Convex Minimization with Integer Minima in $\widetilde{O}(n^4)$ Time[*]

Haotian Jiang[†]   Yin Tat Lee[‡]   Zhao Song[§]   Lichen Zhang[¶]

Abstract

Given a convex function f on $\mathbb{R}^n$ with an integer minimizer, we show how to find an exact minimizer of f using $O(n^2 \log n)$ calls to a separation oracle and $O(n^4 \log n)$ time. The previous best polynomial time algorithm for this problem given in [Jiang, SODA 2021, JACM 2022] achieves $O(n^2 \log \log n / \log n)$ oracle complexity. However, the overall runtime of Jiang's algorithm is at least $\widetilde{\Omega}(n^8)$, due to expensive sub-routines such as the Lenstra-Lenstra-Lovász (LLL) algorithm [Lenstra, Lenstra, Lovász, Math. Ann. 1982] and random walk based cutting plane method [Bertsimas, Vempala, JACM 2004]. Our significant speedup is obtained by a nontrivial combination of a faster version of the LLL algorithm due to [Neumaier, Stehlé, ISSAC 2016] that gives similar guarantees, the volumetric center cutting plane method (CPM) by [Vaidya, FOCS 1989] and its fast implementation given in [Jiang, Lee, Song, Wong, STOC 2020].

For the special case of submodular function minimization (SFM), our result implies a strongly polynomial time algorithm for this problem using $O(n^3 \log n)$ calls to an evaluation oracle and $O(n^4 \log n)$ additional arithmetic operations. Both the oracle complexity and the number of arithmetic operations of our more general algorithm are better than the previous best-known runtime algorithms for this specific problem given in [Lee, Sidford, Wong, FOCS 2015] and [Dadush, Végh, Zambelli, SODA 2018, MOR 2021].

1 Introduction

The problem of minimizing a convex function f over $\mathbb{R}^n$, assuming access to a separation oracle SO, has gained considerable interest since the seminal work of Grötschel, Lovász, and Schrijver [GLS81, GLS88]. The separation oracle SO is such that when queried with a point $x \in \mathbb{R}^n$, it returns "YES" if x minimizes f, or else a hyperplane that separates x from the minimizers of f. One popular and successful approach for this problem is the cutting plane method, which dates back to the center of gravity method, independently discovered by Levin [Lev65] and Newman [New65] in the 1960s. Since then, cutting plane methods have undergone substantial developments and improvements over the past six decades in terms of its oracle complexity[1] and runtime efficiency [Sho77, YN76, Kha80, TKE88, NN89, Vai89, AV95, BV04, LSW15]. In particular, the current fastest cutting plane method is due to Jiang, Lee, Song, and Wong [JLSW20].

Despite outstanding progress on the cutting plane method, direct application of these methods for minimizing a convex function f on $\mathbb{R}^n$ via a separation oracle typically results in weakly-polynomial time algorithms, with the oracle complexity and runtime depending logarithmically on the accuracy parameter $\varepsilon > 0$ and the "size"[2] of the function f.

A fundamental but extremely challenging question is to design a strongly-polynomial time algorithm that efficiently computes an exact minimizer of f on $\mathbb{R}^n$, with its oracle complexity, number of arithmetic operations, and bit size all being polynomial only in the dimension n of the problem and *not* dependent on the "size" of the function f. For example, it remains a major open problem to design a strongly-polynomial time algorithm for solving linear programs (LPs). This problem is widely known as Smale's 9th problem in his list of eighteen unsolved mathematical problems for the 21st century [Sma98]. In particular, recent breakthroughs on LP solvers are all weakly-polynomial time algorithms, e.g., [LS14, LS19, CLS19, BLSS20, JSWZ21]. Despite this obstacle, strongly-polynomial time algorithms for LPs are known under additional structures, such as LPs with at most two non-zero entries per row [Meg83, AC91, CM91] or per column [Vég14, OV20] in the constraint matrix, LPs

---

[1]The oracle complexity of an algorithm is the number of oracle calls made by the algorithm in the worst case.

[2]Depending on the specific problem setting, the "size" of the function can be its range, Lipschitzness, condition number, length of binary representation, etc.

with bounded entries in the constraint matrix [Tar86, VY96, DHNV20], and LPs with 0-1 optimal solutions [Chu12, Chu15].

For minimizing a general convex function f with access to a separation oracle, it is impossible to design a strongly-polynomial time algorithm unless the function f satisfies additional combinatorial properties. One natural and general assumption, satisfied by many fundamental discrete optimization problems such as submodular function minimization (SFM), maximum flow, maximum matching, and shortest path, is that the convex function f has an integer minimizer. Under the integer minimizer assumption, Jiang [Jia21, Jia22], improving upon the classical work of Grötschel, Lovász, and Schrijver [GLS81, GLS88], recently gave a strongly-polynomial time algorithm[3] for finding an exact minimizer of f using $O(n(n \log \log n / \log n + \log R))$ calls to SO, where $R = 2^{\text{poly}(n)}$ is the ℓ_∞-norm of the integer minimizer.

In fact, Jiang's general result implies significant improvement to the oracle complexity of strongly-polynomial time algorithms even for the special problem of SFM (see Subsection 1.2 for more details). Despite its favorable oracle complexity, Jiang's algorithm actually requires $\widetilde{\Omega}(n^8)$ additional arithmetic operations to implement. This additional part of the runtime is prohibitively large for its application to SFM; other state-of-the-art strongly-polynomial time algorithms for SFM [LSW15, JLSW20, DVZ21] have worse oracle complexity but use much fewer arithmetic operations.

Unlike the special case of SFM, for the more general problem of minimizing a convex function with an integer minimizer, the $\widetilde{\Omega}(n^8)$ arithmetic operations in Jiang's algorithm is the state-of-the-art for strongly-polynomial time algorithms with near-optimal oracle complexity. The lack of a fast algorithm for the general problem, as well as the tradeoff between smaller oracle complexity and fewer additional arithmetic operations for state-of-the-art SFM algorithms, naturally lead to the following question:

Can we minimize a convex function with integer minimizers using a separation oracle in a way that is both oracle-efficient and requires much fewer arithmetic operations?

In this paper, we provide an affirmative answer to the above question. Specifically, we give an algorithm that minimizes a convex function with an integer minimizer using only $O(n^2 \log(nR))$ separation oracle calls and an additional $O(n^4 \log(nR))$ arithmetic operations. When applied to SFM, our result implies an algorithm that makes $O(n^3 \log n)$ evaluation oracle queries and uses an additional $O(n^4 \log n)$ arithmetic operations. This improves, for both the oracle complexity and the additional arithmetic operations, the state-of-the-art strongly-polynomial time SFM algorithms in [LSW15, DVZ21]. Compared with Jiang's algorithm [Jia22], despite having a slightly bigger oracle complexity, the additional number of arithmetic operations used by our algorithm is significantly lower by a $\widetilde{\Theta}(n^4)$ factor.

1.1 Our Result The main result of this paper is a much more efficient algorithm for minimizing convex functions with integer minimizers:

THEOREM 1.1. (MAIN RESULT, INFORMAL VERSION OF THEOREM 5.1) *Given a separation oracle* SO *for a convex function f on $\mathbb{R}^n$. If the set of minimizers K^* of f is contained in a box of radius R and all extreme points of K^* are integer vectors, then there exists a randomized algorithm that outputs an integer minimizer of f with high probability using*

- $O(n^2 \log(nR))$ *queries to* SO*, and*

- $O(n^4 \log(nR))$ *additional arithmetic operations.*

The strong assumption that all extreme points of K^* are integer vectors guarantees that the algorithm outputs an *integer* minimizer of f. In fact, this assumption is necessary for the algorithm to efficiently compute an integer minimizer (see Remark 1.4 in [Jia22] for an example). Without such an assumption, the algorithm can still output a minimizer, though not neccessarily an integer one, with the same guarantee as in Theorem 1.1 (see Remark 1.5 in [Jia22]).

Previously, under the same assumptions as in Theorem 1.1, Jiang [Jia22] gave an algorithm that computes an integer minimizer of f using $O(n(n \log \log n / \log n + \log R))$ separation oracle calls, which is better than the oracle

---

[3]In fact, Jiang [Jia21, Jia22] gave a family of algorithms for this problem via an algorithmic reduction to the shortest vector problem. For instance, he gave an algorithm that achieves a nearly-optimal oracle complexity of $O(n \log(nR))$ but using exponential time.

complexity in Theorem 1.1. However, his algorithm uses an enormous $\widetilde{O}(n^8)$ number of additional arithmetic operations because of the following reasons: to obtain a subquadratic oracle complexity, [Jia22] resorts to the random walk based cutting plane method by Bertsimas and Vempala [BV04] to work directly with the polytope K formed by all the separating hyperplanes output by SO. Unfortunately, the sampling approach in [BV04] requires $\Omega(n^6)$ arithmetic operations to perform one iteration of the cutting plane method. In addition, to guarantee that the sampling can be performed efficiently, [Jia22] has to maintain two "sandwiching" ellipsoids, contained in and containing the polytope K respectively, which requires additional computational cost to preserve after reducing the dimension of the problem.

Another computational bottleneck in [Jia22] is that the algorithm approximates the length of the shortest vector after every iteration of the cutting plane method. To do so, [Jia22] leverages the sieve algorithm in [AKS01] that computes a $2^{n \log \log n / \log n}$-approximation to the shortest vector using $\widetilde{O}(n^4)$ arithmetic operations. Compounding with the overall $\widetilde{O}(n^2)$ iterations of the algorithm, this step of approximating the shortest vector takes $\Omega(n^6)$ arithmetic operations in total.

To get around the aforementioned computational bottlenecks, we utilize an intricate combination of a computationally more efficient cutting plane method based on the volumetric center of K due to Vaidya [Vai89] and a faster version of the LLL algorithm given in [NS16]. Instead of working directly with the polytope K and performing volume reduction in a step-by-step manner, we run Vaidya's cutting plane method [Vai89] in blocks of $O(n \log n)$ consecutive steps, the volume decreases by a constant factor per step only in an amortized sense within each block. In particular, recent work by Jiang, Lee, Song and Wong [JLSW20] shows that $O(n \log n)$ steps of Vaidya's method can be implemented using a total of $O(n^3 \log n)$ arithmetic operations. Running Vaidya's method in blocks also enables us to compute only $\widetilde{O}(n)$ approximate shortest vectors in total, while still being able to avoid the appearance of extremely short vectors. Harnessing the faster implementation of the LLL algorithm in [NS16] allows us to compute the $\widetilde{O}(n)$ approximate shortest vectors using a total of $\widetilde{O}(n^4)$ arithmetic operations.

1.2 Applications to Submodular Function Minimization Submodular function minimization (SFM) has been one of the cornerstone problems in combinatorial optimization since the seminal work of Edmonds [Edm03]. Many classic combinatorial problems can be abstracted as optimization over a submodular function, such as graph cut function, set coverage function and economic utility function. For more comprehensive reviews of SFM, we refer readers to [McC05, Iwa08].

Papers	Year	Runtime	Remarks	General?
[GLS81, GLS88]	1981,88	$\widetilde{O}(n^5 \cdot \mathsf{EO} + n^7)$ [McC05]	first strongly	✓
[Sch00]	2000	$O(n^8 \cdot \mathsf{EO} + n^9)$	first comb. strongly	
[IFF01]	2000	$O(n^7 \log n \cdot \mathsf{EO} + \mathrm{poly}(n))$	first comb. strongly	
[FI03]	2000	$O(n^7 \cdot \mathsf{EO} + n^8)$		
[Iwa03]	2002	$O(n^6 \log n \cdot \mathsf{EO} + n^7 \log n)$		
[Vyg03]	2003	$O(n^7 \cdot \mathsf{EO} + n^8)$		
[Orl09]	2007	$O(n^5 \cdot \mathsf{EO} + n^6)$		
[IO09]	2009	$O(n^5 \log n \cdot \mathsf{EO} + n^6 \log n)$		
[LSW15]	2015	$O(n^3 \log^2 n \cdot \mathsf{EO} + n^4 \log^{O(1)} n)$	previous best runtime	
[LSW15]	2015	$O(n^3 \log n \cdot \mathsf{EO} + 2^{O(n)})$	exponential time	
[DVZ21]	2018	$O(n^3 \log^2 n \cdot \mathsf{EO} + n^4 \log^{O(1)} n)$	previous best runtime	
[Jia21]	2021	$O(n^3 \frac{\log \log n}{\log n} \cdot \mathsf{EO} + n^8 \log n)$	best oracle complexity	✓
[Jia21]	2021	$O(n^2 \log n \cdot \mathsf{EO} + 2^{O(n)})$	exponential time	✓
This paper	2023	$O(n^3 \log n \cdot \mathsf{EO} + n^4 \log n)$	best runtime	✓

Table 1: Strongly-polynomial algorithms for submodular function minimization. The oracle complexity measures the number of calls to the evaluation oracle EO. In the case where a paper is published in both conference and journal, the year we provide is the earlier one. In the column "General?", ✓ means that the algorithm works for the more general problem of minimizing convex functions with integer minimizers studied in this paper.

