Some Problems in Conic, Nonsmooth, and Online Optimization

Swati Padmanabhan

A dissertation

submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

University of Washington
2023

Reading Committee:

Yin Tat Lee, Chair

Dmitriy Drusvyatskiy
Maryam Fazel

Program Authorized to Offer Degree:
Electrical and Computer Engineering

University of Washington

Abstract

Some Problems in Conic, Nonsmooth, and Online Optimization

Swati Padmanabhan

Chair of the Supervisory Committee:

Yin Tat Lee

Paul G. Allen School of Computer Science and Engineering

In this thesis, we study algorithms with provable guarantees for structured optimization problems arising in machine learning and theoretical computer science.

One of the threads of this thesis is semidefinite programs (SDPs), a problem class with a variety of uses in engineering, computational mathematics, and computer science. Concretely, we study approximately solving the MaxCUT SDP. Aside from its significance as the SDP relaxation of an NP-hard problem, it has found use in matrix completion algorithms. Our algorithm for this problem combines ideas from the multiplicative weights framework and variance-reduced estimators. We adopt this idea of robust updates to give a faster high-accuracy algorithm to solve general SDPs via interior-point methods. Conceptually, our result for this general problem resolves the prior paradox of cutting-plane methods being faster at solving SDPs than interior-point methods, despite the former tracking far less structural information about the iterates.

A common structural assumption on real-world datasets is that of sparsity or low rank. This structure is mathematically captured by convex non-smooth functions, thus making convex non-smooth optimization a cornerstone of signal processing and machine learning (e.g., in compressed sensing and low-rank matrix problems). Non-smooth optimization has risen in prominence on the non-convex front as well in the context of deep learning (e.g., in deep neural networks). In this thesis, we focus on two problems under the umbrella of nonsmooth optimization: In the

convex setting, we study minimizing finite sum problems with each function depending only on a subset of the coordinates of the problem variable, and our proposed scheme develops a generalized cutting-plane framework; in the nonconvex setting, we focus on the problem of finding a Goldstein stationary point, and our solution combines randomization with geometric insights into prior work along with a novel application of cutting-plane methods.

Optimization techniques have been used with great success to further progress in foundational questions in applied linear algebra. We explore this interplay in two questions. We first study least-squares regression with non-negative data and problem variables. This structure appears in several real-world datasets (e.g., in astronomy, text mining, and image processing) but has generally not been leveraged by standard least-squares algorithms (including ones in commercial software); in contrast, we utilize this structure, yielding improvements in the runtime (in both theory and experiments). We further study the computation of ℓ_p Lewis weights. These are generalized importance scores of a given matrix used to sample a small number of key rows in tall data matrices and thus a crucial primitive in modern machine learning pipelines. We offer a fresh perspective to this problem, departing from the prior approach of using a fixed-point iteration.

We also apply optimization theory in the context of market economics. Specifically, we study budget-constrained online advertising, an important problem for many technological companies, and develop an optimal-regret bidding algorithm under the "return-on-spend" constraint. Our main insight combines a novel white-box analysis of first-order methods for packing LPs with problem-specific structure.

Acknowledgements

This thesis would not exist without the guidance and support of many amazing people.

First, I would like to thank my advisor, Yin Tat Lee. Yin Tat introduced me to theoretical computer science and, over the course of six years, mentored me with tremendous kindness, patience, and care. I will always cherish our numerous discussions on whiteboards and LyX. Yin Tat is not just a great research collaborator, but also a generous and caring advisor, who would regularly check in especially during the difficult pandemic times. It has been a massive privilege to be advised by him.

Next, I would like to thank my thesis committee: Dmitriy Drusvyatskiy, Maryam Fazel, and Mehran Mesbahi. I am especially grateful to Dima for his wonderful class on convex analysis, for introducing me to non-convex optimization, and for his extensive support during my job search; to Maryam for her convex optimization class; and to Mehran for his fun network analysis class.

I am deeply grateful to Jelena Diakonikolas for taking a chance on me by offering me a lovely summer research visit with her and for her supportive mentorship ever since. I am also extremely grateful to Aranyak Mehta, Di Wang, Zhe Feng, Sagi Perel, Richard Zhang, and David Woodruff for fun internships at Google, where I got the chance to work on theory for real-world problems. I am especially grateful to Di for numerous discussions and his constant encouragement.

One of the greatest joys of my PhD was the chance to learn from brilliant collaborators: I am indebted to each of my amazing co-authors — Yin Tat Lee, Haotian Jiang, Tarun Kathuria, Zhao Song, Kevin Tian, Arun Jambulapati, Jerry li, Maryam Fazel, Aaron Sidford, Jelena Diakonikolas, Chenghui Li, Chaobing Song, Dmitriy Drusvyatskiy, Damek Davis, Guanghao Ye, Di Wang, Zhe Feng, Sally Dong, Mehrdad Ghadiri, Richard Zhang, and David Woodruff. I would particularly like to acknowledge Kevin, Arun, and Guanghao, with whom I have spent the most time during my PhD thinking about research — thank you for making research discussions so joyful!

I owe much of my learning at UW to the wonderful CS Theory group, both faculty — Yin Tat, Shayan, Anna, Thomas, James, Paul, Rachel, Stefano, Kevin, Jamie, and Anup — and students — Becca, Harish, Kira, Siva, Makrand, Jeffrey, Cyrus, Xin, Vincent, Alireza, Robbie, Sami, Jennifer, Farzam, Dorna, Ewin, Kuikui, Guanghao, Haotian, Ruoqi, Sally, Xinzhi, Daogao, Victor, Ashrujit, Siddharth, Nathan, Oscar, Ansh, Michael, and Artin. I am also grateful to friends and mentors from the broader optimization community — Reza Eghbali, Courtney Paquette, Kevin Tian, Krishna Pillutla, John Thickstun, Ankit Pensia, Elizabeth Yang, Deeksha Adil, Taisuke Yasuda, Madhur Tulsiani, Lijun Ding, Georgina Hall, Rachitesh Kumar, Balu Sivan, and Stephen Mussmann — for their generous research advice and many technical discussions. I'd also like to express my gratitude to Elise Dorough and to the UW CSE community for help navigating all administrative procedures.

I thank Krishna, Robbie, John, Srini, Chandra, Romain, and Erin for most of my hallway conversations in Allen through much of my PhD. I would like to especially thank Ruta, Jennifer, Bittoo, Elizabeth, Ewin, Becca, and Sparrow for their warm friendship and constant support through the years.

Finally, I thank my parents for a happy, loving, and carefree childhood that continues to this day.

Contents

Chapter 1

Introduction

Mathematical optimization is a key tool in engineering, operations research, computational mathematics, and data science. Due to its ubiquity and centrality, it is now regarded as a mature field of study with algorithms and associated theoretical guarantees well-understood in classical regimes. However, the aforementioned fields are witnessing the rapid emergence of applications for which the theoretical guarantees of existing optimization algorithms do not suffice. This is particularly acute when pursuing the goal of scalability for problems with structures increasingly prevalent in practice. This void necessitates novel algorithmic solutions beyond classical frameworks. In this thesis, we address some of these needs by focusing on the theoretical aspects of designing fast optimization algorithms for structured problems arising in modern data science applications, as we highlight next.

1.1 Semidefinite Programs

Semidefinite programs (SDPs) — a class of convex programs maximizing a linear function over the intersection of a finite number of halfspaces and the positive semidefinite cone — constitute a core optimization primitive generalizing linear and second-order cone programs.

The vastness of this problem class makes SDPs ubiquitous in approximation algorithms [GM12] (e.g., in obtaining the best approximation ratios for multiple NP-Hard problems such as Max-CUT [GW95], coloring 3-colorable graphs [KMS94], and sparsest cut [ARV09]), quantum complexity theory [JJUW11], robust statistics [CG18, CDG19, CDGW19], algorithmic discrepancy and rounding [BDG16, BG17, Ban19]), control theory [BEGFB94], polynomial optimization [Par00, Hal18], and machine learning (e.g., kernel learning [LCB+04], variational inference [Bac22], and robustness certification for neural networks [RSL18]). We study fast algorithms for approximately solving SDPs.

SDPs with Diagonal Constraints. The first problem we study is the MaxCUT SDP, one of the simplest and most well-studied SDPs. Given an $n \times n$ cost matrix C, this problem seeks a positive definite matrix $X \succeq 0$ that satisfies $X_{ii} \leq 1$ for all $i \in [n]$.

The MaxCUT SDP [GW95] arises naturally as the SDP relaxation of the MaxCUT problem and has seen widespread utility in circuit design [CKC83], statistical physics [BGJR88], phase recovery [WdM15], rank minimization [Jag11, SS05, FHB04], community detection [ABH15, GV16, MS16a], the group synchronization problem [SS11, BCSZ14], and semi-supervised learning [WJC13].

Our goal is to solve this problem to ε-accuracy. Specifically, given the $n \times n$ cost matrix C, we seek a matrix $X \succeq 0$ with $X_{ii} \leq 1$ for all $i \in [n]$, such that $C \bullet X \geq \text{OPT} - \varepsilon \sum_{i,j} |C_{ij}|$, where OPT is the optimal value. We focus on first-order methods with a linear dependence on the number of non-zero entries of C) and operating in the regime of moderate ε. Prior work on this problem in this algorithm class includes saddle-point optimization [GH16a], mirror-prox combined with low-rank sketching [BBN13, CDST19a], low-rank first-order methods [YTF+19a], and algorithms for "covering SDPs" [JLL+20a].

Our Contribution. In joint work [LP20] with Yin Tat Lee, we obtain for this problem a first-order algorithm at a cost of $\widetilde{O}\left(\frac{m}{\varepsilon^{3.5}}\right)$, where m is the number of non-zeroes in the cost matrix.

