of new arcs is only at least $|B_1 \cup \cdots \cup B_k|/2 \ge b/4$. □

15.4.4 Running Time

Not much changes to the previous runtime analysis are needed. To avoid repetition, various details already present in the corresponding part of the last section are omitted. Recall $k \le \log n$, and of course, $b \le n$.

For each (roughly) $O(kb \log n)$ iterations of Cutting Plane we either get $x_i = 0$,$x_i = 1$,$x_i = x_j$ or $b/4$ $x_i \le x_j$'s. The former can happen at most n times while in the latter case, the amortized cost of each arc is $O(k \log n)$ iterations of Cutting Plane. In the worst case the overall number of iterations required is $\widetilde{O}(n^2)$. Thus our algorithm has a runtime of $\widetilde{O}(n^3 \cdot \mathrm{EO} + n^4)$ since each iteration is $\widetilde{O}(n \cdot \mathrm{EO} + n^2)$ as shown below.

Theorem 98. *Our algorithm runs in time $O(n^3 \log^2 n \cdot EO + n^4 \log^{O(1)} n)$.*

Proof. We use Corollary 96. First we note that Case 1 can actually be integrated into Case 3 since $\max\{\texttt{upper}(p), -\texttt{lower}(p)\} \geq N/n^{10k} = n^{10}N/n^{10(k+1)} \geq h_i$ for $i \notin B$.

As we have argued in the beginning of the last section, Theorem 82 with $\tau = k \log_b n$ implies that the runtime for each phase is $O(bn \log^2 n \cdot \mathrm{EO} + bn^2 \log^{O(1)} n)$. In each phase we either get $x_i = 0$, $x_i = 1$, $x_i = x_j$ (Case 2) or $b/4$ $x_i \leq x_j$'s (Case 3), the latter of which follows from Corollary 96 and Lemma 97.

Case 2 can only happen n times. Thus the total cost is at most $O(n^3 \log^2 n \cdot \mathrm{EO} + n^4 \log^{O(1)} n)$. The overhead cost is also small. Similar to before, given F and F' represented as a nonnegative combination of facets, we can check for the conditions in Lemma 94 in $O(n)$ time as there are only this many facets of P. This settles Case 2.

For case 3 the amortized cost for each arc is $O(n \log^2 n \cdot \mathrm{EO} + n^2 \log^{O(1)} n)$. Our desired runtime follows since there are only $O(n^2)$ arcs to add. Unlike Case 2 some extra care is needed to handle the overhead cost. The time needed to deduce a new arc (applying Lemmas 79 and 80 to $\vec{y}$ and $p \in B_1 \cup \cdots \cup B_k$) is still $O(n \cdot \mathrm{EO} + n^2)$. But as soon as we get a new arc, we must update A to be its transitive closure so that it is still complete. Given A complete and a new arc $(p, q) \notin A$, we can simply add the arcs from the ancestors of p to q and from p to the descendants of q. There are at most $O(n)$ arcs to add so this takes time $O(n^2)$ per arc, which is okay. □

16 Discussion and Comparison with Previous Algorithms

We compare and contrast our algorithms with the previous ones. We focus primarily on strongly polynomial time algorithms.

Convex combination of BFS's

All of the previous algorithms maintain a convex combination of BFS's and iteratively improve over it to get a better primal solution. In particular, the new BFS's used are typically obtained by making local changes to existing ones. Our algorithms, on the other hand, considers the geometry of the existing BFS's. The weighted "influences"[8] then aggregately govern the choice of the next BFS. We believe that this is the main driving force for the speedup of our algorithms.

Scaling schemes

Many algorithms for combinatorial problems are explicitly or implicitly scaling a potential function or a parameter. In this paper, our algorithms in some sense aim to minimize the volume of the feasible region. Scaling schemes for different potential functions and parameters were also designed in previous works [56, 54, 60, 53]. All of these functions and parameters have an explict form. On the contrary, our potential function is somewhat unusual in the sense that it has no closed form.

Deducing new constraints

As mentioned in the main text, our algorithms share the same skeleton and tools for deducing new constraints with [56, 54, 60, 53]. Nevertheless, there are differences in the way these tools are employed. Our algorithms proceed by invoking them in a geometric manner, whereas previous algorithms were mostly combinatorial.