We briefly recall the standard setting of SFM: we are given a submodular function $f : 2^V \to \mathbb{Z}$ over an

n-element ground set V where the function f is accessed via an evaluation oracle, which returns the value $f(S)$ for a query set S using time EO. The goal is to find the minimizer of f using queries to the evaluation oracle and additional arithmetic operations.

An SFM algorithm is called strongly-polynomial time if its runtime depends polynomially only on EO and the dimension n, but not on the range of the function f. In their seminal work, Grötschel, Lovász, and Schrijver [GLS84, GLS88] gave the first strongly-polynomial time algorithm for SFM based on the ellipsoid method. Since then, there has been a long history of efforts in designing faster strongly-polynomial time algorithms for SFM (see Table 1).

The state-of-the-art strongly-polynomial time algorithms for SFM, in terms of the number of arithmetic operations used, were given by Lee, Sidford, and Wong [LSW15] and Dadush, Végh, and Zambelli [DVZ21]. Both algorithms have runtime $O(n^3 \log^2 n \cdot \mathsf{EO} + n^4 \log^{O(1)} n)$. The oracle complexity of these algorithms were later improved to sub-cubic in terms of n by Jiang [Jia21, Jia22], where he gave a strongly-polynomial time algorithm with runtime $O(n^3 \log \log n / \log n \cdot \mathsf{EO} + n^8 \log^{O(1)} n)$ alongside an exponential time algorithm with a nearly-quadratic $O(n^2 \log n)$ oracle complexity. It remains a major open problem in the area of SFM whether there exists a strongly-polynomial time algorithm with truly sub-cubic, i.e. $O(n^{3-c})$ for some absolute constant $c > 0$, oracle complexity.

Unfortunately, while the algorithm in [Jia22] answers major open questions in [LSW15] and makes significant progress on the oracle complexity for SFM, the number of additional arithmetic operations used by Jiang's algorithm is a factor of $\widetilde{\Theta}(n^4)$ larger than the algorithms in [LSW15, DVZ21]. In this paper, we complement Jiang's algorithm by giving the following strongly-polynomial time SFM algorithm that improves both the oracle complexity and the number of arithmetic operations used by the algorithms in [LSW15, DVZ21].

COROLLARY 1.1. (SUBMODULAR FUNCTION MINIMIZATION) *Given an evaluation oracle* EO *for a submodular function f defined over an n-element ground set, there exists a randomized strongly-polynomial time algorithm that computes an exact minimizer of f with high probability using*

- $O(n^3 \log n)$ *queries to* EO, *and*

- $O(n^4 \log n)$ *additional arithmetic operations.*

Corollary 1.1 almost immediately follows from Theorem 1.1 together with the standard fact that a separation oracle for the *Lovász extension* of the submodular function f can be implemented by making n queries to the evaluation oracle of f and $O(n \log n)$ additional arithmetic operations (e.g., Theorem 6.4 in [Jia22]). The proof of Corollary 1.1 is essentially identical to the proof of Theorem 1.7 in [Jia22], and is therefore omitted.

Compared to the previous best runtime algorithms in [LSW15] and [DVZ21], our algorithm improves their oracle complexity from $O(n^3 \log^2 n)$ to $O(n^3 \log n)$ while also improving the number of arithmetic operations from $O(n^4 \log^{O(1)} n)$ to $O(n^4 \log n)$. We highlight that in [LSW15], the authors manage to achieve an $O(n^3 \log n)$ oracle complexity, but at the expense of an exponential runtime. It is also important to note that our improvements are achieved via a much more *general* algorithm, whereas the algorithms in [LSW15, DVZ21] work specifically for SFM.

In comparison to the current best oracle complexity algorithm in [Jia22], our algorithm has a slightly worse oracle complexity, but we significantly improve the $\widetilde{O}(n^8)$ additional arithmetic operations in his algorithm down to $O(n^4 \log n)$.

Finally, note that the current best implementation of a separation oracle for the Lovász extension requires n queries to EO, and the current fastest cutting plane method requires $O(n^2)$ arithmetic operations per step. So for any cutting plane algorithm for SFM that uses T iterations, the current best runtime we can hope for such a method is $O(Tn \cdot \mathsf{EO} + Tn^2)$ using state-of-the-art techniques. Our algorithm, in fact, matches such a runtime bound. In particular, we use $O(n^2 \log n)$ iterations of the cutting plane method with a total runtime of $O(n^3 \log n \cdot \mathsf{EO} + n^4 \log n)$. So in some sense, our result matches what can be achieved using the current best known algorithmic techniques for cutting plane methods. In contrast, the algorithm in [Jia22] does not have such a feature, with the $\Omega(n^8)$ additional arithmetic operations being much larger than its oracle complexity.

2 Technique Overview

In this section, we provide a brief overview of the techniques in prior work and in our work. In Section 2.1, we review the methods in [Jia22]. In Section 2.2, we present a preliminary overview of our approach. In Section 2.3,

we summarize current progress on the problem of minimizing convex functions with integer minimizers and discuss potential future directions.

2.1 An Overview of Previous Work Before discussing our main technical insights, we first review the approach of [Jia22] that achieves subquadratic oracle calls for minimizing convex functions with integer minimizers. The key ingredient in Jiang's algorithm is an interplay between the polytope, formed by the separating hyperplanes returned by the separation oracle, which contains the integer minimizer, and a lattice that captures the structures of integer points.

In particular, in each iteration of the cutting plane method, Jiang's algorithm examines the length of the (approximate) shortest vector in the lattice under the $\mathrm{Cov}(K)$-norm, where $\mathrm{Cov}(K)$ is the covariance matrix of the polytope K. The length of this vector under $\mathrm{Cov}(K)$-norm provides a measurement for the width of the outer ellipsoid induced by $\mathrm{Cov}(K)^{-1}$. Thus, when this norm is small, it implies that the outer ellipsoid has a small width and we can safely proceed by reducing dimension and projecting the lattice onto the orthogonal complement of the shortest vector. On the other hand, if the shortest vector still has a relatively large norm, then the polytope K can be further refined by using cutting plane method. To achieve the best possible oracle complexity, the algorithm in [Jia22] performs the width measurement in a *step-by-step* fashion, meaning that it actively checks the length of the shortest vector after every single cutting plane step. This ensures that the algorithm can enter the dimension reduction phase as soon as a short lattice vector can be obtained, which happens when the volume of K is small enough.

Unfortunately, such a step-by-step strategy and careful width measurement come at a price – the approximate shortest vector subprocedure will be called in every iteration of the cutting plane method. Moreover, an expensive cutting plane method, such as the random walk based center of gravity method [BV04], that guarantees the volume shrinks in every step needs to be used.

2.2 Our Approach: Cutting Plane in Blocks and Lazy Width Measurement Now we discuss our approach for overcoming the major computational bottleneck of [Jia22], which is the step-by-step measurement of width mentioned above. For simplicity, we assume the integer minimizer $x^* \in \{0, 1\}^n$ (i.e. $R = 1$) in the subsequent discussion. By delaying the width measurement, i.e. find an approximate shortest vector after a block of cutting plane steps, we can utilize the much more efficient cutting plane method due to Vaidya [Vai89]. However, this comes at the cost of a possibly much larger number of oracle calls, since the length of the shortest vector in the lattice might become very small when we actually perform the measurement, which will lead to a large increase in the volume of K after dimension reduction. Nevertheless, we show that by carefully balancing the block size of cutting plane steps and the loss incurred during dimension reduction due to short lattice vectors, we can still achieve an $O(n^2 \log n)$ oracle complexity.

To explain our approach, we first provide a brief introduction to Vaidya's cutting plane method. Unlike classical cutting plane methods, such as the ellipsoid method and the center of gravity method, where each step is guaranteed to shrink the volume, Vaidya's algorithm and analysis rely on controlling the volume of the Dikin ellipsoid induced by the log barrier on polytope K. The algorithm iteratively finds the point ω_K whose corresponding Dikin ellipsoid has the largest volume, which is called the volumetric center. Subsuquently, the algorithm uses a separation oracle on the volumetric center ω_K. If ω_K is a minimizer, then we are done; otherwise, the algorithm computes the leverage score of each constraint with respect to the Hessian matrix, which measures the relative importance of each constraint. If all constraints are relatively important, then a new constraint is added based on the separating hyperplane returned by the separation oracle at ω_K. If one of the constraints has leverage score smaller than some tiny constant, then it is dropped to ensure that the polytope K always has at most $O(n)$ constraints. Due to the increase in the volume of the polytope K when constraints are dropped, the volume of K only shrinks in an amortized sense. In particular, only after $O(n \log n)$ steps, the volume of K is guaranteed to decrease by a multiplicative factor of $2^{-O(n \log n)}$ from the initial volume. It is crucial to note that no guarantee on the volume shrinkage of K is known if $o(n \log n)$ iterations of Vaidya's cutting plane method is run.

Our idea is then to view the $O(n \log n)$ steps as a "block", meaning that each time we measure the shortest vector and realize that it still has a large norm, we execute Vaidya's cutting plane steps for $O(n \log n)$ steps and then re-examine the norm. This naturally induces a strategy that measures the width of the ellipsoid in a *lazy* fashion. The volume decrease of K via this strategy is similar to that when running the cutting plane method in [BV04] for $O(n \log n)$ steps. Therefore, the only issue is the significant decrease in the norm of the shortest

vector during a block of cutting plane steps, which can be as small as $2^{-O(n \log n)}$. Fortunately, this shrinkage of the norm is acceptable and it incurs only an additional $O(\log n)$ factor in the number of oracle calls.

One major advantage of using Vaidya's cutting plane method is that the iterations can be performed very efficiently. Using the fast implementation in [JLSW20], a block of $O(n \log n)$ cutting plane steps can be done in $O(n^3 \log n)$ time. Lazy width measurement also reduces the total number of approximate shortest vectors we need to compute from $\widetilde{O}(n^2)$ to $\widetilde{O}(n)$, which opens up the gate of using faster approximate shortest vector algorithms in [NS16].

Although the efficiency issue of the cutting plane step and width measurement is addressed, we still need to carefully control the complexity of the dimension reduction step. This step can be particularly expensive due to the loss of structural information of the polytope K after collapsing it onto a proper subspace P on which the algorithm recurs, as discussed in [Jia22]. In the standard setting of Vaidya's cutting plane method, the polytope evolves in a "slow-changing" manner that makes it easy to maintain and update the volumetric center and its corresponding Hessian matrix. However, after reducing a dimension, we no longer have such a "slow-changing" property, and thus the new volumetric center and Hessian matrix have to be computed from scratch.

A natural idea would be to formulate the problem of recomputing the volumetric center and Hessian as a convex optimization task constrained to the polytope $K \cap P$ we have access to. On the surface, this problem can be solved straightforwardly using the same cutting plane procedure efficiently. The caveat is that evaluating the volumetric function or its gradient requires $O(n^\omega)$ time[4]. As the cutting plane method evolves for $O(n \log n)$ iterations, this will lead to a total of $O(n^{\omega+1} \log n)$ arithmetic operations. To circumvent this issue, we start from a simpler convex body containing $K \cap P$ whose volumetric center can be easily computed. Specifically, we choose the hyperrectangle centered at the center of the outer ellipsoid that contains $K \cap P$. We show that this method only affects the number of oracle calls by a factor of $O(\log n)$, despite causing the volume to blow up by a factor of $n^{O(n)}$. A similar approach is also taken in [Jia22] to ease the computation of centroid and covariance matrix of $K \cap P$. However, his method requires iterativelty refining the hyperrectangle using constraints of $K \cap P$ until it coincides with $K \cap P$, at which point the algorithm relearns the collapsed polytope $K \cap P$. This approach can take as many as $\widetilde{O}(n^2)$ steps (without calling the separation oracle) with $\Omega(n^2)$ operations per step, and is mainly for achieving the best possible oracle complexity. Our approach is arguably simpler and much more efficient.

Another challenge is to prove that using volumetric centers and Dikin ellipsoids [Dik67] is sufficient to progress the algorithm. In Jiang's algorithm, the use of centroid and covariance matrix makes the analysis straightforward due to the standard nice geometric property that the polytope K is sandwiched between ellipsoids induced by the covariance matrix at the centroid. However, in our analysis, we use the volumetric center and Dikin ellipsoid of the log barrier, which requires a different approach. Past works that exclusively analyze the performance of the cutting plane method based on this approach [Vai89] take a functional value point of view, measuring the progress of the algorithm via the change of the volumetric function. Instead, our analysis extracts key geometric information of Vaidya's cutting plane method, showing that the volumetric center and Dikin ellipsoid evolve in a similar spirit as the centroid and covariance matrix.