This improves upon the previously fastest solver of Arora-Kale[AK07]. We build upon Arora-Kale's multiplicative weights framework with the simple modification of making frequent low-accuracy approximations (to reduce the cost) and infrequent exact computations (to "reset" the error resulting from approximation). Our solver has found use in coding theory [JST21] and adversarial robustness [AJRV20]. We detail our result in Chapter 2.

General SDPs. We adopt our above idea of robust updates in a general-purpose SDP solver as well. We now, however, shift gears from the regime of moderate ε to low ε, for which, broadly, there exist two classes of iterative algorithms: cutting-plane methods and interior-point methods.

Cutting-plane methods iteratively search for a small ball containing the function optimum we seek. They start with a large convex set guaranteed to contain the optimum and, in each iteration, query at a point in this set a separation oracle that tells them which half of the current search space the optimum lies in. Using this information, they shrink this search set by a constant factor in each iteration and, in a finite number of iterations, zero in on the optimum. Since Khachiyan proved [Kha80] that the ellipsoid method solves linear programs in polynomial time, cutting plane methods have become an active area of research in both discrete and continuous optimization [GLS81a, GV02]

In contrast, interior point methods turn the original constrained optimization problem into a sequence of unconstrained optimization problems parametrized by a scaling factor that ascribes relative weight to optimizing the objective versus enforcing feasibility. The solutions to these successive problems form a well-defined central path through the original feasible set and converge to the approximate optimum in a finite number of steps. Since Karmarkar's proof [Kar84] that interior point methods can solve linear programs in polynomial time, these methods have seen tremendous progress [NN92, NN94, Ans00] and have been central to several recent breakthroughs in algorithms for combinatorial optimization [Lee16, LS14, VDBLL+21, vdB21, KLS22].

Thus, the two algorithms greatly differ in terms of the amount of information known about the optimum: For cutting plane methods, all one knows is that the solution lies inside an intersection of all the halfspaces returned by the oracle so far; for interior point methods, we have a system of equations (given by, for example, the KKT conditions) that precisely describe the optimum at each step. Since cutting plane methods use less structural information than interior point methods, they are slower at solving almost all problems where interior point methods are known to apply. However, prior to our work, the fastest cutting-plane method[LSW15] was faster than interior-point methods for SDPs.

Our Contribution. In joint work [JKL+20] with Haotian Jiang, Tarun Kathuria, Yin Tat Lee, and Zhao Song, our contribution is to resolve this paradox via a new interior point method that solves an $n \times n$ variable SDP with m constraints at a cost of $\widetilde{O}(\sqrt{n} \cdot (mn^2 + m^\omega + n^\omega))$.

We obtain our result by a simple, intuitive modification to the IPM framework of Nesterov and Nemirovskii [NN92] Instead of maintaining the true slack matrix, we maintain a spectral approximation to it that admits low-rank updates and introduce a novel potential function to show the correctness of using this approximate matrix. Our technique has spurred further advances in fast algorithms for specialized settings like tall dense SDPs [HJST21] and robust correlation clustering [CPRT22]. We detail our result in Chapter 3.

1.2 Nonsmooth Optimization

In the previous section, we described our modification of the classical interior point method for solving SDPs. In this section, we turn our attention to another second order method, the cutting plane method, and modify it to obtain improved rates in nonsmooth optimization.

This problem class spans a multitude of practical applications: In convex settings, nonsmooth functions capture properties like sparsity and low rank, both pervasive structural assumptions imposed on real-world datasets [UT19] and the backbone of compressed sensing [CRT06] and low-rank matrix problems [RFP10]. Nonsmooth optimization has risen in prominence on the nonconvex front as well in the context of deep learning, for example to train deep neural networks with non-linear activations. We study two problems in nonsmooth optimization, one convex, and one nonconvex.

Nonsmooth Convex Optimization. We study finite-sum minimization of $\min_{\theta \in \mathbb{R}^d} \sum_{i=1}^{n} f_i(\theta)$ of convex, non-smooth functions, where each function is supported on a subset of coordinates of the problem variable. This problem is notably exemplified by decomposable submodular function minimization (SFM), with applications in, e.g., determinantal point processes [KT10] and computer vision [KLT09, VKR09, FJPZ13]. Our goal here is low subgradient oracle complexity.

The general "finite sum minimization" problem has been extensively studied in the setting with smooth f_i's and has spurred the development of well-known variants of stochastic gradient methods [RM51, BC03, Zha04, Bot12] such as [RSB12, SSZ13b, JZ13a, MZJ13, DBLJ14b, Mai15, AZY16, HL16a, SLRB17a], which in turn powered tremendous empirical success in machine learning through widely used software packages such as `libSVM` [CL11a]; almost universally, these algorithms leverage the "sum structure" of the objective by sampling, in each iteration, one f_i with which to make progress.

However, these prior algorithms chose such an f_i *arbitrarily*, which fails to make sufficient progress in our setting. In particular, all variants of gradient descent for this problem have a polynomial dependence on the condition number. Existing cutting plane methods, on the other hand, trade off dependence on condition number for a suboptimal dependence on the dimension. Additionally, the work on non-smooth ERM crucially requires the objective function to be a sum of a smooth ERM part and a non-smooth regularizer. These results do not apply to the many important problems for which the objective function cannot be split in this way.

> **Our Contribution.** In joint work [DJL+22] with Sally Dong, Haotian Jiang, Yin Tat Lee, and Guanghao Ye, we provide an algorithm that solves this problem in a nearly-linear (in total effective dimension) number of queries to the subgradient oracle.

Our subgradient oracle complexity obtained here is nearly optimal. We obtain our result by adaptively choosing the f_i to make progress on in any given iteration; we operate in a conceptually novel cutting-plane framework. This work also implies state-of-the-art theoretical results for decomposable SFM. The detailed presentation is in Chapter 4.

Nonsmooth Nonconvex Optimization. Relative to its convex counterpart, e.g., described above, scant research has investigated non-convexity in the non-smooth setting. However, with the advent of deep learning, the question of studying convergence guarantees of algorithms for optimizing non-smooth non-convex functions (of which modern neural networks are a classic example) has come to be one of paramount importance. While there has been a growing body of research on this topic [BHS05, Kiw07, MMM18, DDKL20, BP21], the question of *non-asymptotic* convergence guarantees has seen relatively scant results.

To make progress towards this goal, the first difficulty one encounters with the class of non-smooth non-convex problems is in defining "convergence". For example, it is generally impossible to obtain local minima or approximate-stationary points [NY83b, ZLSJ20], and it is even impossible to get close to such points within any finite time independent of the dimension [KS21]. Thus, in general, these problems are known to be impossible to solve at a dimension-free rate without further assumptions such as convexity or smoothness.

[ZLSJ20] showed that what is in fact tractable notion of convergence for this problem class is (δ, ϵ)-stationarity, as pioneered by [Gol77]. Put simply, a point is (δ, ϵ)-stationary if within a δ-ball around it one can find a convex combination of subgradients that have a total norm of at most ϵ. We remark that this is weaker than all the aforementioned notions of stationarity [ZLSJ20, KS21]. [ZLSJ20] presented an algorithm that, for an L-Lipschitz function, can achieve (δ, ϵ)-stationarity in $O\left(\frac{\Delta L^2}{\delta \epsilon^3}\right)$ calls to a specific type of first-order oracle. While a remarkable breakthrough in this field, the oracle used in [ZLSJ20] was quite non-standard.

Our Contribution. In joint work [DDL+22] with Damek Davis, Dmitriy Drusvyatskiy, Yin Tat Lee, and Guanghao Ye, we strengthen this result by the use of a standard first-order oracle. We also improve the complexity guarantee for low-dimensional settings.

We obtain our first result by applying simple ideas from *randomization to our geometric insights* into the algorithm of [ZLSJ20]. To achieve our second result, we use a novel cutting-plane technique. We describe this work in Chapter 5.

1.3 Linear Algebraic Problems

Optimization techniques have been used with great success to further progress in the most foundational questions in applied linear algebra. We study two questions at the interplay of these topics.

Non-Negative Least Squares Regression. Our first question under this umbrella concerns the computational complexity of solving least squares regression when the problem data and variables are both element-wise non-negative. Nonnegative least squares (NNLS) problems, defined as $\min_{\mathbf{x} \geq \mathbf{0}} \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$, where $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^m$, have been studied for decades in optimization and statistical learning [LH95, BJ97, KSD13], with various off-the-shelf solvers available in many standard programming languages. Within machine learning, NNLS problems arise whenever having negative labels is not meaningful, for example, when representing prices, age, pixel intensities, chemical concentrations, or frequency counts. NNLS is also widely used as a subroutine in nonnegative matrix factorization [CZPA09, Gil14, KSK13] to extract sparse features in applications like clustering, collaborative filtering, and community detection.

From an algorithmic standpoint, the nonnegativity constraint in NNLS problems is typically viewed as an obstacle: most NNLS algorithms perform additional work to handle it, and the problem is considered harder than unconstrained least squares. However, in many important applications of NNLS, such as text mining [BBL+07], functional MRI [AR04, JHD18], EEG data analysis [MMSBVS08], pulse oximetry [JP87, WPTP88], statistical procedures in observational astronomy [IFAB90], and those traditionally addressed using nonnegative matrix factorization [CZPA09], *the data is also nonnegative*. We study NNLS with nonnegative data and argue that in this setting it possible to obtain *stronger* guarantees than for traditional least squares.

Our Contribution. In joint work [DLPS22] with Jelena Diakonikolas, Chenghui Li, and Chaobing Song, we introduce a width-independent, accelerated algorithm for non-negative least squares on non-negative data. We further improve this to a linear convergence rate and supplement all our theoretical results with experiments on real datasets.

In our result, "width" is the maximum ratio between the non-negative elements of the matrix; *width-independence* means the convergence depends at most poly-logarithmically on the width, a feature highly desirable in many problems such as those in the literature on packing/covering LPs [Wan17]. Without additional structure, width-*dependent* algorithms are often not polynomial-time. We achieve our result by leveraging structural properties specific to this problem, a novel acceleration technique, and by incorporating a restart strategy. We present this result in Chapter 6.