Big elements and bucketing

Our bucketing idea has roots in Iwata-Orlin's algorithm [60] but is much more sophisticated. For instance, it is sufficient for their algorithm to consider only big elements, i.e. $\texttt{upper}(i) \geq N/n^{O(1)}$. Our algorithm, on the other hand, must carefully group elements by the size of both $\texttt{upper}(i)$ and

[8] In the terminology of Part I, these weighted influences are the leverage scores.

`lower`(i). The speedup appears impossible without these new ideas. We do however note that it is unfair to expect such a sophisticated scheme in Iwata-Orlin's algorithm as it would not lead to a speedup. In other words, their method is fully sufficient for their purposes, and the simplicity in their case is a virtue rather than a shortcoming.

16.1 Open Problems

One natural open problem is improving our weakly polynomial algorithm to $O(n^2 \log M \cdot \text{EO} + n^3 \log^{O(1)} n \cdot \log M)$ time. Our application of center of mass to SFM demonstrates that it should be possible.

For strongly polynomial algorithms, the existential result of Theorem 71 shows that SFM can be solved with $O(n^3 \log n \cdot \text{EO})$ oracle calls. Unfortunately, our algorithm incurs an overhead of $\log n$ as there can be as many as $\log n$ buckets each time. One may try to remove this $\log n$ overhead by designing a better bucketing scheme or arguing that more arcs can be added.

The other $\log n$ overhead seem much trickier to remove. Our method currently makes crucial use of the tools developed by [56], where the $\log n$ factors in the runtime seem inevitable. We suspect that our algorithm may have an analogue similar to [93, 90], which do not carry any $\log n$ overhead in the running time.

Perhaps an even more interesting open problem is whether our algorithm is optimal (up to polylogarithmic factors). There are grounds for optimism. So far the best way of *certifying* the optimality of a given solution $S \subseteq V$ is to employ duality and express some optimal solution to the base polyhedron as a convex combination of $n+1$ BFS's. This already takes n^2 oracle calls as each BFS requires n. Thus one would expect the optimal number of oracle calls needed for SFM to be at least n^2. Our bound is not too far off from it, and anything strictly between n^2 and n^3 seems instinctively unnatural.

Acknowledgments

We thank Matt Weinberg for insightful comments about submodular minimization and minimizing the intersection of convex sets that were deeply influential to our work. We thank Yan Kit Chim, Stefanie Jegelka, Jonathan A. Kelner, Robert Kleinberg, Pak-Hin Lee, Christos Papadimitriou, and Chit Yu Ng for many helpful conversations. We thank Chien-Chung Huang for pointing out a typo in an earlier draft of this paper. This work was partially supported by NSF awards 0843915 and 1111109, NSF grants CCF0964033 and CCF1408635, Templeton Foundation grant 3966, NSF Graduate Research Fellowship (grant no. 1122374). Part of this work was done while the first two authors were visiting the Simons Institute for the Theory of Computing, UC Berkeley. Lastly, we would like to thank Vaidya for his beautiful work on his cutting plane method.

References

[1] Dimitris Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *Journal of computer and System Sciences*, 66(4):671–687, 2003.

[2] Martin Aigner and Thomas A Dowling. Matching theory for combinatorial geometries. *Transactions of the American Mathematical Society*, 158(1):231–245, 1971.

[3] Zeyuan Allen-Zhu, Yin Tat Lee, and Lorenzo Orecchia. Using optimization to obtain a width-independent, parallel, simpler, and faster positive sdp solver. *arXiv preprint arXiv:1507.02259*, 2015.

[4] Kurt M Anstreicher. Large step volumetric potential reduction algorithms for linear programming. *Annals of Operations Research*, 62(1):521–538, 1996.

[5] Kurt M Anstreicher. On vaidya's volumetric cutting plane method for convex programming. *Mathematics of Operations Research*, 22(1):63–89, 1997.

[6] Kurt M Anstreicher. Towards a practical volumetric cutting plane method for convex programming. *SIAM Journal on Optimization*, 9(1):190–206, 1998.

[7] Kurt M Anstreicher. The volumetric barrier for semidefinite programming. *Mathematics of Operations Research*, 25(3):365–380, 2000.

[8] Sanjeev Arora, Elad Hazan, and Satyen Kale. Fast algorithms for approximate semidefinite programming using the multiplicative weights update method. In *Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on*, pages 339–348. IEEE, 2005.