In particular, we prove structural result for using Dikin ellipsoids [Dik67] to sandwich of an n-dimensional polytope defined by m constraints. Let us parametrize $K = \{x \in \mathbb{R}^n : Ax \geq b\}$ where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$, define the Hessian of the log-barrier as $H(x) = A^\top S_x^{-2} A$ where $S_x \in \mathbb{R}^{m \times m}$ is the diagonal matrix with i-th diagonal being $a_i^\top x - b_i$. The volumetric function is defined as $F(x) = \frac{1}{2} \log \det(H(x))$ and we let $\mathrm{vc}(K)$ denote the minimizer of F, i.e. $\mathrm{vc}(K)$ is the volumetric center of K. The key structural result we prove is that $E(\mathrm{vc}(K), H(\mathrm{vc}(K))) \subseteq K \subseteq \mathrm{poly}(mn) \cdot E(\mathrm{vc}(K), H(\mathrm{vc}(K)))$, where $E(\mathrm{vc}(K), H(\mathrm{vc}(K)))$ is the ellipsoid centered at $\mathrm{vc}(K)$ and defined by $H(\mathrm{vc}(K))$. While it is standard for K to be sandwiched by Dikin ellipsoids induced by a self-concordant barrier function and centered at the minimizer of that function, we note that $\mathrm{vc}(K)$ is *not* the minimizer of the *log-barrier function* (but rather the minimizer of F), while the Dikin ellipsoid is defined w.r.t. the Hessian H of the log-barrier. In fact, for the Dikin ellipsoid centered at the analytic center denoted by $\mathrm{ac}(K)$ (minimizer of the log-barrier function) and the induced Hessian at $\mathrm{ac}(K)$, we have

$$E(\mathrm{ac}(K), H(\mathrm{ac}(K))) \subseteq K \subseteq O(m) \cdot E(\mathrm{ac}(K), H(\mathrm{ac}(K))).$$

On the other hand, to progress Vaidya's cutting plane method, we have to work with the Dikin ellipsoid centered and induced by $\mathrm{vc}(K)$ instead of $\mathrm{ac}(K)$. We first note that the scaled volumetric function indeed defines

---

[4]ω is the exponent of matrix multiplication [DWZ23, VWXXZ24, LG24]. Currently, $\omega \approx 2.373$.

a self-concordant barrier function and we thus have

$$E(\mathrm{vc}(K), \nabla^2 F(\mathrm{vc}(K))) \subseteq K \subseteq O(mn) \cdot E(\mathrm{vc}(K), \nabla^2 F(\mathrm{vc}(K))).$$

For the left containment, we can safely replace $\nabla^2 F(\mathrm{vc}(K))$ as for any $x \in K$, it is true that $E(x, H(x)) \subseteq K$. For the right containment, we utilize the fact that $H(x)$ and $\nabla^2 F(x)$ are close spectral approximation of each other, thus we have $E(\mathrm{vc}(K), \nabla^2 F(\mathrm{vc}(K))) \subseteq O(\sqrt{m}) \cdot E(\mathrm{vc}(K), H(\mathrm{vc}(K)))$, and we conclude

$$E(\mathrm{vc}(K), H(\mathrm{vc}(K))) \subseteq K \subseteq \mathrm{poly}(mn) \cdot E(\mathrm{vc}(K), H(\mathrm{vc}(K))).$$

We believe this sandwiching condition might be of independent interest for other geometric applications of Vaidya's method.

To carry out the analysis, we use the potential function developed in [Jia22], which captures the volume of the polytope and the density of the lattice simultaneously. We show that the block of cutting plane steps also leads to rapid decrement of the potential. Unlike [Jia22], the block cutting plane steps and Dikin ellipsoids cause extra losses on the volume of K after dimension reduction, due to the possible appearance of very short vectors. We observe that such blowups are always at most $n^{O(n)}$ for each dimension reduction step. Thus, the potential increases by at most $O(n^2 \log n)$ in total, and we need to use a total of $O(n^2 \log n)$ oracle calls to counter this increment. To summarize, by trading for a slightly-worse number of oracle calls, we make more room for the algorithm to gain extra speedup through leverage score maintenance, faster approximate shortest vector, and crude estimation of the convex body after reducing dimension.

2.3 Discussion A natural question that arises from our result is whether it is possible to obtain a strongly-polynomial time algorithm with quadratic or subquadratic oracle complexity, matching the one in [Jia22], while achieving the same improved runtime as ours. Our algorithm has only one $\log n$ factor in the oracle complexity, but we contend that this is inherent for Vaidya's approach since it can only guarantee a volume decrease after $O(n \log n)$ cutting plane steps. Additionally, in the dimension reduction phase, we use a hyperrectangle to approximate the convex body, which introduces a volume increase of $n^{O(n)}$. To resolve this issue, an algorithm that approximately computes the volumetric center in $O(n^3 \log n)$ time would be necessary. However, designing such an algorithm is an interesting and nontrivial data structure task, as discussed in the preceding subsection, since it requires maintaining and updating the log determinant of a Gram matrix under slow diagonal changes.

To achieve a subquadratic oracle complexity, a crucial ingredient in [Jia22] is an approximate shortest vector algorithm with a sub-exponential approximation factor, first given in [AKS01]. The main reason for the sub-exponential approximation ratio is a block-reduction scheme introduced in [Sch87] that computes a more general notion of reduced lattice basis. Recent improvements on basis reduction algorithms make use of this block-reduction idea to obtain more refined recursive structures. Therefore, it is of interest to design approximate shortest vector algorithms using $\widetilde{O}(n^3)$ arithmetic operations while achieving a sub-exponential approximation factor.

3 Preliminary

In Section 3.1, we provide several basic notations, definitions and facts. In Section 3.2, we discuss LLL algorithm and shortest vector problem. In Section 3.3, we define and state several basic tools in convex geometry. In Section 3.4, we provide some related definitions about lattice projection. In Section 3.5, we present the slicing lemma. In Section 3.6, we state a dimension reduction lemma.

3.1 Notations and Basic Facts

Basic Notations. For an integer n, we use $[n]$ to denote the set $\{1, 2, \cdots, n\}$. For any function f, we use $\widetilde{O}(f)$ to denote $f \cdot \mathrm{poly}(\log f)$.

Matrices and Vectors. For a vector x, we use $\|x\|_2$ to denote its ℓ_2 norm. For a vector x, we use $x^\top$ to denote its transpose. For a matrix A, we use $A^\top$ to denote its transpose. We use $\mathbf{0}_n$ to denote a length-n vector where all the entries are zeros. We use $\mathbf{1}_n$ to denote a length-n vector where all the entries are ones.

We say a square matrix $A \in \mathbb{R}^{n \times n}$ is PSD ($A \succeq 0$) if for all vectors $x \in \mathbb{R}^n$, $x^\top A x \geq 0$. For a square matrix A, we use $\det(A)$ to denote the determinant of matrix A. For a square and invertible matrix A, we use A^{-1} to denote the inverse of matrix A.

For a PSD matrix A, we define the induced matrix norm for any vector x as $\|x\|_A := \sqrt{x^\top A x}$.

Ellipsoid. Given a point $x_0 \in \mathbb{R}^n$ and a PSD matrix $A \in \mathbb{R}^{n \times n}$, we use $E(x_0, A)$ to denote the (not necessarily full-rank) ellipsoid given by

$$E(x_0, A) := \{x \in x_0 + W_A : (x - x_0)^\top A(x - x_0) \leq 1\},$$

where W_A is the subspace spanned by eigenvectors corresponding to nonzero eigenvalues of A. When the ellipsoid is centered at $\mathbf{0}_n$, we use the short-hand notation $E(A)$ to denote $E(\mathbf{0}_n, A)$.

Lattices. Let $b_1, \ldots, b_k \in \mathbb{R}^n$ be a set of linearly independent vectors, we use

$$\Lambda(b_1, \ldots, b_k) = \{\sum_{i=1}^k \lambda_i b_i, \lambda_i \in \mathbb{Z}\}$$

denote the lattice generated by $b_1, \ldots, b_k$ and k is the rank of the lattice. If $k = n$, then it's full rank. A basis of $\Lambda := \Lambda(b_1, \ldots, b_k)$ is a set of k linearly independent vectors that generates Λ under integer combinations. Bases of Λ are equivalent under unimodular transforms. We use $\lambda_1(\Lambda)$ to denote the ℓ_2 norm of the shortest nonzero vector in Λ and $\lambda_1(\Lambda, A)$ to denote the induced A-norm of the shortest nonzero vector in Λ.

Given a basis $B \in \mathbb{R}^{n \times k}$, the parallelotope associated to it is the polytope $P(B) = \{\sum_{i=1}^k \lambda_i b_i : \lambda_i \in [0, 1), \forall i \in [k]\}$. The determinant of Λ is the volume of $P(B)$, which is independent of basis.

The dual lattice of Λ is defined as follows:

DEFINITION 3.1. (DUAL LATTICE) *Given a lattice* $\Lambda \subseteq \mathbb{R}^n$, *the dual lattice* Λ^* *is the set of all vectors* $x \in \mathrm{span}\{\Lambda\}$ *such that* $\langle x, y \rangle \in \mathbb{Z}$ *for all* $y \in \Lambda$.

For more backgrounds about lattices, we refer the readers to lecture notes by Rothvoss [Rot16].

Leverage Score. We define leverage score, which is a standard concept in numerical linear algebra [CW13, BWZ16, SWZ17, SWZ19, JLSW20, SXZ22, SYYZ22, DSW22]. We remark that leverage score has multiple equivalent definitions, here we just present one of them.

DEFINITION 3.2. (LEVERAGE SCORE) *Given a matrix* $A \in \mathbb{R}^{m \times n}$, *we define the leverage score for matrix* A *to be* $\sigma \in \mathbb{R}^m$, *i.e,*

$$\sigma_i = a_i^\top (A^\top A)^{-1} a_i, \quad \forall i \in [m]$$

Note that $a_i^\top$ *is the i-th row of A.*

We state a useful fact here.

FACT 3.1. (FOLKLORE) *Given a matrix* $A \in \mathbb{R}^{m \times n}$, *we have the following identity of its leverage score:*

$$\sum_{i=1}^m \sigma_i = n.$$

3.2 LLL Algorithm for Shortest Vector Problem Given a lattice Λ and a corresponding basis $B \in \mathbb{R}^{n \times k}$, it is natural to seek an algorithm that finds the vector with norm $\lambda_1(L)$, or at least approximately finds it. The famous Lenstra-Lenstra-Lovász algorithm serves such a purpose:

THEOREM 3.1. (LLL ALGORITHM, [LLL82]) *Let* $b_1, \cdots, b_k \in \mathbb{Z}^n$ *be a basis for lattice* Λ *and* $A \in \mathbb{Z}^{n \times n}$ *be a PSD matrix that is full rank on* $\mathrm{span}(\Lambda)$. *Let* $D \in \mathbb{R}$ *such that* $\|b_i\|_A^2 \leq D$ *for any* $i \in [k]$. *Then there exists an algorithm that outputs in* $O(n^4 \log(D))$ *arithmetic operations a nonzero vector* b_1' *such that*

$$\|b_1'\|_A^2 \leq 2^{k-1} \cdot \lambda_1^2(\Lambda, A)$$

Moreover, the integers occuring in the algorithm have bit sizes at most $O(n \log(D))$.

Lately, [NS16] improves the runtime of the LLL algorithm by leveraging the block reduction technique introduced in [Sch87]. This is a key component in our $O(n^4 \log n)$ time algorithm.

THEOREM 3.2. (THEOREM 2 OF [NS16]) *Let $b_1, \cdots, b_k \in \mathbb{Z}^n$ be a basis for lattice Λ and $A \in \mathbb{Z}^{n \times n}$ be a PSD matrix that is full rank on $\mathrm{span}(\Lambda)$. Let $D \in \mathbb{R}$ such that $\|b_i\|_A^2 \leq D$ for any $i \in [k]$. Then there exists an algorithm that outputs in $O(n^3)$ arithmetic operations to a nonzero vector b_1' such that*

$$\|b_1'\|_A^2 \leq 2^{k-1} \cdot \lambda_1^2(\Lambda, A)$$

Moreover, the integers occurring in the algorithm have bit sizes at most $O(n \log(D))$.

3.3 Convex Geometry In this section, we define notions about centroid and covariance. The work [Jia22] heavily exploits the structure of objects to design a subqudratic oracle complexity algorithm for minimizing convex functions. Our approach also relates to these notions.

For a convex body K, we use $\mathrm{vol}(K)$ to denote its volume, i.e., $\mathrm{vol}(K) := \int_{x \in K} \mathrm{d}x$. We first define the centroid of convex body,

DEFINITION 3.3. *Let $K \subseteq \mathbb{R}^n$ be a convex body. Let $g : K \to \mathbb{R}_+$ denote the uniform measure on convex body K. We define the* centroid *of K as*

$$\mathrm{cg}(K) = \int_K g(x) \cdot x \; \mathrm{d}x.$$

Equivalently, we can write $\mathrm{cg}(K)$ as

$$\mathrm{cg}(K) = \frac{1}{\mathrm{vol}(K)} \int_K x \; \mathrm{d}x.$$

Then, we define the covariance of convex body,

DEFINITION 3.4. (COVARIANCE OF CONVEX BODY, $\mathrm{Cov}(K)$) *Let $K \subseteq \mathbb{R}^n$ be a convex body. We define the covariance matrix of K under uniform measure as*

$$\mathrm{Cov}(K) = \frac{1}{\mathrm{vol}(K)} \int_K (x - \mathrm{cg}(K))(x - \mathrm{cg}(K))^\top \; \mathrm{d}x.$$

It is well-known that any isotropic[5] convex body is enclosed by two balls.