Computing Lewis Weights to High Precision. While the previous problem was concerned with ℓ_2 regression, we now shift gears to ℓ_p *regression*. This problem has been a fixture of machine learning and theoretical computer science, capturing fundamental problems like linear programming ($p = 1$), least squares regression ($p = 2$), and max-flow ($p = \infty$). As a first step, we study the computation of ℓ_p-Lewis weights. ℓ_p-Lewis weights are "generalized importance scores" one can compute for each row of a matrix, generalizing leverage scores. An alternate geometric interpretation of Lewis weights is as the solution to the problem of computing a minimum-volume ℓ_p ellipsoid.

These two meanings one can assign Lewis weights lends them ubiquity in a variety of fields. Their property of assigning importance to each row of a matrix makes it possible to use them to sample a small number of their key rows of an input matrix in a way that the ℓ_p norms of the product of the matrix with vectors are preserved. This has wide-ranging applications, e.g., in row sampling algorithms for data pre-processing [DMM06, DMIMW12, LMP13, CLM+15a, CP15], for computing dimension-free strong coresets for k-median and subspace approximation [SW18], for fast tensor factorization in the streaming model [CCDS20], and for ℓ_1 regression, a popular model in machine learning used to capture robustness to outliers, in: [DLS18] for stochastic gradient descent pre-conditioning, [LWYZ20] for quantile regression, and [BDM+20] to provide algorithms for linear algebraic problems in the sliding window model. They are also used in statistics (e.g., in D-optimal design) and optimization (e.g., in the construction of nearly-optimal self-concordance barriers for polytopes [LS13]).

Computing ℓ_p-Lewis weights, thus, is an essential algorithmic primitive, and faster ℓ_p-Lewis weights computation can have wider ramifications for exciting runtime improvements in the aforementioned areas. The previous known high-accuracy ℓ_p-Lewis weights solver [CP15] was limited to the range $p \in (0, 4)$, and there was no such result for the range $p \geq 4$.

Our Contribution. In joint work [FLPS22] with Maryam Fazel, Yin Tat Lee, and Aaron Sidford, we give the first high-precision ℓ_p-Lewis weights solver for $p \geq 4$.

Our work thus completes the picture on efficient ℓ_p-Lewis weight computation for all $p > 0$. The prior approach was limited by its use of fixed-point iterations, which, by design, do not converge for $p \geq 4$. Departing from this technique, our new insight for $p \geq 4$ is to instead use the structural properties of the convex program describing ℓ_p-Lewis weights. We present this result in Chapter 7.

1.4 Online Optimization

Outside of the theme of *scalability* as exemplified in the previous sections, we also study the application of optimization theory to the design of *low-regret algorithms in market economics*. This

was work done as an intern at Google Research (Market Algorithms). Specifically, we study budget-constrained online advertising, a problem of paramount importance to modern technological giants. The broad goal of our problem is to determine bids for incoming queries to maximize advertisers' targets subject to their specified constraints. We focus on a single value-maximizing advertiser under an increasingly popular constraint: Return-on-Spend (RoS). We quantify efficiency in terms of regret relative to the optimal algorithm, which knows all queries a priori.

Our Contribution. In joint work [FPW23] with Di Wang and Zhe Feng, we contribute a simple online algorithm that achieves near-optimal regret in expectation while always respecting the specified RoS constraint when the input sequence of queries are i.i.d. samples from some distribution. Integrating this with the previous work of Balseiro, Lu, and Mirrokni, we also achieve near-optimal regret while respecting both RoS and fixed budget constraints.

Our algorithm follows the primal-dual framework and uses online mirror descent (OMD) for the dual updates. However, we need a non-canonical setup of OMD, and therefore the classic low-regret guarantee of OMD, which is for the adversarial setting in online learning, no longer holds. This necessitates a novel white-box analysis of first-order methods seen in the literature on packing LPs [Wan17] in conjunction with problem-specific structure. This result is presented in Chapter 8.

1.5 Organization of the Thesis

The work presented in this thesis is the result of several research collaborations. Prior publications of the work in this thesis are listed below. All publications follow the alphabetical ordering of authors, per the convention in theoretical computer science.

- Chapter 2: An $\widetilde{O}(m/\varepsilon^{3.5})$ Algorithm for Semidefinite Programs with Diagonal Constraints; joint work with Yin Tat Lee; published in Conference on Learning Theory (COLT), 2020
- Chapter 3: A Faster Interior Point Method for Semidefinite Programming; joint work with Haotian Jiang, Tarun Kathuria, Yin Tat Lee, and Zhao Song; published in Foundations of Computer Science (FOCS) 2020
- Chapter 4 Decomposable Non-Smooth Convex Optimization with Nearly-Linear Gradient Oracle Complexity; joint work with Sally Dong, Haotian Jiang, Yin Tat Lee, and Guanghao Ye; published in Advances in Neural Information Processing Systems (NeurIPS) 2022
- Chapter 5 A gradient sampling method with complexity guarantees for Lipschitz functions in high and low dimensions; joint work with Damek Davis, Dmitriy Drusvyatskiy, Yin Tat Lee, and Guanghao Ye; published in Advances in Neural Information Processing Systems (NeurIPS) 2022
- Chapter 6: A Fast Scale-Invariant Algorithm for Non-negative Least Squares with Non-negative Data; joint work with Jelena Diakonikolas, Chenghui Li, and Chaobing Song; published in Advances in Neural Information Processing Systems (NeurIPS) 2022
- Chapter 7 Computing Lewis Weights to High Precision; joint work with Maryam Fazel, Yin Tat Lee, and Aaron Sidford; published in Symposium on Discrete Algorithms (SODA) 2022
- Chapter 8: Online Bidding Algorithms for Return-on-Spend Constrained Advertisers; joint work with Di Wang and Zhe Feng; published in TheWebConf (called WWW until recently) 2023

Chapter 2

An $\widetilde{O}(m/\varepsilon^{3.5})$ Cost Algorithm for Semidefinite Programs with Diagonal Constraints

In this chapter, we study semidefinite programs with diagonal constraints. This problem class appears in combinatorial optimization and has a wide range of engineering applications such as in circuit design, channel assignment in wireless networks, phase recovery, covariance matrix estimation, and low-order controller design. We give a first-order algorithm to solve this problem to ε-accuracy, with a run time of $\widetilde{O}(m/\varepsilon^{3.5})$, where m is the number of non-zero entries in the cost matrix. We improve upon the previous best run time of $\widetilde{O}(m/\varepsilon^{4.5})$ by Arora and Kale. As a corollary of our result, given an instance of the Max-Cut problem with n vertices and $m \gg n$ edges, our algorithm when applied to the standard SDP relaxation of Max-Cut returns a $(1-\varepsilon)-\alpha_{GW}$ cut in time $\widetilde{O}(m/\varepsilon^{3.5})$, where $\alpha_{GW} = 0.878567$ is the Goemans-Williamson approximation ratio. We obtain this run time via the stochastic variance reduction framework applied to the Arora-Kale algorithm, by constructing a constant-accuracy estimator to maintain the primal iterates.

2.1 Introduction

Consider the SDP maximizing $C \bullet X \stackrel{\text{def}}{=} \text{Tr}(CX)$ over the set of $n \times n$ positive semidefinite matrices with every diagonal entry bounded by a constant:

$$\text{maximize } C \bullet X \text{ subject to } X \succeq 0, X_{ii} \leq 1 \text{ for all } i \in [n]. \tag{2.1.1}$$

We seek a matrix $\widetilde{X}^* \succeq 0$ with $\widetilde{X}^*_{ii} \leq 1$ for all indices $i \in [n]$, such that $\widetilde{X}^*$ satisfies $C \bullet \widetilde{X}^* \geq C \bullet X^* - \varepsilon \sum_{i,j} \left|C_{ij}\right|$, where X^* is an optimal solution of (2.1.1). This is *not* an ε-multiplicative guarantee ($C \bullet \widetilde{X}^* \geq C \bullet X^*(1-\varepsilon)$), but a slightly weaker one, since[1] we have $\sum_{i,j} |C_{ij}| \geq C \bullet X^*$. We remark that a multiplicative guarantee is not always easy to provide; indeed, even many classical optimization algorithms provide a guarantee that is only additive in some quantity that bounds from above the difference of the function values between the initial and optimal points. For example, gradient descent on an L-smooth convex function f over a set with diameter D returns, after k iterations, a point x_k such that $f(x_k) - f(x^*) \leq O(LD^2k^{-1})$, where $f(x_0) - f(x^*) \leq O(LD^2)$.

To solve (2.1.1) as per the above accuracy criterion, it suffices to solve (2.1.2):

$$\text{minimize } f(X) \stackrel{\text{def}}{=} -\widehat{C} \bullet X + \sum_{i=1}^{n} (X_{ii} - \rho_i)^+, \text{ subject to } X \succeq 0. \tag{2.1.2}$$

[1] Since $X^* \succeq 0$, we have $|X^*_{ij}| \leq X_{ii}$, which, by the SDP constraint, is upper bounded by 1. Then, applying Hölder's inequality gives $C \bullet X^* \leq \sum_{i,j} |C_{ij}| \max_{i,j} |X^*_{ij}| \leq \sum_{i,j} |C_{ij}|$.

This problem is derived from (2.1.1) by promoting the diagonal constraints to the objective and appropriately scaling C to $\widehat{C} \stackrel{\text{def}}{=} \mathbf{diag}(1/\sqrt{\rho})C\mathbf{diag}(1/\sqrt{\rho})$, where $\rho \in \mathbb{R}^n$ such that $\rho_i = \sum_{j\in[n]} \left|C_{ij}\right|$. By rescaling the entries of the matrix C as $C_{ij} = nC_{ij}/\sum_{i,j}|C_{ij}|$, we assume $\sum_{i\in[n]} \rho_i = n$. Lemma 2.1.4 gives a solution of (2.1.1) from a solution of (2.1.2).