[9] Sanjeev Arora and Satyen Kale. A combinatorial, primal-dual approach to semidefinite programs. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 227–236. ACM, 2007.

[10] David S Atkinson and Pravin M Vaidya. A cutting plane algorithm for convex programming that uses analytic centers. *Mathematical Programming*, 69(1-3):1–43, 1995.

[11] Francis Bach. Learning with submodular functions: A convex optimization perspective. *Foundations and Trends in Machine Learning*, 2013.

[12] Francisco Barahona and William H Cunningham. A submodular network simplex method. In *Mathematical Programming at Oberwolfach II*, pages 9–31. Springer, 1984.

[13] Dimitris Bertsimas and Santosh Vempala. Solving convex programs by random walks. *Journal of the ACM (JACM)*, 51(4):540–556, 2004.

[14] Carl Brezovec, Gerard Cornuéjols, and Fred Glover. Two algorithms for weighted matroid intersection. *Mathematical Programming*, 36(1):39–53, 1986.

[15] Sébastien Bubeck, Yin Tat Lee, and Mohit Singh. A geometric alternative to nesterov's accelerated gradient descent. *arXiv preprint arXiv:1506.08187*, 2015.

[16] Yang Cai, Constantinos Daskalakis, and S Matthew Weinberg. Optimal multi-dimensional mechanism design: Reducing revenue to welfare maximization. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pages 130–139. IEEE, 2012.

[17] Yang Cai, Constantinos Daskalakis, and S Matthew Weinberg. Reducing revenue to welfare maximization: Approximation algorithms and other generalizations. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 578–595. SIAM, 2013.

[18] Nam-Kee Chung and Dong-Wan Tcha. A dual algorithm for submodular flow problems. *Operations research letters*, 10(8):489–495, 1991.

[19] Michael B. Cohen, Yin Tat Lee, Cameron Musco, Christopher Musco, Richard Peng, and Aaron Sidford. Uniform sampling for matrix approximation. *CoRR*, abs/1408.5099, 2014.

[20] William H Cunningham. On submodular function minimization. *Combinatorica*, 5(3):185–192, 1985.

[21] William H Cunningham. Improved bounds for matroid partition and intersection algorithms. *SIAM Journal on Computing*, 15(4):948–957, 1986.

[22] William H Cunningham and András Frank. A primal-dual algorithm for submodular flows. *Mathematics of Operations Research*, 10(2):251–262, 1985.

[23] Constantinos Daskalakis and S Matthew Weinberg. Bayesian truthful mechanisms for job scheduling from bi-criterion approximation algorithms. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1934–1952. SIAM, 2015.

[24] James Demmel, Ioana Dumitriu, Olga Holtz, and Robert Kleinberg. Fast matrix multiplication is stable. *Numerische Mathematik*, 106(2):199–224, 2007.

[25] EA Dinic. An algorithm for the solution of the max-flow problem with the polynomial estimation. *Doklady Akademii Nauk SSSR*, 194(4):1277–1280, 1970.

[26] Jack Edmonds. Matroid partition. *Mathematics of the Decision Sciences*, 11:335–345, 1968.

[27] Jack Edmonds. Submodular functions, matroids, and certain polyhedra. *Edited by G. Goos, J. Hartmanis, and J. van Leeuwen*, page 11, 1970.

[28] Jack Edmonds. Matroid intersection. *Annals of discrete Mathematics*, 4:39–49, 1979.

[29] Jack Edmonds and Rick Giles. A min-max relation for submodular functions on graphs. *Studies in Integer Programming (PL Hammer, EL Johnson and BH Korte, eds.), Ann. Discrete Math*, 1:185–204, 1977.

[30] Lisa Fleischer and Satoru Iwata. Improved algorithms for submodular function minimization and submodular flow. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 107–116. ACM, 2000.

[31] Lisa Fleischer and Satoru Iwata. A push-relabel framework for submodular function minimization and applications to parametric optimization. *Discrete Applied Mathematics*, 131(2):311–322, 2003.

[32] Lisa Fleischer, Satoru Iwata, and S Thomas McCormick. A faster capacity scaling algorithm for minimum cost submodular flow. *Mathematical Programming*, 92(1):119–139, 2002.

[33] András Frank. A weighted matroid intersection algorithm. *Journal of Algorithms*, 2(4):328–336, 1981.