LEMMA 3.1. (ELLIPSOIDAL APPROXIMATION OF CONVEX BODY, [KLS95]) *Let K be an isotropic convex body in $\mathbb{R}^n$. Then,*

$$\sqrt{\frac{n+1}{n}} \cdot B_2 \subseteq K \subseteq \sqrt{n(n+1)} \cdot B_2,$$

where B_2 is the unit Euclidean ball in $\mathbb{R}^n$.

If K is non-isotropic, then

$$\sqrt{\frac{n+1}{n}} \cdot E(\mathrm{cg}(K), \mathrm{Cov}(K)^{-1}) \subseteq K \subseteq \sqrt{n(n+1)} \cdot E(\mathrm{cg}(K), \mathrm{Cov}(K)^{-1}).$$

3.4 Lattice Projection We collect some standard facts of lattice projection that is directly implied by Gram-Schmidt process. We use $\Pi_W(\cdot)$ to denote the orthogonal projection onto the subspace W.

FACT 3.2. (LATTICE PROJECTION) *Let Λ be a full rank lattice in $\mathbb{R}^n$ and W be a linear subspace such that $\dim(\mathrm{span}(\Lambda \cap W)) = \dim(W)$. Then*

$$\det(\Lambda) = \det(\Lambda \cap W) \cdot \det(\Pi_{W^\perp}(\Lambda)).$$

FACT 3.3. (DUAL OF LATTICE PROJECTION) *Let Λ be a full rank lattice in $\mathbb{R}^n$ and W be a linear subspace such that $\dim(\mathrm{span}(\Lambda \cap W)) = \dim(W)$. Then*

$$(\Pi_W(\Lambda))^* = \Lambda^* \cap W.$$

---

[5]A convex body is isotropic if the uniform distribution over the body has zero mean and covariance matrix being the identity.

3.5   Slicing Lemma We present a variant of Lemma 3.2 in [Jia21] in our Lemma 4.10. In particular, we replace the norm with respect to $\mathrm{Cov}(K)$ to the norm with respect to H_K^{-1}. Before stating the our new lemma (Lemma 4.10), we first recall the high dimension slicing lemma of [Jia21].

LEMMA 3.2. (LEMMA 3.2 IN [JIA21]) *Let K be a convex body and L be a full-rank lattice in $\mathbb{R}^n$. Let W be an $(n-k)$-dimensional subspace of $\mathbb{R}^n$ such that $\dim(L \cap W) = n - k$. Then*

$$\frac{\mathrm{vol}(K \cap W)}{\det(L \cap W)} \leq \frac{\mathrm{vol}(K)}{\det(L)} \cdot \frac{k^{O(k)}}{\lambda_1(L^*, \mathrm{Cov}(K))^k}$$

where L^* is the dual lattice (see Definition 3.1) and $\lambda_1(L^*, \mathrm{Cov}(K))$ is the shortest nonzero vector in L^* under the norm $\|\cdot\|_{\mathrm{Cov}(K)}$.

3.6   Dimension Reduction Lemma The key building block of both [Jia21] and our algorithm is a lattice-dimension reduction step. The following result from [Jia21] shows that all integral points are preserved after a dimension reduction step.

LEMMA 3.3. (LEMMA 3.1 OF [JIA21]) *Given an affine subspace $W = x_0 + W_0$ where W_0 is a subspace of $\mathbb{R}^n$ and $x_0 \in \mathbb{R}^n$ is some fixed point, and an ellipsoid $E = E(x_0, A)$ that has full rank on W. Given a vector $v \in \Pi_{W_0}(\mathbb{Z}^n) \setminus \{\mathbf{0}_n\}$ with $\|v\|_{A^{-1}} \leq 1/2$ then there exists a hyperplane $P \not\supseteq W$ such that $E \cap \mathbb{Z}^n \subseteq P \cap W$.*

4   Cutting Plane Method

In Section 4.1, we introduce the definition of log-barrier and volumetric center. In Section 4.2, we introduce leverage score and related notations. In Section 4.3, we present the convergence lemma. In Section 4.4, we present the sandwiching lemma. In Section 4.5, we show the closeness between Hessian of log-barrier and covariance matrix. In Section 4.6, we show the stability of approximate center. In Section 4.7, we show the closeness between approximate center and true center. In Section 4.8, we present a novel slicing lemma. In Section 4.9, we present the dynamic leverage score maintenance data structure. In Section 4.10, we present our main lemma.

4.1   Log-barrier and Volumetric Center We start with defining log-barrier, and it has been widely used in convex optimization [Ren88, NN94, JKL+20, HJS+22, JNW22, SYZ23, LSZ+23].

DEFINITION 4.1. (LOG-BARRIER) *Let $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. Let $a_i^\top$ denote the i-th row of A. The* log-barrier *is defined as*

$$\phi(x) = \sum_{i=1}^m -\ln(a_i^\top x - b_i)$$

for $x \in \mathbb{R}^n$.

Let K be the bounded full-dimensional polytope $L = \{x : Ax \geq b\}$ where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and $x \in \mathbb{R}^n$.

DEFINITION 4.2. (HESSIAN AND VOLUMETRIC) *Given $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. Let $H(x)$ be defined as*

$$H(x) = \sum_{i=1}^m \frac{a_i a_i^\top}{(a_i^\top x - b_i)^2}$$

where $a_i^\top$ denotes the i-th row of A.
Let (volumetric) function $F(x)$ be as

$$F(x) := \frac{1}{2} \ln(\det(H(x)))$$

$H(x)$ is the Hessian of the logarithmic barrier function $\sum_{i=1}^m -\ln(a_i^\top x - b_i)$ and is positive definite for all x in the interior of K.

DEFINITION 4.3. (VOLUMETRIC CENTER) *We define the volumetric center of K as*

$$\mathrm{vc}(K) := \arg\min_{x \in K} F(x).$$

Observe that F is a convex function, hence one can run Newton-type algorithm to approximate $\mathrm{vc}(K)$ very fast.

4.2 Leverage Score of Log Hessian We start with some definitions.

DEFINITION 4.4. *We define $\sigma_i(x)$ to be the i-th leverage score of matrix $H(x)$ as*

$$\sigma_i(x) = \frac{a_i^\top H(x)^{-1} a_i}{(a_i^\top x - b_i)^2}, \quad \forall i \in [m].$$

Using σ_i, we can write the gradient of F in an convenient form:

$$\nabla F(x) = -\sum_{i=1}^m \sigma_i(x) \frac{a_i}{a_i^\top x - b_i}.$$

DEFINITION 4.5. *We define $Q(x)$ as*

$$Q(x) = \sum_{i=1}^m \sigma_i(x) \frac{a_i a_i^\top}{(a_i^\top x - b_i)^2}.$$

where $\sigma_i(x)$ is defined as Definition 4.4.

It is well-known that $Q(x)$ is a good approximation of $\nabla^2 F(x)$:

LEMMA 4.1. (LEMMA 3 OF [VAI89]) *For any $x \in \mathbb{R}^n$, we have*

$$Q(x) \preceq \nabla^2 F(x) \preceq 5Q(x).$$

Finally, we define $\mu(x)$ which quantifies:

DEFINITION 4.6. *Let $\mu(x)$ be the largest number λ such that*

$$Q(x) \succeq \lambda H(x).$$

The following lemma provides an upper bound on $\mu(x)$:

LEMMA 4.2. (LEMMA 4 OF [VAI89]) *For any $x \in K$, we have*

$$\frac{1}{4m} \leq \mu(x) \leq 1.$$

Further, we have

$$\mu(x) \geq \min_{i \in [m]} \{\sigma_i(x)\}.$$

4.3 Volume Shrinking The following result (Lemma 4.3) bounds the progress of adding or deleting a plane of Vaidya's CPM.

LEMMA 4.3. *Let $\delta \leq 10^{-4}$ and $\epsilon \leq 10^{-3}\delta$ be some constants and let ρ^k denote the value of $F(\mathrm{vc}(K))$ at the beginning of k-th iteration. Then at the beginning of each iteration there exists an z satisfying the condition*

$$F(z) - F(\mathrm{vc}(K)) \leq \epsilon^4 \mu(\mathrm{vc}(K)).$$

Furthermore, the following statements hold:

- *If $\min_{i \in [m]}\{\sigma_i(z)\} \geq \epsilon$ at k-th iteration then*

$$\rho^{k+1} - \rho^k \geq \frac{(\delta\epsilon)^{1/2}}{5}.$$

- *Otherwise, we have*

$$\rho^k - \rho^{k+1} \leq 5\epsilon.$$

Next, in Lemma 4.4, we show that after $T = O(n \log m)$ iterations, the volume of the resulting convex body is only $(\frac{1}{m})^n$ fraction of the original convex body.

LEMMA 4.4. *Let $K \subseteq \mathbb{R}^n$ be a convex body with non-empty interior. Suppose we run CUTTINGPLANEMETHOD for $T = O(n \log m)$ iterations, then we obtain a convex body K' such that*

$$\mathrm{vol}(K') \leq \left(\frac{1}{m}\right)^n \cdot \mathrm{vol}(K)$$

Proof. Let π^k denote the volume of the polytope K at the beginning of k-th iteration. Note that $\mathrm{vol}(K') = \pi^T$. Using Lemma 4.3 we shall obtain an upper bound on π^k and show that after $O(n \log m)$ iterations the volume decreases by a factor of $(\frac{1}{m})^n$. First, we show that

$$\rho^k \geq \rho^0 + \frac{1}{2}k\epsilon.$$

Since K is bounded, the number of bounding planes is at least $n + 1$ and to start with this number is exactly $n + 1$. Thus by the k-th iteration the case of adding a plane must have occurred at least as often as the case of deleting a plane otherwise the number of planes would have fallen below $n + 1$. So by the k-th iteration adding a plane must happen at least $k/2$ times and removing a plane must happen at most $k/2$ times. Hence

$$\rho^k - \rho^0 \geq \frac{1}{2}\left(\frac{1}{5}k(\delta\epsilon)^{1/2} - 5k\epsilon\right) \geq \frac{1}{2}k\epsilon$$

where the last step follows from $\epsilon \leq 10^{-3}\delta$.

Set $k = T$, we have

$$(4.1) \qquad \rho^T - \rho^0 \geq \frac{1}{2}T\epsilon.$$

We note that by Lemma 4.5, the polytope K contains $E(\mathrm{vc}(K), H_K)$ so its π^0 can be lower bounded by the volume of $E(\mathrm{vc}(K), H_K)$. Therefore,

$$
\begin{aligned}
\ln(\pi^0) \geq \ &-\frac{1}{2}\ln(\det(H_K)) - n \log n \\
= \ &-\rho^0 - n \log n.
\end{aligned}
$$
(4.2)

To obtain an upper bound on π^T, we note that if x^* is the point that maximizes the logarithmic barrier over K', then

$$K' \subseteq \{x : (x - x^*)^\top H(x^*)(x - x^*) \leq m^2\}.$$

Then from the relationship between determinants and volume it follows that

$$
\begin{aligned}
\mathrm{vol}(K') \leq \ &(2m)^n (\det(H(x^*)))^{-1/2} \\
\leq \ &(2m)^n (\det(H(\mathrm{vc}(K'))))^{-1/2} \\
\leq \ &(2m)^n \exp(-F(\mathrm{vc}(K'))).
\end{aligned}
$$
(4.3)

where the last step follows from definitions of H and F (see Definition 4.2).

Since $\sum_{i=1}^{m} \sigma_i(x) = n$ (by Fact 3.1), the case of deleting a plane is forced to happen at an iteration if the number of planes defining K is greater than n/ϵ and hence m never exceeds n/ϵ.

Let us bound the difference between $\ln(\pi^T)$ and $\ln(\pi^0)$:

$$
\begin{aligned}
\ln(\pi^T) - \ln(\pi^0) &\leq n\log(2m) - \rho^T - \ln(\pi^0) \\
&\leq n\log(2m) - \rho^0 - \frac{1}{2}T\epsilon - \ln(\pi^0) \\
&\leq n\log(2m) - \frac{1}{2}T\epsilon + n\log n \\
&\leq -n\log m,
\end{aligned}
$$

the first step is by Eq. (4.3), the second step is by Eq. (4.1), the third step is by Eq. (4.2) and the last step is by $T = O(n\log m)$.

Exponentiating both sides, we obtain

$$
\mathrm{vol}(K') \leq \left(\frac{1}{m}\right)^n \cdot \mathrm{vol}(K).
$$

Thus, we complete the proof. $\qquad\square$

4.4   Sandwiching Lemma We derive some sandwiching conditions regarding the ellipsoid induced by the Hessian of log barrier.