For (2.1.1), [AK07] have the previous best run time linear in $m \stackrel{\text{def}}{=} \text{nnz}(C)$, the size of the input. Though there exist algorithms with better dependence on ε, their dependence on n is superlinear, as we describe in Section 2.1.1. In this chapter, *we operate in the regime of moderate ε and large n, focusing on first-order methods with linear dependence on m.*

To solve (2.1.1), [AK07] use the algorithm "matrix multiplicative weights (MMW) update", which, in this setting, can be interpreted as mirror descent in the nuclear norm[2], using the negative entropy function, $\Phi(X) = X \bullet \log X$, over the scaled simplex, $\mathcal{D} = \{X : X \succeq 0, \operatorname{Tr} X = n\}$, as the mirror map. Their iterates at iteration t are given by

$$X^{(t)} = n\frac{\exp\left(Y^{(t)}\right)}{\operatorname{Tr}\exp(Y^{(t)})}, \text{ where } Y^{(t)} = \sum_{s=1}^{t-1} -\eta\nabla f(X^{(s)}), \tag{2.1.3}$$

with step size $\eta = O(\varepsilon)$ and gradient $\nabla f(M) = \mathbf{diag}(\mathbf{1}_{M\geq\rho}) - \widehat{C}$. Computing this gradient entails only comparing the diagonal entries of the current iterate with a fixed vector. Therefore, in each iteration, the naïve computational cost of this method is dominated by $\Omega(n^\omega)$ for the matrix exponentiation [PC99], prohibitively expensive for a large problem dimension. [AK07] circumvent this by *approximating* the diagonal entries of the matrix exponential. Therefore, their overall cost is composed of the following three parts: (1) mirror descent requiring $O(1/\varepsilon^2)$ iterations to converge, (2) degree $O(1/\varepsilon)$ Taylor approximation of the matrix exponential, each matrix-vector product costing $O(m)$, and (3) $O(1/\varepsilon^2)$ random projections [JL84] to estimate the diagonal entries of the matrix exponential; combined, these give a run time of $\widetilde{O}(m/\varepsilon^5)$, which, [AZL17a] observe, can be sped up to $O(m/\varepsilon^{4.5})$ by using Chebyshev (instead of Taylor) approximation of matrix exponentials (see [SV+14]).

Our contribution. In this chapter (published, in joint work [LP20] with Yin Tat Lee, at the Conference on Learning Theory, 2020), we solve (2.1.1) with a run time of $\widetilde{O}(m/\varepsilon^{3.5})$, thus speeding up the previous best run time for this problem. Our result (formally stated in Theorem 2.2.1) is effected by careful technical work that incorporates into the Arora-Kale framework of mirror descent for SDPs, variance-reduced estimators and fast products of matrix exponentials with vectors.

We use the generalized negative entropy, $\Phi(X) = X \bullet \log(X) - \operatorname{Tr} X$, as our mirror map, and our primary high-level idea is the following: *instead of exactly computing the primal iterate in each iteration, we frequently approximate it at a low accuracy (to reduce the cost) and infrequently at a high accuracy (to "reset" the error resulting from approximation).* This idea is inspired by recent variance-reduction methods [SSZ13c, JZ13b, DBLJ14a, HL16b, SLRB17b]. The periodic high-accuracy computations and small bias and variance of estimators in the low-accuracy computations ensure sufficient closeness, in the appropriate norm, of the estimated iterates to the true ones, which, by the convergence guarantee of approximate mirror descent, leads to an ε-optimal solution. Making this variance-reduction work in the MMW setting requires several technical ideas, as follows.

We introduce the technical idea of expanding the domain of our mirror map by a polylogarithmic factor. Due to the expanded domain and our choice of the mirror map, the gradient step of mirror descent falls in the *interior* of this domain. An upshot of these modifications to the domain and mirror

[2]The nuclear norm of a matrix $X \in \mathbb{R}^{m\times n}$ is the sum of its singular values: $\|X\|_{\text{nuc}} \stackrel{\text{def}}{=} \sum_{i=1}^{\min(m,n)} \sigma_i(X)$.

map is that the primal iterate is related to the dual via simply a matrix exponential, with no trace normalization of the form seen in Equation (2.1.3). Thus, the quantity for which we require an estimator is greatly simplified. Drawing on the observation of [AK07] that the gradient uses only the diagonal entries of the primal iterate, we build an estimator, with a small bias and variance, for the change in diagonal entries of the (dual) matrix exponential. We also prove the strong convexity parameter of our mirror map on the expanded domain by confecting classical results from convex analysis in a novel way. Due to the ubiquity of the MMW framework in optimization, efficient algorithms for SDPs, balanced separators, Ramanujan sparsifiers, packing/covering, and machine learning, we anticipate that our technical contributions will be useful for problems that hinge on the MMW foundation.

Applications. When C is a graph Laplacian, (2.1.1) is the SDP relaxation of the Max-Cut problem, as was given by Goemans and Williamson [GW95]. An NP-complete problem [Kar72], Max-Cut has seen widespread utility in circuit design [CKC83], statistical physics [BGJR88], semi-supervised learning [WJC13], and phase recovery [WdM15]. Another instance of (2.1.1) is max-norm regularization [Jag11], a convex surrogate for rank minimization [SS05] enforcing simplicity in modeling observations [FHB04]. SDPs of the form of (2.1.1) have also found applications in community detection [ABH15, GV16, MS16a] and as relaxations to the maximum-likelihood estimator in the group synchronization problem [SS11, BCSZ14]. We also highlight an interesting recent application of (2.1.1) in the area of adversarial robustness [AJRV20].

Chapter outline. After surveying related work (Section 2.1.1), we lay out required background (Section 2.1.2). Our algorithm is in Section 2.2, our estimator and proof sketch of the main result in Section 2.2.1, and complete mathematical details in the Appendices.

2.1.1 Related work

We describe in this section previous work on (2.1.1) using first-order methods, other than that of [AK07]. Of note is that most papers below solve problems more general than (2.1.1), and the run times we mention occur when specialized to (2.1.1). For the sake of completeness of exposition, we explain some details of these results in Section A.1. Though our focus is restricted to *first-order methods*, for completeness, we mention that the interior-point method by [HRVW96] applied to (2.1.1) costs $O(n^{\omega+1/2})$, and the current fastest cutting plane method [LSW15] costs $\widetilde{O}(n(n^{\omega}+m))$.

Saddle-point formulation. Since any SDP can be instantiated as an online convex optimization problem we apply to our setting some notable results from online convex optimization. To do so, we first reduce (2.1.1) to a feasibility problem following the approach of [AHK05].

The first step in this reduction requires obtaining a range of values of the optimum solution, which is what we derive in this paragraph. Recall our assumption that $\sum_{i,j}|C_{ij}| = n$. The facts $X^* \succeq 0$ and $X^*_{ii} \leq 1$ for $i \in [n]$ together imply that $|X^*_{ij}|^2 \leq X^*_{ii}X^*_{jj} \leq 1$, which in turn bounds the optimum from above as $OPT = \sum_{i,j} C_{ij}X^*_{ij} \leq \sum_{i,j}|C_{ij}||X^*_{ij}| \leq n$. We can also bound the optimum from below by choosing X to be the zero matrix, thus bounding OPT by some variable λ that satisfies the inclusion $\lambda \in [0, n]$.

Next, we use this range to construct appropriate feasibility problems, which when solved, reduce to solutions to (2.1.1). Let $A_0 = \frac{1}{\lambda}C$, $b_0 = 1$, $A_i = -e_ie_i^\top$, and $b_i = -1$ for $i \in [n]$. Therefore, solving (2.1.1) requires, for each guess of λ (obtained via a binary search over its range), solving the feasibility problem:

$$\text{Find } Z \in \mathbb{S}^n_{\geq 0} \text{ subject to } A_i \bullet Z - b_i \geq 0, \text{ for all } i \in \{0, n\}, \operatorname{Tr} Z \leq n. \tag{2.1.4}$$

To solve (2.1.4), we leverage the technique of [GH16a], in solving the saddle point problem

$$\max_{X\in\mathbb{S}^n_{\succeq 0},\operatorname{Tr} X=1}\ \min_{p\in\mathbb{R}^m_{\geq 0},\|p\|_1=1}\ \sum_{i=1}^{m} p_i(A_i \bullet X - b_i). \tag{2.1.5}$$

Notice that the inner (minimization) problem returns the j corresponding to the smallest value of $A_j \bullet X - b_j$, since p is a vector in the simplex, and we can choose it to be the vector of all zeroes with one at the j-th index. Then, if the optimum of (2.1.5) is non-negative, solving (2.1.5) up to an additive accuracy of ε is equivalent to finding a solution in the spectrahedron that satisfies all $A_i \bullet X - b_i \geq 0$ upto an additive error of ε. Solving (2.1.5) using the definitions of A_i, b_i, and C from the problem (2.1.4), we get a solution X that satisfies $X_{ii} \approx 1/n \pm \varepsilon$. However, note that the requirement of (2.1.1) is $X_{ii} \approx 1 \pm \varepsilon$, and therefore the accuracy parameter of (2.1.5) needs to be scaled down to ε/n. This causes the run time of [GH16a] for (2.1.1) to be $\widetilde{O}(m(n/\varepsilon)^{2.5})$.

By the same reasoning, when solving (2.1.1) to ε-multiplicative accuracy, the work of [BBN13], which uses a randomized Mirror-Prox algorithm, incurs a cost of $\widetilde{O}(n^5/\varepsilon^3)$, and Follow the Compressed Leader by [AZL17a] and rank-1 sketch by [CDST19a] incur a cost of $\widetilde{O}(m(n\|C\|_\infty/\varepsilon)^{2.5})$.