[34] András Frank and Éva Tardos. An application of simultaneous diophantine approximation in combinatorial optimization. *Combinatorica*, 7(1):49–65, 1987.

[35] S. Fujishige. Algorithms for solving the independent-flow problems. *Journal of the Operations Research Society of Japan*, 1978.

[36] Satoru Fujishige. An out-of-kilter method for submodular flows. *Discrete applied mathematics*, 17(1):3–16, 1987.

[37] Satoru Fujishige and Satoru Iwata. Algorithms for submodular flows. *IEICE TRANSACTIONS on Information and Systems*, 83(3):322–329, 2000.

[38] Satoru Fujishige, Hans Röck, and Uwe Zimmermann. A strongly polynomial algorithm for minimum cost submodular flow problems. *Mathematics of Operations Research*, 14(1):60–69, 1989.

[39] Satoru Fujishige and Zhang Xiaodong. An efficient cost scaling algorithm for the independent assignment problem. *Journal of the Operations Research Society of Japan*, 38(1):124–136, 1995.

[40] Mituhiro Fukuda, Masakazu Kojima, Kazuo Murota, and Kazuhide Nakata. Exploiting sparsity in semidefinite programming via matrix completion i: General framework. *SIAM Journal on Optimization*, 11(3):647–674, 2001.

[41] François Le Gall. Powers of tensors and fast matrix multiplication. *arXiv preprint arXiv:1401.7714*, 2014.

[42] J-L Goffin, Jacek Gondzio, Robert Sarkissian, and J-P Vial. Solving nonlinear multicommodity flow problems by the analytic center cutting plane method. *Mathematical Programming*, 76(1):131–154, 1997.

[43] Jean-Louis Goffin, Zhi-Quan Luo, and Yinyu Ye. Complexity analysis of an interior cutting plane method for convex feasibility problems. *SIAM Journal on Optimization*, 6(3):638–652, 1996.

[44] Jean-Louis Goffin and Jean-Philippe Vial. Shallow, deep and very deep cuts in the analytic center cutting plane method. *Mathematical Programming*, 84(1):89–103, 1999.

[45] Jean-Louis Goffin and Jean-Philippe Vial. Convex nondifferentiable optimization: A survey focused on the analytic center cutting plane method. *Optimization Methods and Software*, 17(5):805–867, 2002.

[46] Andrew V Goldberg and Robert E Tarjan. A new approach to the maximum-flow problem. *Journal of the ACM (JACM)*, 35(4):921–940, 1988.

[47] Andrew V Goldberg and Robert E Tarjan. Finding minimum-cost circulations by successive approximation. *Mathematics of Operations Research*, 15(3):430–466, 1990.

[48] Jacek Gondzio, O Du Merle, Robert Sarkissian, and J-P Vial. Accpmï¿œxa library for convex optimization based on an analytic center cutting plane method. *European Journal of Operational Research*, 94(1):206–211, 1996.

[49] Martin Grötschel, László Lovász, and Alexander Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

[50] Martin Grötschel, László Lovász, and Alexander Schrijver. Geometric algorithms and combinatorial optimization. Springer, 1988.

[51] Christoph Helmberg and Franz Rendl. A spectral bundle method for semidefinite programming. *SIAM Journal on Optimization*, 10(3):673–696, 2000.

[52] Satoru Iwata. A capacity scaling algorithm for convex cost submodular flows. *Mathematical programming*, 76(2):299–308, 1997.

[53] Satoru Iwata. A fully combinatorial algorithm for submodular function minimization. *Journal of Combinatorial Theory, Series B*, 84(2):203–212, 2002.

[54] Satoru Iwata. A faster scaling algorithm for minimizing submodular functions. *SIAM Journal on Computing*, 32(4):833–840, 2003.

[55] Satoru Iwata. Submodular function minimization. *Mathematical Programming*, 112(1):45–64, 2008.

[56] Satoru Iwata, Lisa Fleischer, and Satoru Fujishige. A combinatorial strongly polynomial algorithm for minimizing submodular functions. *Journal of the ACM (JACM)*, 48(4):761–777, 2001.

[57] Satoru Iwata, S Thomas McCormick, and Maiko Shigeno. A faster algorithm for minimum cost submodular flows. In *SODA*, pages 167–174, 1998.

[58] Satoru Iwata, S Thomas McCormick, and Maiko Shigeno. A strongly polynomial cut canceling algorithm for the submodular flow problem. In *Integer Programming and Combinatorial Optimization*, pages 259–272. Springer, 1999.