LEMMA 4.5. (SANDWICH CONVEX BODY VIA LOG HESSIAN) *Let $K = \{x : Ax \geq b\}$ be a polytope where $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$. Let $H(x)$ be defined as Definition 4.2. Then for any $x \in K$, we have that*

$$
E(x, H(x)) \subseteq K,
$$

and the following upper bound

$$
K \subseteq 2m^{1.5}n \cdot E(\mathrm{vc}(K), H(\mathrm{vc}(K))).
$$

Proof. Let us first consider the ellipsoid contained in K. By definition, we have that

$$
E(x, H(x)) = \{y : (y-x)^\top H(x)(y-x) \leq 1\},
$$

without loss of generality assume $x = 0$, then $y \in E(x, H(x))$ means

$$
\sum_{i=1}^{m} \frac{(a_i^\top y)^2}{b_i^2} \leq 1,
$$

which means that for any $i \in [m]$, it holds that

$$
(a_i^\top y)^2 \leq b_i^2.
$$

Since $x \in K$, we must have that $b_i \leq 0$ for all $i \in [m]$. Hence, the square condition only requires that $|a_i^\top y| \leq |b_i|$, so if $a_i^\top y \geq 0$, then clearly $y \in K$. Otherwise due to the absolute value constraint, it must be the case that $a_i^\top y \geq b_i$. This concludes the proof of $E(x, H(x)) \subseteq K$.

For the ellipsoid that contains K, we note that the volumetric function F scaled by $\sqrt{m}$ is also a self-concordant barrier with complexity parameter $\sqrt{m}n$ [Vai89, VA93, Ans97] and scaling does not change the minimizer, therefore we have

$$
K \subseteq mn \cdot E(\mathrm{vc}(K), \nabla^2 F(\mathrm{vc}(K))),
$$

recall that $\nabla^2 F(\mathrm{vc}(K)) \succeq \mu(\mathrm{vc}(K)) \cdot H(\mathrm{vc}(K))$ (due to Lemma 4.1 and Definition 4.6) and $\frac{1}{4m} \leq \mu(\mathrm{vc}(K)) \leq 1$ (See Lemma 4.2), we conclude that

$$
\frac{1}{4m} \cdot H(\mathrm{vc}(K)) \preceq \nabla^2 F(\mathrm{vc}(K)),
$$

thus, $E(\mathrm{vc}(K), \nabla^2 F(\mathrm{vc}(K))) \subseteq 2\sqrt{m} \cdot E(\mathrm{vc}(K), H(\mathrm{vc}(K)))$ and we conclude the desired result. $\qquad\square$

4.5 Closeness of Log Hessian and Covariance Note that Lemma 3.1 and Lemma 4.5 together imply the spectral approximation between the matrix $H(\text{vc}(K))$ (See Definition 4.2) and $\text{Cov}(K)$ (See Definition 3.3).

LEMMA 4.6. (CLOSENESS OF LOG HESSIAN AND COVARIANCE) *Let* $K = \{x : Ax \geq b\}$ *be a polytope for* $A \in \mathbb{R}^{m \times n}$ *and* $b \in \mathbb{R}^m$. *Then*

$$\frac{1}{4m^3n^2} \cdot H(\text{vc}(K)) \preceq \text{Cov}(K)^{-1} \preceq 4n^2 \cdot H(\text{vc}(K)).$$

Proof. The proof relies on two sandwiching lemmas.

By Lemma 3.1, we have that

$$E(\text{cg}(K), \text{Cov}(K)^{-1}) \subseteq K \subseteq 2n \cdot E(\text{cg}(K), \text{Cov}(K)^{-1}),$$

by Lemma 4.5,

$$E(\text{vc}(K), H(\text{vc}(K))) \subseteq K \subseteq 2m^{1.5}n \cdot E(\text{vc}(K), H(\text{vc}(K))).$$

We thus have

$$E(\text{cg}(K), \text{Cov}(K)^{-1}) \subseteq 2m^{1.5}n \cdot E(\text{vc}(K), H(\text{vc}(K))),$$
$$E(\text{vc}(K), H(\text{vc}(K))) \subseteq 2n \cdot E(\text{cg}(K), \text{Cov}(K)^{-1}).$$

Without loss of generality, let's prove one side containment. Given

$$E(\text{cg}(K), \text{Cov}(K)^{-1}) \subseteq 2m^{1.5}n \cdot E(\text{vc}(K), H(\text{vc}(K))),$$

we note that re-centering and making their centers the same does not change the containment relation, therefore, we have the following:

$$E(\text{Cov}(K)^{-1}) \subseteq 2m^{1.5}n \cdot E(H(\text{vc}(K))),$$

this directly implies the Lowener ordering

$$\frac{1}{4m^3n^2} \cdot H(\text{vc}(K)) \preceq \text{Cov}(K)^{-1}$$

Following a similar approach, we can show that

$$\text{Cov}(K)^{-1} \preceq 4n^2 \cdot H(\text{vc}(K))$$

this completes the proof. $\square$

4.6 Stability of Approximate Center In this section, we prove another useful lemma that concerns properties of the Hessian matrix of the log barrier. It compares the Hessian evaluated at the volumetric center and an approximate center.

Before proceeding to the second lemma, we define some notions.

DEFINITION 4.7. *Let* $K = \{x : Ax \geq b\}$ *be a polytope with* m *constraints of dimension* n. *Let* $r \in (0,1)$. *Define* $\Sigma(x,r)$ *to be the region*

$$\Sigma(x,r) = \left\{ y : \forall i \in [m], 1 - r \leq \frac{a_i^\top y - b_i}{a_i^\top x - b_i} \leq 1 + r \right\}.$$

Recall the CPM of [Vai89] maintains an approximate volumetric center that will be updated via Newton's method. By performing Newton's method, it can be guaranteed that evaluating function F on the approximate center is not too far away from the volumetric center. The next lemma shows that the approximate center is also in the region $\Sigma(\text{vc}(K), r)$ for proper r.

LEMMA 4.7. (LEMMA 10 OF [VAI89]) *Let F be defined as in Definition 4.2. Let Q be defined as in Definition 4.5. Let $\epsilon \leq 10^{-4}$ be a small constant and let z be a point in K such that $F(z) - F(\mathrm{vc}(K)) \leq \epsilon\sqrt{\mu(\mathrm{vc}(K))}$. Then we have*

- $z \in \Sigma(\mathrm{vc}(K), 5\epsilon^{1/2})$.

- $\mu(\mathrm{vc}(K)) \leq 1.5\mu(z)$.

- $0.1\nabla F(z)^\top Q(z)^{-1}\nabla F(z) \leq F(z) - F(\mathrm{vc}(K)) \leq 2\nabla F(z)^\top Q(z)^{-1}\nabla F(z)$.

In the following Lemma, we show the stability of approximate center.

LEMMA 4.8. (STABILITY OF APPROXIMATE CENTER) *Let K be a convex polytope with $\mathrm{vc}(K)$ being its volumetric center. Let H and F be defined as in Definition 4.2. Let $\epsilon \in (0, 10^{-4})$ be a small constant. Let $z \in K$ be a point such that $F(z) - F(\mathrm{vc}(K)) \leq \epsilon\sqrt{\mu(\mathrm{vc}(K))}$, then we have*

$$(1 - 30\epsilon) \cdot H(\mathrm{vc}(K)) \preceq H(z) \preceq (1 + 30\epsilon) \cdot H(\mathrm{vc}(K)).$$

Proof. By Lemma 4.7, we know that $z \in \Sigma(\mathrm{vc}(K), 5\epsilon^{1/2})$, which means that for any $i \in [m]$, we have that

$$\frac{a_i^\top z - b_i}{a_i^\top \mathrm{vc}(K) - b_i} \in [1 - 5\epsilon^{1/2}, 1 + 5\epsilon^{1/2}].$$

Recall we define the H (Definition 4.2) matrix as

$$H(x) = \sum_{i=1}^m \frac{a_i a_i^\top}{(a_i^\top x - b_i)^2},$$

which means for different arguments, the only part differs is the coefficients $(a_i^\top x - b_i)^2$. If we can approximate the coefficients well, then we can show $H(z)$ and $H(\mathrm{vc}(K))$ are spectrally close. Without loss of generality we prove the upper bound, lower bound is similar:

$$
\begin{aligned}
H(z) &= \sum_{i=1}^m \frac{a_i a_i^\top}{(a_i^\top z - b_i)^2} \\
&\preceq \sum_{i=1}^m \frac{a_i a_i^\top}{(1 - 5\epsilon^{1/2})^2 (a_i^\top \mathrm{vc}(K) - b_i)^2} \\
&\preceq \frac{1}{1 - 10\epsilon} \sum_{i=1}^m \frac{a_i a_i^\top}{(a_i^\top \mathrm{vc}(K) - b_i)^2} \\
&= \frac{1}{1 - 10\epsilon} H(\mathrm{vc}(K)) \\
&\preceq (1 + 30\epsilon) H(\mathrm{vc}(K)),
\end{aligned}
$$

where the first step follows from definition H, the second step follows from $(a_i^\top z - b_i)^2 \geq (1 - 5\epsilon^{1/2})^2(a_i^\top \mathrm{vc}(K) - b_i)^2$, the third step follows from $(1 - 5\epsilon^{1/2})^2 \geq 1 - 10\epsilon$, the forth step follows from definition of H, and the last step follows from $(1 - 10\epsilon)(1 + 30\epsilon) \geq 1$ when $\epsilon \in (0, 0.01)$. This completes our proof. $\square$

4.7 Closeness of Approximate and True Center The goal of this section is to prove Lemma 4.9, which states that under the induced-$H(\mathrm{vc}(K))$ norm, the approximate center and the volumetric center is at most ϵm away. We will later show that this discrepancy is in fact acceptable for our algorithm to make progress.

LEMMA 4.9. (CLOSENESS OF APPROXIMATE AND TRUE CENTER IN TERMS OF H NORM) *Let K be a convex polytope with $\mathrm{vc}(K)$ being its volumetric center and let $\epsilon \in (0, 10^{-4})$. Let H and F be defined as in Definition 4.2. Let $z \in K$ be a point such that $F(z) - F(\mathrm{vc}(K)) \leq \frac{\epsilon}{5}\sqrt{\mu(\mathrm{vc}(K))}$, then we have*

$$\|z - \mathrm{vc}(K)\|_{H(\mathrm{vc}(K))} \leq \epsilon m.$$

Proof. Throughout the proof, we set ϵ to $\epsilon/5$.

By Lemma 4.7, we have that $z \in \Sigma(\mathrm{vc}(K), 5\epsilon^{1/2})$, which means

$$(1 - \epsilon^{1/2}) \cdot (a_i^\top \mathrm{vc}(K) - b_i) \leq a_i^\top z - b_i \leq (1 + \epsilon^{1/2}) \cdot (a_i^\top \mathrm{vc}(K) - b_i).$$

This means that if $a_i^\top z \geq a_i^\top \mathrm{vc}(K)$,

$$
\begin{aligned}
a_i^\top z - a_i^\top \mathrm{vc}(K) &= (a_i^\top z - b_i) - (a_i^\top \mathrm{vc}(K) - b_i) \\
&\leq \epsilon^{1/2}(a_i^\top \mathrm{vc}(K) - b_i).
\end{aligned}
$$

On the other hand if $a_i^\top z < a_i^\top \mathrm{vc}(K)$,

$$
\begin{aligned}
a_i^\top \mathrm{vc}(K) - a_i^\top z &= (a_i^\top \mathrm{vc}(K) - b_i) - (a_i^\top z - b_i) \\
&\leq (a_i^\top \mathrm{vc}(K) - b_i) - (1 - \epsilon^{1/2})(a_i^\top \mathrm{vc}(K) - b_i) \\
&= \epsilon^{1/2}(a_i^\top \mathrm{vc}(K) - b_i).
\end{aligned}
$$

We thus have shown that $|a_i^\top (z - \mathrm{vc}(K))| \leq \epsilon^{1/2}(a_i^\top \mathrm{vc}(K) - b_i)$. We proceed to measure the squared-$H(\mathrm{vc}(K))$ norm of $z - \mathrm{vc}(K)$:

$$
\begin{aligned}
\|z - \mathrm{vc}(K)\|^2_{H(\mathrm{vc}(K))} &= \sum_{i=1}^m \frac{(a_i^\top z - a_i^\top \mathrm{vc}(K))^2}{(a_i^\top \mathrm{vc}(K) - b_i)^2} \\
&\leq \sum_{i=1}^m \frac{\epsilon(a_i^\top \mathrm{vc}(K) - b_i)^2}{(a_i^\top \mathrm{vc}(K) - b_i)^2} \\
&= \epsilon m.
\end{aligned}
$$

This completes the proof of the lemma. $\square$

4.8 High Dimensional Slicing Lemma We present a novel high dimensional slicing lemma that uses Hessian of log barrier, instead of the covariance matrix as in [Jia21]. It incurs an extra $(mn)^{O(k)}$ term, but as we will see later, this does not affect the total number of oracle calls too much.

LEMMA 4.10. (HIGH DIMENSIONAL SLICING LEMMA, VOLUMETRIC VERSION) *Let K be a convex body, let H_K denote the Hessian matrix of log barrier of K at its volumetric center. Let L be a full-rank lattice in $\mathbb{R}^n$. Let W be an $(n - k)$-dimensional subspace of $\mathbb{R}^n$ such that $\dim(L \cap W) = n - k$. Then we have*

$$\frac{\mathrm{vol}(K \cap W)}{\det(L \cap W)} \leq \frac{\mathrm{vol}(K)}{\det(L)} \cdot \frac{k^{O(k)} \cdot (mn)^{O(k)}}{\lambda_1(L^*, H_K^{-1})^k}$$

where L^* is the dual lattice and $\lambda_1(L^*, H_K^{-1})$ is the shortest nonzero vector in L^* under the norm $\|\cdot\|_{H_K^{-1}}$.