Thus, all the above saddle-point algorithms, when specialized to (2.1.1) with our accuracy requirements, have superlinear runtimes. We emphasize that [GH16a], [AZL17a], and [CDST19a] provide algorithms satisfying ε-additive accuracy. When we translate *our* accuracy results to *their* language, the costs are not quite comparable. For instance, [CDST19a], for ε-additive accuracy for (2.1.1), incurs a cost of $\widetilde{O}(m(n\|C\|_\infty/\varepsilon)^{2.5})$. Our algorithm, using this accuracy criterion, incurs a cost of $\widetilde{O}(m(\sum_{i,j}|C_{ij}|/\varepsilon)^{3.5})$. Unless we assume additional structure on the matrix C, the comparison between these two costs is inconclusive.

Low-rank updates. When C is the graph Laplacian in (2.1.1), it is known that there exists an ε-accurate solution of rank as low as $O(1/\varepsilon)$ [RS09, MS16b, MMMO17]. Many researchers capitalize on this fact and perform low-rank updates, which reduces cost per iteration.

For example, [KL96] base their algorithm on the framework of [PST91] in conjunction with the power method to achieve a run time of $\widetilde{O}(mn/\varepsilon^3)$. As another example, [Haz08] incorporates into the Frank-Wolfe algorithm [FW56] fast computation of an approximate minimum eigenvector and provides an $\widetilde{O}(mn^3/\varepsilon^3)$-algorithm. Another noteworthy result [YTF+19a] returns a rank-R approximation to an ε-optimal solution at a cost $\widetilde{O}(Rn/\varepsilon^2 + n/\varepsilon^3)$. Even though, as alluded to earlier, there exists a rank-$O(1/\varepsilon)$ solution to the MaxCut SDP, perturbing such a solution by an appropriately small amount gives an ε-optimal solution that is in fact full rank. Indeed, per Theorem 6.2 of [YTF+19a], for any $r < R$, the iterate $\widehat{X}_t$ returned by their algorithm in iteration t satisfies $\limsup_{t\to\infty} \mathbb{E}_\Omega \operatorname{dist}_*(\widehat{X}_t, \Psi_*) \leq (1 + r/(R-r-1)) \cdot \max_{X\in\Psi_*} \|X - [X]_r\|_*$, where Ω is the randomness in their algorithm, Ψ_* is the solution set, R is the rank of the iterate returned, and $[X]_r$ is an r-truncated singular value decomposition of matrix X. The existence of full-rank matrices in the solution set Ψ^* implies a possibly large bound on the right hand side above, thereby making it inconclusive that [YTF+19a] is faster than our achieved run time. We do conjecture, though, that exploiting the existence of a low-rank solution is a promising approach to speed up our run time even more.

Polynomial mirror map. An attempt we had previously made to improve the run time over that of [AK07] was to use a "polynomial-style" mirror map inspired by that in [AZL17a], in the hopes that we can avoid matrix exponentiation. Specifically, we considered the map $\Phi(X) = \frac{1}{1+1/2p} \operatorname{Tr} X^{1+1/2p}$. One reason this mirror map fails to give us the desired speed-up is that the projection step with this

map is $X = (Y^+)^{2p}$, where Y^+ is the matrix obtained by zeroing out the negative eigenvalues of Y; note that this step is as expensive as matrix exponentiation. We believe that the exploration of alternative mirror maps with cheaper projection steps is an interesting (and fundamental) open problem.

Variance-reduction methods. Our main idea of low-accuracy estimation interspersed with periodic high-accuracy estimates may, at first glance, seem similar to that in variance reduction algorithms such as SVRG [JZ13b]. We wish to highlight one key difference here: in [JZ13b], the objective is a sum of functions ψ_i, each usually representing the loss incurred by the use of a training example and label, and the algorithm employs an unbiased estimator of the gradient. In our case, neither is (2.1.2) a sum of such functions, nor is its gradient $(\mathbf{diag}(\mathbf{1}_{X>=\rho}) - \widehat{C})$ cheap to estimate due to X being a matrix exponential. In view of these obstacles, we use dimension-reduction techniques to maintain an estimator of the slow-changing $\mathbf{diag}(X)$. Since this quantity is not the gradient itself, its estimator need not be unbiased, a fact we use to our benefit in reducing the cost of estimation.

2.1.2 Preliminaries

Notation. We use $\mathbb{R}^n$ to denote the subspace of n-dimensional real vectors, $\mathbf{1}$ for the vector of all ones, and $\mathbf{1}_{\{\mathcal{E}\}}$ for the all-zero vector with one at coordinates where $\mathcal{E}$ is true. We use x^+ to denote the non-smooth function that equals x when $x \geq 0$ and truncated to zero otherwise. Denote by $\mathbb{S}^n$ the subspace of $n \times n$ symmetric matrices and by I_n the $n \times n$ identity matrix. For $u \in \mathbb{R}^n$, $\mathbf{diag}(u)$ is the $n \times n$ diagonal matrix with $\mathbf{diag}(u)_{ii} = u_i$. For $A, B \in \mathbb{S}^n$, the trace inner product is $A \bullet B \stackrel{\text{def}}{=} \text{Tr}(AB) = \sum_{i,j} A_{ij}B_{ij}$. We define $\|\!|A|\!\| = \sum_i |A_{ii}|$. Given a scalar function f and a vector u, we use $f(u)$ to mean that entrywise, and similarly, for a symmetric matrix A with eigendecomposition $A = U\Lambda U^\top$, $f(A) = Uf(\Lambda)U^\top$. Given $A \in \mathbb{R}^{n\times n}$ and $p \in \mathbb{R}^n$, $A \geq p$ means $A_{ii} \geq p_i$ for all $i \in [n]$. For $u \in \mathbb{R}^n$, $N \in \mathbb{N}$, and vectors $\zeta_k \overset{\text{i.i.d.}}{\sim} \mathbb{N}(0, I_n)$ for $k \in [N]$, the scalar $v = \mathbf{RandProj}(u, N) \stackrel{\text{def}}{=} \frac{1}{N}\sum_{k=1}^N (u^\top \zeta_k)^2$. This definition gives us the property $\mathbb{E}v = \|u\|_2^2$, by using linearity of expectation. We overload the definition of **RandProj** by applying it to symmetric matrices: given $A \in \mathbb{S}^n$, we use $\mathbf{RandProj}(A, N)$ to denote the diagonal matrix obtained by applying **RandProj** to each row of A. Then the diagonal matrix $B = \mathbf{RandProj}(A, N)$ satisfies the property $\mathbb{E}B = \mathbf{diag}(A)^2$. We use $\widetilde{O}$ to denote polylogarithmic factors. The superscript * denotes optimality for variables and Fenchel conjugate for functions.

Fact 2.1.1 ([ALO16a]). *Given $A \geq 0$, $B \in \mathbb{S}^n$, and $\alpha \in [0, 1]$, the inequality $\text{Tr}\left(BA^\alpha BA^{1-\alpha}\right) \leq A \bullet B^2$ holds.*

Fact 2.1.2 ([Wil67]). *For a symmetric matrix-valued function $X(t)$ with argument scalar t, we have $\frac{d}{dt}\exp(X(t)) = \int_{\alpha=0}^1 \exp(\alpha X(t))\frac{d}{dt}X(t)\exp((1-\alpha)X(t))d\alpha$.*

Setup. Our underlying algorithm to solve (2.1.2) is a slight variant of lazy mirror descent (also called Nesterov's Dual Averaging [Nes09]), which we term *approximate lazy mirror descent*. To solve $\min_{x\in\mathcal{X}} f(x)$ using this algorithm, select a mirror map $\Phi : \mathcal{D} \to \mathbb{R}$ and a norm; the associated Bregman Divergence is $\mathcal{D}_\Phi(x, y) \stackrel{\text{def}}{=} \Phi(x) - \Phi(y) - \langle \nabla\Phi(y), x - y\rangle$; set $x^{(1)} \in \text{argmin}_{\mathcal{X}\cap\mathcal{D}}\Phi(x)$ and $z^{(1)} \in \nabla^{-1}\Phi(0)$. We repeat, in succession, the gradient update, $\nabla\Phi(z^{(t+1)}) = \nabla\Phi(z^{(t)}) - \eta\nabla f(x^{(t)})$, and the *approximate* projection, finding $\widetilde{x}^{(t+1)}$ satisfying $\mathbb{E}\|\widetilde{x}^{(t+1)} - x^{(t+1)}\| \leq \delta$, where $x^{(t+1)} \in \text{argmin}_{x\in\mathcal{X}\cap\mathcal{D}}\mathcal{D}_\Phi(x, z^{(t+1)})$. This is displayed in the appendix in Algorithm A.2.1 with its convergence guarantee in Theorem 2.1.3.

Theorem 2.1.3 (Convergence of Lazy Mirror Descent). *Fix a norm $\|\cdot\|$. Given an α-strongly convex mirror map $\Phi : \mathcal{D} \to \mathbb{R}$ and a convex, G-Lipschitz objective $f : \mathcal{X} \to \mathbb{R}$, run Algorithm A.2.1 with step size η and $\mathbb{E}\|x^{(t)} - \widetilde{x}^{(t)}\| \leq \delta$. Let $D \stackrel{def}{=} \sup_{x\in\mathcal{X}\cap\mathcal{D}}\Phi(x) - \inf_{x\in\mathcal{X}\cap\mathcal{D}}\Phi(x)$ and $x^* = \arg\min_{\mathcal{X}} f(x)$. Then,*

Algorithm A.2.1, after T iterations, returns $\widetilde{x}^{t^*}$, satisfying

$$\mathbb{E} f(\widetilde{x}^{(t^*)}) - f(x^*) \leq \frac{D}{T\eta} + \frac{2\eta G^2}{\alpha} + \delta G. \tag{2.1.6}$$

Lemma 2.1.4. *Given $C \in \mathbb{R}^{n\times n}$ and $0 \preceq X$, let $\rho \in \mathbb{R}^n$ with $\rho_i = \sum_{j=1}^n |C_{ij}|$; diagonal matrix S with $S_{ii} = \min(1/\sqrt{\rho_i}, 1/\sqrt{X_{ii}})$ for $i \in [n]$; $\widehat{X} = SXS$; $\widehat{C} = \mathbf{diag}(1/\sqrt{\rho})C\mathbf{diag}(1/\sqrt{\rho})$. Then, $\widehat{X} \succeq 0$, $\widehat{X}_{ii} \leq 1$ for all $i \in [n]$, and $\widehat{C} \bullet X - \sum_{i=1}^n (X_{ii} - \rho_i)^+ \leq C \bullet \widehat{X}$.*

2.2 Our Approach

We present below our main result.