[59] Satoru Iwata, S Thomas McCormick, and Maiko Shigeno. A fast cost scaling algorithm for submodular flow. *Information Processing Letters*, 74(3):123–128, 2000.

[60] Satoru Iwata and James B Orlin. A simple combinatorial algorithm for submodular function minimization. In *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1230–1237. Society for Industrial and Applied Mathematics, 2009.

[61] Rahul Jain and Penghui Yao. A parallel approximation algorithm for positive semidefinite programming. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 463–471. IEEE, 2011.

[62] Klaus Jansen. Approximate strong separation with application in fractional graph coloring and preemptive scheduling. *Theoretical Computer Science*, 302(1):239–256, 2003.

[63] Ravindran Kannan and Hariharan Narayanan. Random walks on polytopes and an affine interior point method for linear programming. *Mathematics of Operations Research*, 37(1):1–20, 2012.

[64] Richard M Karp and Christos H Papadimitriou. On linear characterizations of combinatorial optimization problems. *SIAM Journal on Computing*, 11(4):620–632, 1982.

[65] Leonid G Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72, 1980.

[66] LG Khachiyan, SP Tarasov, and II Erlikh. The method of inscribed ellipsoids. In *Soviet Math. Dokl*, volume 37, pages 226–230, 1988.

[67] Adam R Klivans and Daniel Spielman. Randomness efficient identity testing of multivariate polynomials. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 216–223. ACM, 2001.

[68] Andreas Krause. http://submodularity.org/.

[69] Kartik Krishnan and John E Mitchell. A unifying framework for several cutting plane methods for semidefinite programming. *Optimization methods and software*, 21(1):57–74, 2006.

[70] Kartik Krishnan and John E Mitchell. Properties of a cutting plane method for semidefinite programming. *Pacific Journal of Optimization*, 8(4):779–802, 2012.

[71] Kartik Krishnan and Tamás Terlaky. Interior point and semidefinite approaches in combinatorial optimization. In *Graph theory and combinatorial optimization*, pages 101–157. Springer, 2005.

[72] Eugene L Lawler. Matroid intersection algorithms. *Mathematical programming*, 9(1):31–56, 1975.

[73] Yin Tat Lee, Satish Rao, and Nikhil Srivastava. A new approach to computing maximum flows using electrical flows. In *The 45th ACM Symposium on Theory of Computing (STOC)*, pages 755–764, 2013.

[74] Yin Tat Lee and Aaron Sidford. Path finding ii: An\˜ o (m sqrt (n)) algorithm for the minimum cost flow problem. *arXiv preprint arXiv:1312.6713*, 2013.

[75] Yin Tat Lee and Aaron Sidford. Path-finding methods for linear programming : Solving linear programs in õ(sqrt(rank)) iterations and faster algorithms for maximum flow. In *55th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2014, 18-21 October, 2014, Philadelphia, PA, USA*, pages 424–433, 2014.

[76] Yin Tat Lee and Aaron Sidford. Efficient inverse maintenance and faster algorithms for linear programming. *arXiv preprint arXiv:1503.01752*, 2015.

[77] A. Yu Levin. On an algorithm for the minimization of convex functions. *Soviet Math. Doklady*, 1965.

[78] Mu Li, Gary L Miller, and Richard Peng. Iterative row sampling. 2012.

[79] László Lovász and Santosh Vempala. Simulated annealing in convex bodies and an $o^*(n^4)$ volume algorithm. *J. Comput. Syst. Sci.*, 72(2):392–417, 2006.

[80] Michael W. Mahoney. Randomized algorithms for matrices and data. *Foundations and Trends in Machine Learning*, 3(2):123–224, 2011.

[81] S McCormick. *Submodular Function Minimization*. 2013.

[82] S Thomas and McCormick. Canceling most helpful total submodular cuts for submodular flow. In *IPCO*, pages 343–353, 1993.

[83] Renato DC Monteiro. First-and second-order methods for semidefinite programming. *Mathematical Programming*, 97(1-2):209–244, 2003.

[84] Kazuhide Nakata, Katsuki Fujisawa, Mituhiro Fukuda, Masakazu Kojima, and Kazuo Murota. Exploiting sparsity in semidefinite programming via matrix completion ii: Implementation and numerical results. *Mathematical Programming*, 95(2):303–327, 2003.