Proof. Let v denote the shortest nonzero vector in L^* under the norm $\|\cdot\|_{\mathrm{Cov}(K)}$ and u be the shortest nonzero vector in L^* under the norm $\|\cdot\|_{H_K^{-1}}$, by Lemma 4.6, we know that

$$O\left(\frac{1}{m^3 n^2}\right) \cdot H_K^{-1} \preceq \mathrm{Cov}(K) \preceq O(n^2) \cdot H_K^{-1}.$$

We have

$$
\begin{aligned}
\|v\|_{\mathrm{Cov}(K)} &\geq \frac{1}{2m^{1.5}n} \cdot \|v\|_{H_K^{-1}} \\
&\geq \frac{1}{2m^{1.5}n} \cdot \|u\|_{H_K^{-1}},
\end{aligned}
$$

therefore, we have that

$$\frac{1}{\lambda_1(L^*, \mathrm{Cov}(K))} \leq \frac{2m^{1.5}n}{\lambda_1(L^*, H_K^{-1})}.$$

Using Lemma 3.2, we have

$$\frac{\mathrm{vol}(K \cap W)}{\det(L \cap W)} \leq \frac{\mathrm{vol}(K)}{\det(L)} \cdot \frac{k^{O(k)}}{\lambda_1(L^*, \mathrm{Cov}(K))^k}$$

Finally, combining the above equations we conclude

$$\frac{\mathrm{vol}(K \cap W)}{\det(L \cap W)} \leq \frac{\mathrm{vol}(K)}{\det(L)} \cdot \frac{k^{O(k)} \cdot (mn)^{O(k)}}{\lambda_1(L^*, H_K^{-1})^k}.$$

Thus, we complete the proof. □

4.9 Faster Cutting Plane via Leverage Score Maintenance The vanilla Vaidya's CPM algorithm requires to compute all m leverage scores per iteration, which would require $O(n^\omega)$ time to form a projection matrix. [JLSW20] shows that by carefully designing data structures for leverage score maintenance, each iteration can be improved to $O(n^2)$ amortized time. The following lemma states that to correct the extra error induced by using such data structures, it is sufficient to run for an extra $O(\epsilon^{-1})$ step. This provides guarantee for [JLSW20], and we leverage it to speed up our algorithm.

LEMMA 4.11. (APPROXIMATING FUNCTION VALUE VIA APPROXIMATE LEVERAGE SCORE, LEMMA A.3 OF [JLSW20]) *If the following conditions hold*

- *The Vaidya's Newton step [Vai89] uses exact leverage score σ and runs in $T = O(n \log n)$ iterations to obtain an approximate point z such that*

$$F(z) - F(\mathrm{vc}(K)) \leq 0.1.$$

- *The closeness between true leverage score and approximate leverage score, $\|\widetilde{\sigma} - \sigma\|_2 \leq 1/\log^{O(1)}(n)$*

Then, running Vaidya's Newton step [Vai89] with approximate leverage score $\widetilde{\sigma}$ for $\widetilde{T} = T + O(1/\epsilon)$ iterations, we can obtain a $\widetilde{z}$ such that

$$F(\widetilde{z}) - F(\mathrm{vc}(K)) \leq \epsilon.$$

To obtain various guarantees, we need to find an $z \in K$ with $F(z) - F(\mathrm{vc}(K)) \leq c \cdot \sqrt{\mu(\mathrm{vc}(K))}$ for small constant $c \leq 10^{-4}$. Note that whenever the smallest leverage score is at least ϵ, we only need to run an extra $O(\epsilon^{-1})$ iterations of Newton's step. In the other case, one can show that the old point z is still a good starting point for the Newton's step, therefore running an extra $O(1)$ iterations suffices.

We state the data structure of [JLSW20] for completeness.

LEMMA 4.12. (THEOREM 5.1 OF [JLSW20]) *Given an initial matrix $A \in \mathbb{R}^{m \times n}$ with $m = O(n)$, initial weight $w \in \mathbb{R}_{\geq 0}^m$, there is a randomized data structure that approximately maintains the leverage score*

$$\sigma_i(w) = (\sqrt{W} A (A^\top W A)^{-1} A^\top \sqrt{W})_{i,i}, \quad \forall i \in [m]$$

where $W \in \mathbb{R}^{m \times m}$ is the diagonal matrix that puts $w \in \mathbb{R}^m$ on its diagonal. The data structure uses $O(n^{2+o(1)})$ space and supports the following operations:

- INIT($A \in \mathbb{R}^{m \times n}, w \in \mathbb{R}^m$): *The data structure initializes in $O(n^{\omega+o(1)})$ time.*

- UPDATE($w \in \mathbb{R}^m, u \in \mathbb{R}^n, w_u \in \mathbb{R}, i \in [m], \mathsf{act} \in \{\mathrm{ins}, \mathrm{del}, \mathrm{upd}\}$): *The data structure updates by*

– If $\mathsf{act} = \mathrm{ins}$, *we insert a row u with weight w_u into $(A^{(k-1)}, w^{(k-1)})$ such that*

$$w_u u u^\top \preceq 0.01 (A^{(k-1)})^\top W^{(k-1)} A^{(k-1)}$$

Suppose currently $A^{(k-1)}$ has i_u rows already, we append u to the $(i_u + 1)$-th row of $A^{(k-1)}$ and append w_u to the $(i_u + 1)$-th row of $W^{(k-1)}$. In this case, we ignore the w and i from input parameters.

– If $\mathsf{act} = \mathrm{del}$, *let $i_v = i$, let v, w_v denote the i_v-the row of $(A^{(k-1)}, w^{(k-1)})$. We delete the i_v-th row from $(A^{(k-1)}, w^{(k-1)})$ such that*

$$w_v v v^\top \preceq 0.01 (A^{(k-1)})^\top W^{(k-1)} A^{(k-1)}$$

In this case we ignore the w, u, w_u from input parameters of update function.

– If $\mathsf{act} = \mathrm{upd}$, *we update the weight vector from $w^{(k-1)}$ to $w^{(k)}$ such that*

$$\| \log(w^{(k)}) - \log(w^{(k-1)}) \|_2 \leq 0.01.$$

Here $w^{(k)}$ denote the w from input of update function, and $w^{(k-1)}$ denote the weight we stored from last iteration. In this case, we ignore the u, w_u, i from input parameters of update function.

This step takes amortized $O(n^2)$ time.

- QUERY()*: The data structure outputs an approximate leverage score $\widetilde{\sigma} \in \mathbb{R}^m$ such that*

$$\| \widetilde{\sigma} - \sigma \|_2 \leq O(1 / \log^c(n)),$$

this step takes $O(n)$ time. Here $c > 1$ is some fixed constant.

In our application, we will invoke the data structure in the following fashion: each time the data structure is initialized, it will be updated and queried for a consecutive of $O(n \log n)$ steps, which takes a total of $O(n^3 \log n)$ time. We will then perform such sequence of operations for $O(n)$ times, which leads to a total of $O(n^4 \log n)$ time.

4.10 Main Lemma The meta lemma of this section states that if we invoke $O(n \log m)$ oracle calls to add planes for Vaidya's CPM, we end up with a polytope whose volume is only a fraction of $\left(\frac{1}{m}\right)^n$ of the original polytope.

LEMMA 4.13. *Given a separation oracle SO for a convex function f defined on $\mathbb{R}^n$. Let vc be defined as Definition 4.3. Let H be defined as Definition 4.2. Given a polytope $K \subseteq \mathbb{R}^n$ with m constraints that contains minimizer x^* of f, and an error parameter $\epsilon > 0$, there exists a cutting plane method $\mathrm{CUTTINGPLANEMETHOD}(\mathsf{SO}, K, T, \epsilon)$ with $T = O(n \log(m/\epsilon))$ that uses at most $O(n \log(m/\epsilon))$ calls to SO and an extra $O(n^3 \log(m/\epsilon))$ arithmetic operations to output a polytope K' with at most $O(n/\epsilon)$ constraints, an approximate volumetric center z of K' and Hessian matrix H of log barrier of K' such that the following holds:*

- *Part 1. $x^* \in K'$ and K' is the intersection of K with T hyperplanes outputted by SO.*

- *Part 2. $E(\mathrm{vc}(K'), H(\mathrm{vc}(K'))) \subseteq K' \subseteq O(mn) \cdot E(\mathrm{vc}(K'), H(\mathrm{vc}(K')))$.*

- *Part 3. $\mathrm{vol}(K') \leq (\frac{1}{m})^n \cdot \mathrm{vol}(K)$.*

- *Part 4. $\| z - \mathrm{vc}(K') \|_{H(\mathrm{vc}(K'))} \leq \epsilon m$.*

- *Part 5. $(1 - \epsilon) \cdot H(\mathrm{vc}(K')) \preceq H(z) \preceq (1 + \epsilon) \cdot H(\mathrm{vc}(K'))$.*

Proof. We prove the lemma item by item.

For **Part 1**, it is implied by the original Vaidya's algorithm as in [Vai89].

Part 2 is due to the sandwiching lemma for polytope as in Lemma 4.5.

Part 3 is owing to Lemma 4.4.

For **Part 4**, we prove it in Lemma 4.9.

For **Part 5**, we show it in Lemma 4.8.

Regarding the runtime, we will use the data structure of Lemma 4.12 for a consecutive of T operations, which takes a total of $O(n^{\omega+o(1)} + n^3 \log m) = O(n^3 \log m)$ arithmetic operations. Note that each iteration the query guarantees the approximate leverage score satisfy $\|\widetilde{\sigma} - \sigma\|_2 \leq O(1/\log^c(n))$, by Lemma 4.11, we only need to run extra $O(1)$ iterations of Newton's step to obtain an approximate volumetric center z with desired guarantee. Thus, the overall arithmetic operation count is $O(n^3 \log m)$.

□

5 Efficient Minimization via Fast Cutting and Lazy Width Measurement

In Section 5.1, we present our main algorithm, Algorithm 1. In Section 5.2, we present the correctness of our algorithm. In Section 5.3, we show the oracle complexity of our algorithm. In Section 5.4, we show the overall running time of our algorithm. In Section 5.5, we summarize our main result.

5.1 Our Algorithm Our algorithm is a mixture of efficient cutting plane method of [JLSW20], fast shortest vector algorithm of [NS16] and a novel adaptation and simplification of [Jia22]. The algorithm maintains an affine subspace W, a lattice Λ and a polytope K. The algorithm then proceeds as follows: it computes the approximate shortest vector on Λ with respect to H_K^{-1} norm, where H_K is the Hessian of log barrier function at an approximate volumetric center. If the H_K^{-1} norm of the shortest vector is relatively large, the algorithm performs a sequence of CuttingPlaneMethod for $T = O(n \log n)$ rounds.

Algorithm 1 Our Algorithm

1: **procedure** Main(SO, R)   ▷ Theorem 1.1
2:     $m \leftarrow 2n$
3:     $W \leftarrow \mathbb{R}^n$ be an affine subspace
4:     $K \leftarrow B_\infty(R)$ be a polytope   ▷ K can be parameterized by $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$
5:     $\Lambda \leftarrow \mathbb{Z}^n$ be a lattice
6:     $x_K \leftarrow 0$ be the approximate volumetric center
7:     $H_K \leftarrow \sum_{i=1}^m \frac{a_i a_i^\top}{(a_i^\top x_K - b_i)^2}$ be the Hessian of log barrier
8:     $T \leftarrow O(n \log m), \epsilon \leftarrow 0.01$
9:     **while** $\dim(W) > 1$ **do**
10:         $v \leftarrow$ FasterShortestVector(Λ, H_K^{-1})   ▷ Theorem 3.2
11:         **if** $\|v\|_{H_K^{-1}} \geq 2^{-100n \log n}$ **then**
12:             $(K', x_{K'}, H_K') \leftarrow$ CuttingPlaneMethod(SO, K, T, ϵ)   ▷ Lemma 4.13
13:             $K \leftarrow K', x_K \leftarrow x_{K'}, H_K \leftarrow H_{K'}$
14:         **else**
15:             Find $z \in \mathbb{Z}^n$ such that $v = \Pi_{W - x_K}(z)$
16:             $P \leftarrow \{y : v^\top y = (v - z)^\top x_K + [z^\top x_K]\}$
17:             $W \leftarrow W \cap P$
18:             Let $E(W, a)$ be the ellipsoid $3m^{1.5}n \cdot E(x_K, H_K) \cap P$
19:             $K \leftarrow w + A^{-1/2}B_\infty(1)$
20:             $x_K \leftarrow w, H_K = \sum_{i=1}^m \frac{a_i a_i^\top}{(a_i^\top x_K - b_i)^2}$
21:             Construct hyperplane $P_0 \leftarrow \{y : v^\top y = 0\}$
22:             $\Lambda \leftarrow \Pi_{P_0}(\Lambda)$
23:         **end if**
24:     **end while**
25:     Find integral minimizer $x^* \in \mathbb{Z}^n \cap K$
26:     **return** x^*
27: **end procedure**

5.2 Correctness of Our Algorithm We prove the output guarantee of our algorithm (Algorithm 1) in Lemma 5.1.

LEMMA 5.1. (FORMAL VERSION OF THEOREM 1.1, OUTPUT GUARANTEE PART) *Given a separation oracle* SO *for a convex function f defined on $\mathbb{R}^n$ and a γ-approximation algorithm* APPROXSHORTESTVECTOR *for the shortest vector problem which takes $\mathcal{T}_{\text{APPROXSV}}$ arithmetic operations. Suppose the set of minimizers K^* of f is contained in a box of radius R and satisfies all extreme points of K^* are integral, Algorithm 1 finds an integral minimizer of f.*

Proof. Recall that lattice projection and the dual of lattice projections are defined as in Definition 3.2 and Definition 3.3.