Theorem 2.2.1 (Main Result)**.** *Given $C \in \mathbb{R}^{n\times n}$ with $m \geq n$ non-zero entries and $0 < \varepsilon \leq \frac{1}{2}$, we can find, in time $\widetilde{O}(m/\varepsilon^{3.5})$ and with high probability, a matrix $Y \in \mathbb{S}^n$ with $O(m)$ non-zero entries and a diagonal matrix $S \in \mathbb{R}^{n\times n}$ so that[3] $\widetilde{X}^* \stackrel{def}{=} S \cdot \exp Y \cdot S$ satisfies $\widetilde{X}^* \succeq 0$, $\widetilde{X}^*_{ii} \leq 1$ for $i \in [n]$, and $C \bullet \widetilde{X}^* \geq C \bullet X^* - \varepsilon \sum_{i,j} |C_{ij}|$, where X^* is an optimal solution of (2.1.1).*

Our algorithm achieving this result is presented in Algorithm 2.2.1, with parameters in Table 2.1. As a corollary, for the Max-Cut problem on a graph with n nodes and m edges, our algorithm gives a cut that is $(1-\varepsilon)\alpha_{GW}$ optimal[4], in time $\widetilde{O}(m/\varepsilon^{3.5})$, where $\alpha_{GW} \approx 0.878567$.

Algorithm 2.2.1 Our Algorithm

Input: Cost matrix $C \in \mathbb{R}^{n\times n}$, accuracy ε
Parameters: Displayed in Table 2.1

Initialize $t \leftarrow 0$, $Y^{(1)} \leftarrow \mathbf{0}$. Set $\widehat{C}$ and ρ from Lemma 2.1.4 and $\nabla f(X) = \mathbf{diag}(\mathbf{1}_{X\geq\rho}) - \widehat{C}$
for T_{outer} iterations **do**
  $t \leftarrow t + 1$
  $\widetilde{\exp}(\frac{1}{2}Y^{(t)}) \leftarrow \mathbf{ChebyExp}(\frac{1}{2}Y^{(t)}, \mathrm{T}_{\text{Cheby}}, \delta_{\text{Cheby}})$ ▷ Defined in Corollary A.4.17
  $\widetilde{X}^{(t)} \leftarrow \mathbf{RandProj}(\widetilde{\exp}(\frac{1}{2}Y^{(t)}), \mathrm{T}_{\text{jl}})$ ▷ High-accuracy projection
  $Y^{(t+1)} \leftarrow Y^{(t)} - \eta\nabla f(\widetilde{X}^{(t)})$ ▷ Gradient update
  for $t_i = 1 \to T_{\text{inner}}$ **do**
    $t \leftarrow t + 1$
    $\widehat{\theta}^{(t_i)} \leftarrow \mathbf{UpdateEstimator}(\widetilde{X}^{(t-1)}, Y^{(t-1)}, \varepsilon, \eta)$ ▷ See Algorithm 2.2.2
    $\widetilde{X}^{(t)}_{jj} \leftarrow (\sqrt{\widetilde{X}^{(t-1)}_{jj} + 1} + \widehat{\theta}^{(t_i)}_j)^2 - 1$ for $j \in [n]$ ▷ Constant-accuracy projection
    $Y^{(t+1)} \leftarrow Y^{(t)} - \eta\nabla f(\widetilde{X}^{(t)})$ ▷ Gradient update
  end for
end for
For $t^* \stackrel{\text{unif.}}{\sim} \{1, 2, \ldots, T_{\text{outer}}\}$, return $Y^{(t^*)}$ and S, where S is from Lemma 2.1.4.

Before proceeding to the proof sketch of Theorem 2.2.1, we call attention to a technical concept crucial to our analysis: *we add to (2.1.2) the constraint* $\operatorname{Tr} X \leq K$, *where* $K = 40n(\log n)^{10}$. The optimal X^* remains valid under this constraint because $\operatorname{Tr} X^* = n < K$. We show, in Lemma 2.2.8, that

[3] Since $\widetilde{X}^*$ can be dense, we represent it implicitly by only returning the matrices Y and S.
[4] Assuming the Unique Games Conjecture, this is the best we can hope for Max-Cut [KKMO07].

Parameter	Value	Proof
Diameter D	$K \log K$	Lemma A.4.1
Strong convexity α	$1/(4K)$	Lemma 2.2.9
Step size η	$\frac{1}{8\times 10^4 (\log(n/\varepsilon))^{11}} \varepsilon^2$	Lemma A.4.24
Inner iteration count T_{inner}	ε^{-2}	Section A.4.4
Outer iteration count T_{outer}	$\frac{1}{\varepsilon} \cdot 24 \times 10^5 (\log(n/\varepsilon))^{11} \log n$	Lemma 2.2.8
JL projection count T_{jl}	$(2 \times 10^5) \cdot (\log n)^{21} \cdot \varepsilon^{-2}$	Lemma A.4.24
Chebyshev approximation degree $\mathrm{T}_{\text{Cheby}}$	$150 \log(n/\varepsilon) \cdot \varepsilon^{-1/2}$	Lemma A.4.19
Chebyshev approximation accuracy δ_{Cheby}	$(\varepsilon/n)^{401}$	Lemma A.4.19

Table 2.1: All Algorithm 2.2.1 parameters and where their values are set. $K = 40n(\log n)^{10}$.

throughout our algorithm, this inequality remains inactive. Coupled with the Legendre dual of our mirror map $\Phi(X) = X \bullet \log X - \operatorname{Tr} X$, this fact results in $X = \exp(Y)$ (Lemma A.4.25). Since the gradient of our objective in (2.1.2) requires computing only the diagonal entries of the primal iterate, approximate mirror descent requires estimators only for the *diagonal entries* of $\exp(Y)$.

Proof Sketch of Theorem 2.2.1. We now compute the run time of Algorithm 2.2.1, thus proving Theorem 2.2.1. In doing so, we provide intuition for the parameters in Table 2.1. This sketch assumes that we are in iteration t and drops all superscripts.

(1) To compute $\exp(Y)_{ii}$, we first approximate $\widetilde{\exp}(Y/2)$ to ε-accuracy using Chebyshev polynomials. We show in Lemma A.4.18 that the spectrum of Y lies in the range $[-\widetilde{O}(1/\varepsilon), \widetilde{O}(1)]$, which allows for Chebyshev approximation with $\widetilde{O}(1/\sqrt{\varepsilon})$ terms, thus giving the cost of each projection to be $\widetilde{O}(m/\sqrt{\varepsilon})$. The upper bound of $\widetilde{O}(1)$ on the spectrum is critical to getting this cost, because in case of a symmetric range of $[-\widetilde{O}(1/\varepsilon), \widetilde{O}(1/\varepsilon)]$, the number of terms required would be $\widetilde{O}(1/\varepsilon)$. The $\widetilde{O}(1/\sqrt{\varepsilon})$ terms is in contrast with the $\widetilde{O}(1/\varepsilon)$ required for Taylor approximation. Having approximated $\widetilde{\exp}(Y/2)$, we then estimate each $\exp(Y)_{ii}$ with $\widetilde{O}(1/\varepsilon^2)$ projections via the JL sketch in the high-accuracy steps and with $\widetilde{O}(1)$ randomized projections in the T_{inner} low-accuracy steps. Therefore the total cost of the algorithm over T_{outer} iterations is roughly $T_{\text{outer}} \cdot (m/\sqrt{\varepsilon}) \cdot (1/\varepsilon^2 + T_{\text{inner}})$. From this expression, the optimal choice of T_{inner} (up to polylogarithmic factors) is $T_{\text{inner}} = 1/\varepsilon^2$.

(2) Due to the small bias and variance of our estimator, after T_{inner} inner iterations, the estimated iterate is roughly within εK distance of the true iterate. Thus, the condition in Theorem 2.1.3 is satisfied, and its error bound applies at the end of our algorithm: $\mathbb{E} f(\widetilde{X}^*) - f(X^*) \le D/(T\eta) + 2\eta G^2/\alpha + \delta G$. Using D, G, and α from Table 2.1 and T_{inner} from Step 1 and bounding by εK, this inequality simplifies to $\varepsilon^2/(\eta T_{\text{outer}}) + \eta \le \varepsilon$.

(3) The step size η is chosen by studying the error generated in each estimation step versus the error our framework can tolerate. Estimating $(\exp(Y + \Delta))_{ii}$ from $(\exp Y)_{ii}$ via a first-order approximation accrues an error of $\operatorname{Tr}(\Delta \exp Y)$. Applying Hölder's inequality, the value of G, and the trace bound enforced by Lemma 2.2.8 yields $\operatorname{Tr}(\Delta \exp Y) \le \eta K$. Therefore, after T_{inner} iterations, the variance of the error is $T_{\text{inner}} \eta^2 K^2$. Equivalently, the overall error after T_{inner} iterations is $\sqrt{T_{\text{inner}}} \eta K$. For this to be bounded by εK, we must have $\eta \le \varepsilon/\sqrt{T_{\text{inner}}}$. Plugging in T_{inner} from Step 1 gives $\eta \approx \varepsilon^2$.