[85] Arkadi Nemirovski. Efficient methods in convex programming. 1994.

[86] D. B. Nemirovsky, A. S., & Yudin. Problem complexity and method efficiency in optimization. 1983.

[87] Yu Nesterov. Complexity estimates of some cutting plane methods based on the analytic barrier. *Mathematical Programming*, 69(1-3):149–176, 1995.

[88] Yu Nesterov and Arkadi Nemirovskiy. *Self-concordant functions and polynomial-time methods in convex programming*. USSR Academy of Sciences, Central Economic & Mathematic Institute, 1989.

[89] Yu Nesterov and A Nemirovsky. Conic formulation of a convex programming problem and duality. *Optimization Methods and Software*, 1(2):95–115, 1992.

[90] James B Orlin. A faster strongly polynomial time algorithm for submodular function minimization. *Mathematical Programming*, 118(2):237–251, 2009.

[91] James B Orlin, John VandeVate, et al. On a" primal" matroid intersection algorithm. 1983.

[92] Srinivasan Ramaswamy and John E Mitchell. A long step cutting plane algorithm that uses the volumetric barrier. *Department of Mathematical Science, RPI, Troy, NY*, 1995.

[93] Alexander Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *Journal of Combinatorial Theory, Series B*, 80(2):346–355, 2000.

[94] Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer, 2003.

[95] Jack Sherman and Winifred J Morrison. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics*, pages 124–127, 1950.

[96] Maiko Shigeno and Satoru Iwata. A dual approximation approach to weighted matroid intersection. *Operations research letters*, 18(3):153–156, 1995.

[97] Naum Z Shor. Cut-off method with space extension in convex programming problems. *Cybernetics and systems analysis*, 13(1):94–96, 1977.

[98] Maurice Sion. On general minimax theorems. *Pacific J. Math*, 8(1):171–176, 1958.

[99] Daniel A Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011.

[100] Éva Tardos. A strongly polynomial minimum cost circulation algorithm. *Combinatorica*, 5(3):247–255, 1985.

[101] Michael J Todd. Semidefinite optimization. *Acta Numerica 2001*, 10:515–560, 2001.

[102] Nobuaki Tomizawa and Masao Iri. Algorithm for determining rank of a triple matrix product axb with application to problem of discerning existence of unique solution in a network. *ELECTRONICS & COMMUNICATIONS IN JAPAN*, 57(11):50–57, 1974.

[103] Pravin M. Vaidya. A new algorithm for minimizing convex functions over convex sets (extended abstract). In *FOCS*, pages 338–343, 1989.

[104] Pravin M Vaidya. Speeding-up linear programming using fast matrix multiplication. In *Foundations of Computer Science, 1989., 30th Annual Symposium on*, pages 332–337. IEEE, 1989.

[105] Pravin M Vaidya. A new algorithm for minimizing convex functions over convex sets. *Mathematical Programming*, 73(3):291–341, 1996.

[106] Lieven Vandenberghe and Stephen Boyd. Semidefinite programming. *SIAM review*, 38(1):49–95, 1996.

[107] Jens Vygen. A note on schrijver's submodular function minimization algorithm. *Journal of Combinatorial Theory, Series B*, 88(2):399–402, 2003.

[108] S. Fujishige W. Cui. A primal algorithm for the submodular flow problem with minimum mean cycle selection. *Journal of the Operations Research Society of Japan*, 1988.

[109] C Wallacher and Uwe T Zimmermann. A polynomial cycle canceling algorithm for submodular flows. *Mathematical programming*, 86(1):1–15, 1999.

[110] Virginia Vassilevska Williams. Multiplying matrices faster than coppersmith-winograd. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 887–898. ACM, 2012.

[111] Yinyu Ye. Complexity analysis of the analytic center cutting plane method that uses multiple cuts. *Mathematical Programming*, 78(1):85–104, 1996.

[112] David B Yudin and Arkadii S Nemirovski. Evaluation of the information complexity of mathematical programming problems. *Ekonomika i Matematicheskie Metody*, 12:128–142, 1976.

[113] U Zimmermann. Minimization on submodular flows. *Discrete Applied Mathematics*, 4(4):303–323, 1982.

[114] Uwe Zimmermann. Negative circuits for flows and submodular flows. *Discrete applied mathematics*, 36(2):179–189, 1992.