We start by showing that the lattice Λ is the orthogonal projection of $\mathbb{Z}^n$ onto the subspace W_0 via induction. In the beginning of each iteration we have $K \subseteq W$ and $\Lambda \subseteq W_0$ where W_0 is a translation of W passing through the origin. In the beginning of the algorithm, $\Lambda = \mathbb{Z}^n$ and $W = \mathbb{R}^n$, so $\Lambda = \Pi_{W_0}(\mathbb{Z}^n)$. Note that the lattice Λ and subspace W are only updated in the dimension reduction step of Algorithm 1. For inductive step, let Λ_{t-1} to denote the lattice at the $(t-1)^{\text{th}}$ dimension reduction step and $\Lambda_{t-1} = \Pi_{W_0}(\mathbb{Z}^n)$ and we prove for t.

$$
\begin{aligned}
\Lambda_t &= \Pi_{P_0}(\Lambda_{t-1}) \\
&= \Pi_{P_0}(\Pi_{W_0}(\mathbb{Z}^n)) \\
&= \Pi_{W_0 \cap P_0}(\Pi_{W_0}(\mathbb{Z}^n)) \\
&= \Pi_{W_0 \cap P_0}(\mathbb{Z}^n),
\end{aligned}
$$

to see the third equality, we note that P_0 is the orthogonal subspace of v and $v \in W_0$. As initially $W_0 = \mathbb{R}^n$, we can inductively show that at time t, the subspace W_0 is the orthogonal complement to $v_1, \ldots, v_{t-1}$ where v_i is the (approximate) shortest vector we use in iteration i. As $\Lambda_{t-1} = \Pi_{W_0}(\mathbb{Z}^n)$, it is a subspace orthogonal to $\text{span}(v_1, \ldots, v_{t-1})$. Projecting this subspace onto P_0 makes it orthogonal to v_t. Thus, first projecting onto W_0 then projecting onto P_0 is equivalent to first projecting onto W_0 then projecting onto $W_0 \cap P_0$. The last equality follows from $W_0 \cap P_0$ is a subspace of W_0. This completes the proof of the lattice property.

Now we are ready to show that Algorithm 1 indeed finds the integral minimizer. Assuming f has a unique minimizer $x^* \in \mathbb{Z}^n$, we note that CUTTINGPLANEMETHOD preserves x^*, so it suffices to show that the dimension reduction step also preserves x^*. We show that in fact, the dimension reduction step preserves all integral points in K.

Lemma 4.5 gives the following sandwiching condition:

$$
\frac{1}{2} \cdot E(\text{vc}(K), H(\text{vc}(K))) \subseteq K \subseteq 2m^{1.5}n \cdot E(\text{vc}(K), H(\text{vc}(K))).
$$

Recall that we set H_K to be a $(1 \pm \epsilon)$-spectral approximation to $H(\text{vc}(K))$:

$$
(1 - \epsilon) \cdot H_K \preceq H(\text{vc}(K)) \preceq (1 + \epsilon) \cdot H_K
$$

We know that

$$
\|x_K - \text{vc}(K)\|_{H(\text{vc}(K))} \leq \epsilon m,
$$

this means that $\text{vc}(K) \in 2\epsilon^{1/2} m^{1/2} \cdot E(x_K, H(x_K))$. Consequently, we have

$$
(5.4) \qquad\qquad (\text{vc}(K) - x_K)^\top H(x_K)(\text{vc}(K) - x_K) \leq \epsilon m.
$$

Let $y \in K$, by the sandwiching condition, we also have $y \in 2m^{1.5}n \cdot E(\text{vc}(K), H(x_K))$ and subsequently

$$
(5.5) \qquad\qquad (y - \text{vc}(K))^\top H(x_K)(y - \text{vc}(K)) \leq 4m^3 n^2.
$$

Combining Eq. (5.4) and (5.5), we conclude

$$
\begin{aligned}
&(y - x_K)^\top H(x_K)(y - x_K) \\
&= \|y - x_K\|_{H(x_K)}^2 \\
&\leq 2\|\operatorname{vc}(K) - x_K\|_{H(x_K)}^2 + 2\|y - \operatorname{vc}(K)\|_{H(x_K)}^2 \\
&\leq 8m^3 n^2 + 2\epsilon m \\
&\leq 9m^3 n^2.
\end{aligned}
$$

We thus have shown that

$$
K \subseteq O(m^{1.5} n) \cdot E(x_K, H_K).
$$

Now we proceed to show that each dimension reduction iteration preserves all integral points in K. We have

$$
K \cap \mathbb{Z}^n \subseteq 3m^{1.5} n \cdot E(x_K, H_K) \cap \mathbb{Z}^n .
$$

Since $\|v\|_{H_K^{-1}} \leq 1/(10n)$ is satisfied in a dimension reduction step, we can invoke Lemma 3.3, which states that all integral points in $3m^{1.5} n \cdot E(x_K, H_K)$ lie on the hyperplane given by

$$
P = \{y : v^\top y = (v - z)^\top x_K + [z^\top x_K]\}.
$$

Thus, we have $K \cap \mathbb{Z}^n \subseteq K \cap P$ and this finishes the proof of the lemma. $\quad\square$

5.3 Oracle Complexity We prove the oracle complexity of Algorithm 1.

LEMMA 5.2. (ORACLE COMPLEXITY PART OF THEOREM 5.1) *Given a separation oracle* SO *for a convex function f on $\mathbb{R}^n$ such that the set of minimizers K^* of f is contained in a box of radius R and all extreme points of K^* are integral, then there exists a randomized algorithm (Algorithm 1) that outputs an integral minimizer of f with at most $O(n^2 \log n + n \log(\gamma n R))$ calls to* SO *with high probability.*

Proof. We consider the potential function

$$
\Phi := \log(\operatorname{vol}(K) \cdot \det(\Lambda)).
$$

In the beginning, $\Phi = \log(\operatorname{vol}(B_\infty) \cdot \det(I)) = n \log R$. A sequence of $O(n \log m)$ calls to CUTTING-PLANEMETHOD reduce the volume by a factor of $(\frac{1}{m})^n = (\frac{1}{2})^{n \log m}$, consequently the potential decreases by $n \log m$, additively.

Without loss of generality, let us assume we have a maximal sequence of dimension reduction steps at $t = 1, 2, \ldots, k+1$.

Note that the potential at the beginning of this maximal sequence of dimension reduction iteration is

$$
\begin{aligned}
e^{\Phi^{(0)}} &= \operatorname{vol}(K^{(0)}) \cdot \det(\Lambda^{(0)}) \\
&= \frac{\operatorname{vol}(K^{(0)})}{\det((\Lambda^{(0)})^*)}.
\end{aligned}
$$

Between t and $t+1$, we note that $K^{(t+1)}$ is designed in the following fashion: first compute $K^{(t)} \cap W^{(t+1)}$, then consider its outer ellipsoid $E(w, A) = 3m^{1.5} n \cdot E(x_K, H_K) \cap P$, note that this blows up the volume by a factor of $n^{O(n)}$. Finally, set K to be the hyperrectangle containing $E(w, A)$ which is $w + A^{-1/2} B_\infty(1)$. This again blows up the volume by a factor of $n^{O(n)}$.

In the beginning of $t = 1$, we have $W^{(1)} = W^{(0)}$ and $K^{(1)} \subseteq W^{(1)}$. By the proceeding discussion, we have that

$$
\operatorname{vol}(K^{(2)}) \leq n^{c_1 n} \cdot \operatorname{vol}(K^{(1)} \cap W^{(2)})
$$

for some absolute constant c_1, similarly, we can upper bound the volume of $K^{(3)}$ as follows:

$$\text{vol}(K^{(3)}) \leq n^{c_2 n} \cdot \text{vol}(K^{(2)} \cap W^{(3)})$$
$$\leq n^{c_1 c_2 n} \cdot \text{vol}(K^{(1)} \cap W^{(3)}),$$

recursively apply this inequality, we conclude

$$\text{vol}(K^{(k+1)}) \leq n^{O(nk)} \cdot \text{vol}(K^{(1)} \cap W^{(k+1)})$$

The potential after this sequence of dimension reduction iterations is

$$e^{\Phi^{(k+1)}} = \text{vol}(K^{(k+1)}) \cdot \det(\Pi_{W_0^{(k+1)}}(\Lambda^{(0)}))$$
$$= n^{O(nk)} \cdot \text{vol}(K^{(1)} \cap W^{(k+1)}) \cdot \det(\Pi_{W_0^{(k+1)}}(\Lambda^{(0)}))$$
$$= n^{O(nk)} \cdot \frac{\text{vol}(K^{(1)} \cap W^{(k+1)})}{\det(\Pi_{W_0^{(k+1)}}((\Lambda^{(0)}))^*)}$$
$$= n^{O(nk)} \cdot \frac{\text{vol}(K^{(1)} \cap W^{(k+1)})}{\det((\Lambda^{(0)})^* \cap W_0^{(k+1)})}$$
$$\leq n^{O(nk)} \cdot \frac{\text{vol}(K^{(0)} \cap W^{(k+1)})}{\det((\Lambda^{(0)})^* \cap W_0^{(k+1)})}$$

where the third step is by taking the dual lattice projection (Fact 3.3), the fourth step is due to $(\Pi_{W_0^{(k+1)}}(\Lambda^{(0)}))^* = (\Lambda^{(0)})^* \cap W_0^{(k+1)}$ by Fact 3.2, and the last step is by the volume shrinking after cutting.

Since $W^{(k+1)}$ is a translation of the subspace $W_0^{(k+1)}$, we can apply Lemma 4.10 by taking $L = (\Lambda^{(1)})^*$ to obtain

$$(5.6) \qquad e^{\Phi^{(k+1)}} \leq e^{\Phi^{(0)}} \cdot \frac{n^{O(nk)} \cdot k^{O(k)} \cdot (mn)^{O(k)}}{\lambda_1(\Lambda^{(0)}, (H_K^{(0)})^{-1})^k}$$

It remains to provide a lower bound on $\lambda_1(\Lambda^{(1)}, K^{(1)})$.

As CuttingPlaneMethod is used in iteration t_0, we have

$$\|v^{(0)}\|_{(H_K^{(0)})^{-1}} \geq \min\{\frac{1}{10\gamma n}, 2^{-100 n \log n}\}$$

for the output vector $v^{(0)} \in \Lambda^{(0)}$, and that $\Lambda^{(0)} = \Lambda^{(1)}$ since a cutting plane iteration doesn't change the lattice.

Since the ApproxShortestVector procedure is γ-approximation and that $H_K^{(0)}$ is a $(1 \pm \epsilon)$-spectral approximation to $H_{\text{vc}(K)}^{(0)}$, this implies that

$$(5.7) \qquad \lambda_1(\Lambda^{(0)}, (H_K^{(0)})^{-1}) \geq \frac{\|v^{(0)}\|_{(H_K^{(0)})^{-1}}}{\gamma} \geq \frac{\Omega(1)}{\gamma n^n}$$

Combining Eq. (5.6) and Eq. (5.7), we have

$$e^{\Phi^{(k+1)}} \leq e^{\Phi^{(0)}} \cdot \gamma^{O(k)} \cdot n^{O(nk)}$$

as $m = O(n)$.

This shows that after a sequence of k dimension reduction iterations, the potential increases additively by at most $O(k \log(\gamma n) + nk \log n)$. As there are at most $n - 1$ such iterations the total amount of potential increase is at most $O(n \log(\gamma n) + n^2 \log n)$.

Finally we note that whenever the potential becomes smaller than $-100n\log(100\gamma n)$, Minkowski's first theorem shows the existence of a nonzero vector $v \in \Lambda$ with $\|v\|_{H_K^{-1}} \leq 1/(100\gamma n)$. This implies the γ-approximation algorithm APPROXSHORTESTVECTOR for the shortest vector problem will find a nonzero vector $v' \in \Lambda$ with $\|v'\|_{H_K^{-1}} \leq 1/(100n)$. So the next iteration where we run the LLL algorithm will always reduce the dimension.

Therefore, the sequence of T CUTTINGPLANEMETHOD will be run at most

$$O\Big(\frac{\log(\gamma nR)}{\log n} + n\Big)$$

times.

Since each run of CUTTINGPLANEMETHOD uses $T = O(n\log m) = O(n\log n)$ oracle calls, this also gives the total number of oracle calls

$$O(n^2 \log n + n\log(\gamma nR)).$$

This finishes the proof of the theorem.