(4) The value of η from Step 3 and the inequality from Step 2 give $T_{\text{outer}} \approx 1/\varepsilon$. Plugging this value of T_{outer} above gives the overall algorithm cost $\widetilde{O}(m/\varepsilon^{3.5})$. We show the cost breakdown comparing our algorithm to Arora-Kale in Table 2.2.

We boost our result to the high probability statement of Theorem 2.2.1 over multiple runs of the algorithm. We sidestep the issue of storage cost of $\widetilde{X}^*$ and cost of matrix-matrix products by dimension reduction techniques. This finishes the proof of our error guarantee. Lemma 2.1.4 implies that $\widetilde{X}^* \succeq 0$ and satisfies the diagonal constraints. Table 2.2 shows the steps in our algorithm that help us improve upon the runtime of [AK07]. □

Table 2.2: Comparing [AK07] to our algorithm.

	[AK07]	**Algorithm 2.2.1**		
	(Previous best)	Low-accuracy steps	+	High-accuracy steps
Number of iterations	$\widetilde{O}(\varepsilon^{-2})$	$\widetilde{O}(\varepsilon^{-3})$	+	$\widetilde{O}(\varepsilon^{-1})$
Number of projections per iteration	$\widetilde{O}(\varepsilon^{-2})$	$\widetilde{O}(1)$	+	$\widetilde{O}(\varepsilon^{-2})$
Cost per projection	$O(m\varepsilon^{-1})$	$\widetilde{O}(m\varepsilon^{-1/2})$	+	$\widetilde{O}(m\varepsilon^{-1/2})$
Total Cost	$\widetilde{O}(m\varepsilon^{-5})$	$\widetilde{O}(m\varepsilon^{-3.5})$	+	$\widetilde{O}(m\varepsilon^{-3.5})$

2.2.1 Our Estimator

In this section, we consider the t_i'th iteration in the inner loop of Algorithm 2.2.1; suppose this is the t'th overall iteration. For now, we drop all superscripts and fix the notation.

Definition 2.2.2. *Let* $\Delta = -\eta \nabla f(X)$, $Y_s = Y + s\Delta$ *for* $s \in [0,1]$, $\bar{\tau} = 1 - \tau$, $\delta_{\exp} = \frac{4800\varepsilon^{401}}{n^{390}}$, $\theta_{1_i} = (\exp(Y_s)_{ii} + 1)^{-1/2}$, $\theta_{2_i} = \frac{1}{2}(\exp(\bar{\tau}Y_s)\Delta \exp((\tau - 1/2)Y_s)\exp((1/2)Y_s))_{ii}$, $b_{1_i} = \theta_{1_i}(2\delta_{\exp} + \sqrt{2}(1 + 2\delta_{\exp})(\varepsilon/n)^{400})$, *and* $b_{2_i} = 15\delta_{\exp}\eta K$.

To construct an estimator for the update from $\exp(Y)$ to $\exp(Y + \Delta)$, we estimate the update in $\sqrt{(\exp Y)_{ii} + 1}$ using the Fundamental Theorem of Calculus. The motivation for this choice of function is two-fold: (1) because of the square root, the variance of error in update is controlled by the trace of the matrix exponential, which in turn is bounded by Lemma 2.2.8; (2) the update term has the derivative of the function at the current term in it, and since the derivative of the square root is the inverse square root, we need to use $\sqrt{\exp(Y)_{ii} + 1}$ instead of $\sqrt{\exp(Y)_{ii}}$ to prevent the update term from becoming unbounded in case $\exp(Y)_{ii}$ is too small. By chain rule, Fact 2.1.2, and the fundamental theorem of Calculus,

$$\sqrt{(\exp(Y+\Delta))_{jj} + 1} = \sqrt{(\exp(Y))_{jj} + 1} + \underbrace{\int_{s=0}^{1} \underbrace{((\exp Y_s)_{jj} + 1)^{-1/2}}_{\stackrel{\text{def}}{=} \theta_{1_j};\ \text{estimated using } \widehat{\theta}_{1_j}} \underbrace{\tfrac{1}{2}\left(\int_{\tau=0}^{1} \exp(\tau Y_s)\Delta \exp(\bar{\tau}Y_s)d\tau\right)_{jj}}_{\stackrel{\text{def}}{=} \theta_{2_j};\ \text{estimated using } \widehat{\theta}_{2_j}} ds}_{\stackrel{\text{def}}{=} \theta_j;\ \text{estimated using } \widehat{\theta}_j}. \tag{2.2.1}$$

As indicated in Equation (2.2.1), we split the quantity to be estimated into two parts, separately estimating each. Estimating the first part, $\widehat{\theta}_{1_j}$, requires first estimating $\exp(Y_s)_{jj} + 1$ using a JL sketch and then passing through the following Taylor approximation for the function $g(u) = u^{-1/2}$, where

$g^{(k)}(x)$ is the k'th derivative of g at x,

$$\textbf{InvSqrt}(\widetilde{X}, N) \stackrel{\text{def}}{=} \sum_{k=0}^{N-1} \frac{1}{k!} g^{(k)}(x_0) \prod_{j=1}^{k} (x_{k,j} - x_0), \text{ where } x_0, x_{k,j} \overset{\text{i.i.d.}}{\sim} \widetilde{X}. \tag{2.2.2}$$

Since $\widehat{\theta}_{1_j}$ must be *unbiased*, it is essential to do the Taylor approximation instead of simply evaluating $g(u) = u^{-1/2}$ at the estimator of $\exp(Y_s)_{jj} + 1$. Indeed, for a general f and a random variable $\widetilde{x}$ that is an unbiased estimator of x, $\mathbb{E} f(\widetilde{x}) = f(\mathbb{E}\widetilde{x})$ does not hold, as evidenced by Jensen's inequality; on the other hand, the intuition for the quantity from Equation (2.2.2) to be unbiased is that each term in the sum is a product of independent, unbiased random variables. We estimate θ_{2_j} by splitting it into carefully chosen parts and applying the JL sketch. Algorithm 2.2.2 is the complete estimator.

Algorithm 2.2.2 UpdateEstimator(Primal X, dual Y, accuracy ε, step size η)

1: Parameters $\mathrm{T}_{\mathrm{est}_{\mathrm{jl}}} = 2^{22}10^4(\log(n/\varepsilon))^2$ and $\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}} = 1600\log(n/\varepsilon)$ (set in Lemma 2.2.4)
2: Sample s and τ uniformly from $[0,1]$. Compute Δ and Y_s as per Definition 2.2.2. Let $\widetilde{X}_s = \textbf{RandProj}(\widetilde{\exp}(Y_s/2), \mathrm{T}_{\mathrm{est}_{\mathrm{jl}}})$. Sample $\zeta \sim \mathbb{N}(0, I_n)$.
3: Compute $\widehat{\theta}_{1_j} = \textbf{InvSqrt}(\widetilde{X}_{s_{jj}} + 1, \mathrm{T}_{\mathrm{est}_{\mathrm{isq}}})$ for $j \in [n]$.
4: Compute $\widehat{\theta}_{2_j} = \frac{1}{2}(\widetilde{\exp}((\tau - \frac{1}{2})Y_s)\Delta\widetilde{\exp}(\bar{\tau}Y_s)\zeta)_j\,(\widetilde{\exp}(Y_s/2)\zeta)_j$ for $j \in [n]$.
5: Return the overall estimator, $\widehat{\theta}_j = \widehat{\theta}_{1_j}\widehat{\theta}_{2_j}$, for $j \in [n]$. ▷ Coordinate-wise product

Properties of ohe estimator. The bounds on bias and variance of the estimator, as required by Theorem 2.2.1, are stated in Lemma 2.2.3. Since $\widehat{\theta}$ is constructed from $\widehat{\theta}_1$ and $\widehat{\theta}_2$, we first state their properties and use them to sketch a proof of Lemma 2.2.3.

Lemma 2.2.3. *The estimator $\widehat{\theta}^{(t)}$ has the following bounds on its first and second moments.*

(1) $|\mathbb{E}\widehat{\theta}_i - \int_{s=0}^{1}\int_{\tau=0}^{1} \theta_{1_i}\theta_{2_i} ds d\tau| \le b_{1_i}\theta_{2_i} + b_{2_i}\theta_{1_i} + b_{1_i}b_{2_i}$ *for* $i \in [n]$.
(2) $\mathbb{E}\|\widehat{\theta}\|_2^2 \le 19600\log(n/\varepsilon)K\eta^2 + 147000K^2\eta^2\delta_{\exp}$.

Lemma 2.2.4. *Given* $\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}} = 1600\log(n/\varepsilon)$, $\mathrm{T}_{\mathrm{est}_{\mathrm{jl}}} = 2^{14}\mathrm{T}^2_{\mathrm{est}_{\mathrm{isq}}}$, $Z \in \mathbb{S}^n$, *and* $\varepsilon \in (0, 1/2)$, *let* $\widetilde{Z^2} = \textbf{\textit{RandProj}}(Z, \mathrm{T}_{\mathrm{est}_{\mathrm{jl}}})$ *and* $\widehat{\theta}_{1_i} \sim \textbf{\textit{InvSqrt}}((\widetilde{Z^2})_{ii} + 1, \mathrm{T}_{\mathrm{est}_{\mathrm{isq}}})$ *for* $i \in [n]$. *Then,*

(1) *The first moment satisfies* $\left|\mathbb{E}\widehat{\theta}_{1_i} - \frac{1}{\sqrt{(Z^2)_{ii}+1}}\right| \le \frac{\sqrt{2}(\varepsilon/n)^{400}}{\sqrt{(Z^2)_{ii}+1}}$.
(2) *The second moment satisfies* $\mathbb{E}|\widehat{\theta}_{1_i}|^2 \le \frac{1}{(Z^2)_{ii}}1630\log(n/\varepsilon)$.