□

5.4 Runtime Analysis
The goal of this section is to prove Lemma 5.3.

LEMMA 5.3. (RUNTIME PART OF THEOREM 5.1) *Given a separation oracle* SO *for a convex function f on $\mathbb{R}^n$ such that the set of minimizers K^* of f is contained in a box of radius R and all extreme points of K^* are integral, then there exists a randomized algorithm (Algorithm 1) that outputs an integral minimizer of f, and uses $O(n^4 \log(nR))$ arithmetic operations.*

Proof. As we use γ-APPROXIMATESHORTESTVECTOR with $\gamma = O(2^n)$, the total number of oracle calls is $O(n^2 \log(nR))$.

The dimensional reduction step occurs at most $O(n)$ times. Each dimension reduction step takes at most $O(n^3)$ arithmetic operations, amounts to an $O(n^4)$ arithmetic operations in total.

The CUTTINGPLANEMETHOD is called with $T = O(n\log n)$, so each call uses $O(n^3 \log n)$ arithmetic operations. As such calls happen at most $O(\frac{n\log(nR)}{\log n})$ times, the total arithmetic operations for CUTTINGPLANEMETHOD is at most $O(n^4 \log(nR))$.

Regarding the number of calls to FASTERSHORTESTVECTOR, we note that there are at most $O(n)$ dimension reduction steps, and at most $O(n\log R)$ calls to the sequence of CPM. Thus, the total number of calls to FASTERSHORTESTVECTOR can be upper bounded by $O(n\log R)$, yielding a total of $O(n^4 \log R)$ arithmetic operations. □

5.5 Main Result
The goal of this section is to prove Theorem 5.1.

THEOREM 5.1. (MAIN RESULT, FORMAL VERSION OF THEOREM 1.1) *Given a separation oracle* SO *for a convex function f on $\mathbb{R}^n$ such that the set of minimizers K^* of f is contained in a box of radius R and all extreme points of K^* are integral, then there exists a randomized algorithm (Algorithm 1) that outputs an integral minimizer of f with high probability, and uses*

- $O(n^2 \log(nR))$ *calls to* SO.

- $O(n^4 \log(nR))$ *additional arithmetic operations.*

Proof. The proof follows from directly combining Lemma 5.1, Lemma 5.2 and Lemma 5.3. □

Acknowledgement

References

[AC91] Ilan Adler and Steven Cosares. A strongly polynomial algorithm for a special class of linear programs. *Operations Research*, 39(6):955–960, 1991.

[AKS01] Miklós Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*, STOC '01, 2001.

[Ans97] Kurt M. Anstreicher. Volumetric path following algorithms for linear programming. *Math. Program.*, 1997.

[AV95] David S. Atkinson and Pravin M. Vaidya. A cutting plane algorithm for convex programming that uses analytic centers. *Math. Program.*, 1995.

[BLSS20] Jan van den Brand, Yin Tat Lee, Aaron Sidford, and Zhao Song. Solving tall dense linear programs in nearly linear time. In *STOC*, 2020.

[BV04] Dimitris Bertsimas and Santosh Vempala. Solving convex programs by random walks. *J. ACM*, 2004.

[BWZ16] Christos Boutsidis, David P Woodruff, and Peilin Zhong. Optimal principal component analysis in distributed and streaming models. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 236–249, 2016.

[Chu12] Sergei Chubanov. A strongly polynomial algorithm for linear systems having a binary solution. *Mathematical programming*, 134(2):533–570, 2012.

[Chu15] Sergei Chubanov. A polynomial algorithm for linear optimization which is strongly polynomial under certain conditions on optimal solutions, 2015.

[CLS19] Michael B Cohen, Yin Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. In *Proceedings of the 51st Annual ACM Symposium on Theory of Computing (STOC)*, 2019.

[CM91] Edith Cohen and Nimrod Megiddo. Improved algorithms for linear inequalities with two variables per inequality. In *Proceedings of the twenty-third annual ACM symposium on Theory of Computing*, pages 145–155, 1991.

[CW13] Kenneth L Clarkson and David P Woodruff. Low rank approximation and regression in input sparsity time. In *Proceedings of the forty-fifth annual ACM symposium on Theory of Computing (STOC)*, pages 81–90, 2013.

[DHNV20] Daniel Dadush, Sophie Huiberts, Bento Natura, and László A Végh. A scaling-invariant algorithm for linear programming whose running time depends only on the constraint matrix. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 761–774, 2020.

[Dik67] II Dikin. Iterative solution of problems of linear and quadratic programming. In *Doklady Akademii Nauk*, volume 174, pages 747–748. Russian Academy of Sciences, 1967.

[DSW22] Yichuan Deng, Zhao Song, and Omri Weinstein. Discrepancy minimization in input-sparsity time. *arXiv preprint arXiv:2210.12468*, 2022.

[DVZ21] Dan Dadush, László A Végh, and Giacomo Zambelli. Geometric rescaling algorithms for submodular function minimization. *Mathematics of Operations Research*, 46(3):1081–1108, 2021.

[DWZ23] Ran Duan, Hongxun Wu, and Renfei Zhou. Faster matrix multiplication via asymmetric hashing. In *FOCS*, 2023.

[Edm03] Jack Edmonds. Submodular functions, matroids, and certain polyhedra. In *Combinatorial Optimization—Eureka, You Shrink! Papers Dedicated to Jack Edmonds 5th International Workshop Aussois, France, March 5–9, 2001 Revised Papers*, pages 11–26. Springer, 2003.

[FI03] 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.

[GLS81] Martin Grötschel, László Lovász, and Alexander Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

[GLS84] Martin Grötschel, László Lovász, and Alexander Schrijver. Geometric methods in combinatorial optimization. In *Progress in combinatorial optimization*, pages 167–183. Elsevier, 1984.

[GLS88] Martin Grotschel, Laszlo Lovasz, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization*. Springer, 1988.

[HJS⁺22] Baihe Huang, Shunhua Jiang, Zhao Song, Runzhou Tao, and Ruizhe Zhang. Solving sdp faster: A robust ipm framework and efficient implementation. In *FOCS*, 2022.

[IFF01] 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.

[IO09] 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. SIAM, 2009.

[Iwa03] Satoru Iwata. A faster scaling algorithm for minimizing submodular functions. *SIAM Journal on Computing*, 32(4):833–840, 2003.

[Iwa08] Satoru Iwata. Submodular function minimization. *Math. Program.*, 2008.

[Jia21] Haotian Jiang. Minimizing convex functions with integral minimizers. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms*, 2021.

[Jia22] Haotian Jiang. Minimizing convex functions with rational minimizers. *ACM Journal of the ACM (JACM)*, 2022.

[JKL⁺20] Haotian Jiang, Tarun Kathuria, Yin Tat Lee, Swati Padmanabhan, and Zhao Song. A faster interior point method for semidefinite programming. In *FOCS*, 2020.

[JLSW20] Haotian Jiang, Yin Tat Lee, Zhao Song, and Sam Chiu-wai Wong. An improved cutting plane method for convex optimization, convex-concave games, and its applications. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 944–953, 2020.

[JNW22] Shunhua Jiang, Bento Natura, and Omri Weinstein. A faster interior-point method for sum-of-squares optimization. *49th International Colloquium on Automata, Languages, and Programming (ICALP)*, 2022.

[JSWZ21] Shunhua Jiang, Zhao Song, Omri Weinstein, and Hengjie Zhang. Faster dynamic matrix inverse for faster lps. In *STOC*, 2021.

[Kha80] L.G. Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 1980.

[KLS95] Ravi Kannan, László Lovász, and Miklós Simonovits. Isoperimetric problems for convex bodies and a localization lemma. *Discrete & Computational Geometry*, 13(3):541–559, 1995.

[Lev65] Anatoly Yur'evich Levin. An algorithm for minimizing convex functions. In *Doklady Akademii Nauk*, volume 160, pages 1244–1247. Russian Academy of Sciences, 1965.

[LG24] Francois Le Gall. Faster rectangular matrix multiplication by combination loss analysis. In *SODA*, 2024.

[LLL82] Arjen K Lenstra, Hendrik Willem Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische annalen*, 1982.

[LS14] Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in $O(\sqrt{rank})$ iterations and faster algorithms for maximum flow. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 424–433. IEEE, 2014.

[LS19] Yin Tat Lee and Aaron Sidford. Solving linear programs with sqrt (rank) linear system solves. *arXiv preprint arXiv:1910.08033*, 2019.

[LSW15] Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. A faster cutting plane method and its implications for combinatorial and convex optimization. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 1049–1065. IEEE, 2015.

[LSZ+23] S Cliff Liu, Zhao Song, Hengjie Zhang, Lichen Zhang, and Tianyi Zhou. Space-efficient interior point method, with applications to linear programming and maximum weight bipartite matching. In *ICALP*, 2023.

[McC05] S Thomas McCormick. Submodular function minimization. *Discrete Optimization*, 12:321–391, 2005.

[Meg83] Nimrod Megiddo. Towards a genuinely polynomial algorithm for linear programming. *SIAM Journal on Computing*, 12(2):347–353, 1983.

[New65] Donald J Newman. Location of the maximum on unimodal surfaces. *Journal of the ACM (JACM)*, 12(3):395–398, 1965.

[NN89] I.U.E. Nesterov and A.S. Nemirovski. *Self-concordant Functions and Polynomial-time Methods in Convex Programming*. USSR Academy of Sciences, Central Economic & Mathematic Institute, 1989.

[NN94] Yurii Nesterov and Arkadi Nemirovski. *Interior-point polynomial algorithms in convex programming*, volume 13. Siam, 1994.

[NS16] Arnold Neumaier and Damien Stehlé. Faster lll-type reduction of lattice bases. In *Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 373–380, 2016.

[Orl09] James B Orlin. A faster strongly polynomial time algorithm for submodular function minimization. *Mathematical Programming*, 118(2):237–251, 2009.

[OV20] Neil Olver and László A Végh. A simpler and faster strongly polynomial algorithm for generalized flow maximization. *Journal of the ACM (JACM)*, 67(2):1–26, 2020.

[Ren88] James Renegar. A polynomial-time algorithm, based on newton's method, for linear programming. *Math. Program.*, 1988.

[Rot16] Thomas Rothvoss. Integer optimization and lattices. *University of Washington, Spring*, 2016.

[Sch87] C.P. Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theor. Comput. Sci.*, 1987.

[Sch00] Alexander Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *Journal of Combinatorial Theory, Series B*, 80(2):346–355, 2000.

[Sho77] Naum Shor. Cut-off method with space extension in convex programming problems. *Cybernetics and systems analysis*, 1977.

[Sma98] Steve Smale. Mathematical problems for the next century. *The mathematical intelligencer*, 20(2):7–15, 1998.

[SWZ17] Zhao Song, David P Woodruff, and Peilin Zhong. Low rank approximation with entrywise l1-norm error. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 688–701, 2017.

[SWZ19] Zhao Song, David P Woodruff, and Peilin Zhong. Relative error tensor low rank approximation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2772–2789. SIAM, 2019.

[SXZ22] Zhao Song, Zhaozhuo Xu, and Lichen Zhang. Speeding up sparsification using inner product search data structures. *arXiv preprint arXiv:2204.03209*, 2022.

[SYYZ22] Zhao Song, Xin Yang, Yuanyuan Yang, and Tianyi Zhou. Faster algorithm for structured john ellipsoid computation. *arXiv preprint arXiv:2211.14407*, 2022.

[SYZ23] Zhao Song, Mingquan Ye, and Lichen Zhang. Streaming semidefinite programs: $O(\sqrt{n})$ passes, small space and fast runtime. *arXiv preprint arXiv:2309.05135*, 2023.

[Tar86] Éva Tardos. A strongly polynomial algorithm to solve combinatorial linear programs. *Operations Research*, 34(2):250–256, 1986.

[TKE88] S. P. Tarasov, L. G. Khachiyan, and I. I. Èrlikh. The method of inscribed ellipsoids. *Dokl. Akad. Nauk SSSR*, 1988.

[VA93] Pravin M. Vaidya and David S. Atkinson. *A Technique for Bounding the Number of Iterations in Path Following Algorithms*, pages 462–489. World Scientific Publishing Company, 1993.

[Vai89] P.M. Vaidya. A new algorithm for minimizing convex functions over convex sets. In *30th Annual Symposium on Foundations of Computer Science*, pages 338–343, 1989.

[Vég14] László A Végh. A strongly polynomial algorithm for generalized flow maximization. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 644–653, 2014.

[VWXXZ24] Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. New bounds for matrix multiplication: from alpha to omega. In *SODA*, 2024.

[VY96] Stephen A Vavasis and Yinyu Ye. A primal-dual interior point method whose running time depends only on the constraint matrix. *Mathematical Programming*, 74(1):79–120, 1996.

[Vyg03] Jens Vygen. A note on schrijver's submodular function minimization algorithm. *Journal of Combinatorial Theory, Series B*, 88(2):399–402, 2003.

[YN76] David Yudin and Arkadii Nemirovski. Evaluation of the information complexity of mathematical programming problems. *Ekonomika i Matematicheskie Metody*, 1976.