Lemma 2.2.5. *Consider* Z_1, Z_2, Z, *and* Δ *all in* $\mathbb{S}^n$. *Sample* $\zeta \sim \mathcal{N}(\mathbf{0}, I_n)$, *and define* $\widehat{\theta}_2 \in \mathbb{R}^n$ *as* $\widehat{\theta}_{2_i} = (Z_1\Delta Z_2\zeta)_i\,(Z\zeta)_i$. *Define* $\theta_{2_i} \stackrel{\text{def}}{=} (Z_1\Delta Z_2 Z)_{ii}$. *Then for* $i \in [n]$:

(1) *The first moment satisfies* $\mathbb{E}\widehat{\theta}_{2_i} = \theta_{2_i}$
(2) *The second moment satisfies* $\mathbb{E}|\widehat{\theta}_{2_i}|^2 \le 3\left(Z_1\Delta Z_2^2\Delta Z_1\right)_{ii}\left(Z^2\right)_{ii}$.

Proof sketch for Lemma 2.2.3. By construction, $\mathbb{E}_{s,\tau,\zeta_1,\zeta_2}\|\widehat{\theta}\|_2^2 = \int_{s=0}^1 \int_{\tau=0}^1 \sum_{i=1}^n \mathbb{E}_{\zeta_1}|\widehat{\theta}_{1_i}|^2 \mathbb{E}_{\zeta_2}|\widehat{\theta}_{2_i}|^2 ds d\tau$. Plugging in the second moment bounds from Lemma 2.2.4 and Lemma 2.2.5 gives

$$\mathbb{E}_{s,\tau,\zeta_1,\zeta_2}\|\widehat{\theta}\|_2^2 = 4890 \log(n/\varepsilon) \int_{s=0}^1 \int_{\tau=0}^1 \mathrm{Tr}(\widetilde{\exp}(2\bar{\tau} Y_s)\Delta \widetilde{\exp}((2\tau-1)Y_s)\Delta) ds d\tau.$$

This step is made possible by the careful choice of split in $\widehat{\theta}_2$ that enable cancellations of $\frac{1}{(\widetilde{\exp} Y_s)_{ii}}$ and $(\widetilde{\exp} Y_s)_{ii}$. Applying Fact 2.1.1 and the fact that $\widetilde{\exp} Y_s$ is close to the true $\exp Y_s$, the above trace term is bounded by $\mathrm{Tr}\left(\exp(Y + s\Delta)\Delta^2\right)$ (plus a small error term). Applying Hölder's Inequality, Lemma 2.2.8, and values of η and G completes the proof. □

To provide proof sketches of Lemma 2.2.4 and Lemma 2.2.5, we need two technical lemmas (proved in the Appendix) about **RandProj** and **InvSqrt**, the main workhorses for our estimators.

Lemma 2.2.6. *Consider a positive random variable x sampled from a distribution X with mean μ and variance σ^2. For some integer $k > 0$, construct the distribution $\mathcal{G}(X) = \textbf{\textit{InvSqrt}}(X, k)$ defined in Equation (2.2.2). Then the random variable $g \sim \mathcal{G}(X)$ satisfies*

(1) $|\mathbb{E}g - \mu^{-1/2}| \le \mathbb{E}\left(\frac{|x-\mu|^k}{\min(\mu,x)^{k+1/2}}\right)$

(2) $\mathbb{E}|g|^2 \le k\sum_{j=0}^{k-1} \mathbb{E}\left(\frac{\left(\sigma^2 + (\mu-x)^2\right)^j}{x^{2j+1}}\right)$.

Lemma 2.2.7. *Given $u \in \mathbb{R}^n$ such that $\mu \overset{def}{=} \|u\|_2^2 \neq 0$, and positive integers $k > 1$ and $N \ge 4k+6$, the following are true for x sampled from $X = \textbf{\textit{RandProj}}(u, N)$.*

(1) $\mathbb{E}x = \mu$

(2) $\sigma^2 \overset{def}{=} \mathbb{E}(x-\mu)^2 = \frac{2\mu^2}{N}$

(3) $\mathbb{E}\left(\frac{\left(\sigma^2 + (x-\mu)^2\right)^k}{\min(x,\mu)^{2k+1}}\right) \le \frac{1}{\mu}\left(\frac{e^{N/2}}{2^{N-17k}} + \frac{2^{13k}k^{2k}}{N^k}\right)$

Proof sketches of Lemma 2.2.4 and Lemma 2.2.5. Consider $x \sim \widetilde{Z^2}_{ii}$. By Lemma 2.2.7, $\mathbb{E}x = Z_{ii}^2$. This satisfies the bias requirement of Lemma 2.2.6, and therefore by applying Lemma 2.2.6, Jensen's inequality, and a slight modification of (3) in Lemma 2.2.7, we obtain the following inequalities.

$$\begin{aligned}
\left|\mathbb{E}\widehat{\theta}_{1_i} - \frac{1}{\sqrt{1+(Z^2)_{ii}}}\right| &\le \mathbb{E}\left(\frac{\left|x-(Z^2)_{ii}\right|^{\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}}}{\min(x+1,(Z^2)_{ii}+1)^{\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}+\frac{1}{2}}}\right) \\
&\le \sqrt{\mathbb{E}\frac{(x-(Z^2)_{ii})^{2\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}}}{\min(x+1,(Z^2)_{ii}+1)^{2\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}+1}}} \\
&\le \sqrt{\frac{1}{(Z^2)_{ii}+1}\left(\frac{e^{\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}/2}}{2^{\mathrm{T}_{\mathrm{est}_{\mathrm{jl}}}-17\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}}} + \frac{2^{13\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}}\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}{}^{2\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}}}{\mathrm{T}_{\mathrm{est}_{\mathrm{jl}}}{}^{\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}}}\right)}.
\end{aligned}$$

The values of $T_{\text{est}_{\text{isq}}}$ and $T_{\text{est}_{\text{jl}}}$ from Algorithm 2.2.2 give the final bias bound. The second moment bound follows similarly, and the properties of $\widehat{\theta}_2$ follow from those of the Gaussian distribution. □

2.2.2 Technical Concepts: Domain Expansion and Strong Convexity

In this section we state and sketch the proofs of two key technical concepts: (1) the addition of the trace constraint as described before the proof of Theorem 2.2.1, and (2) the value of the strong convexity parameter of our mirror map over this new domain.

Lemma 2.2.8. *For any iteration t of Algorithm 2.2.1, $\widetilde{X}^{(t)}$ satisfies $\operatorname{Tr}\widetilde{X}^{(t)} < K$ for $K = 40n(\log n)^{10}$.*

Proof sketch. We assume that for any iteration t, the primal iterate is close to the optimal point and satisfies $\|\widetilde{X}^{(t)} - X^*\| \leq 38n\,(\log n)^{10}$. In Algorithm 2.2.1, $Y^{(1)} = 0$ implies $\widetilde{X}^{(1)} = I$. We also know that the optimal point satisfies $\operatorname{Tr} X^* = n$. Therefore, in the base case, $\|\widetilde{X}^{(1)} - X^*\| \leq 2n \leq 38n\,(\log n)^{10}$. Suppose that the hypothesis is true for some $t = t'$. We complete the proof by first proving a weak bound for $\|\widetilde{X}^{(t)} - X^*\|$ using the triangle inequality of norms and then boosting our bound (thereby obtaining the stronger guarantee of the induction hypothesis) by invoking the strong convexity of the Bregman divergence. The full proof is presented in Section A.4.6. □

We now sketch the proof of the strong convexity parameter of our mirror map, the *generalized* negative entropy function, which has also been used in [AO15] and later in Chapter 8.

Lemma 2.2.9. *The function $\Phi(X) = X \bullet \log X - \operatorname{Tr} X$ is $\frac{1}{4K}$-strongly convex with respect to the nuclear norm over the domain $\mathcal{D} = \{X : X \succeq 0, \operatorname{Tr} X \leq K\}$.*

Proof sketch. We invoke the duality between strong convexity and smoothness by [KST09], the characterization of matrix smooth functions by [JN08], and the generalization of convexity of a permutation-invariant function on vectors to a spectral function on matrices by [Lew95]. Our proof requires the following definition.

Definition 2.2.10. *Define the vector functions $\psi_1(y) = \sum_{i=1}^n \exp y_i$, $\psi_2(y) = 2K \log \psi_1(y) - 2K \log(2K) + 2K$, $\psi(y) = \psi_1(y)$ if $\psi_1(y) \leq 2K$ and $\psi_2(y)$ otherwise; $\Psi(Y) = \Psi_1(Y)$ if $\Psi_1(Y) \leq 2K$ and $\Psi_2(Y)$ otherwise; and $\phi(x) = \sum_{i=1}^n x_i \log x_i - \sum_{i=1}^n x_i$. Define the corresponding matrix functions $\Psi_1(Y) = \operatorname{Tr} \exp Y$, $\Psi_2(Y) = 2K \log \Psi_1(Y) - 2K \log(2K) + 2K$, and $\Phi(X) = X \bullet \log X - \operatorname{Tr} X$.*

Our first step is to show that Ψ, the matrix version of ψ, satisfies the property $\Psi^*(Y) = \Phi(Y)$ over $\{Y : Y \succeq 0, \operatorname{Tr} Y \leq K\}$. To prove this, we first prove that ψ and its matrix version, Ψ, are both continuously differentiable at the boundary of definition of their respective two parts. We then show that ψ_1 and ψ_2 are convex; combining this with the claim about continuous differentiability implies convexity of ψ, which immediately extends to Ψ by a result of [Lew95]. We then show that ψ and ϕ satisfy $\psi_1^*(x) = \phi(x)$ for $x \in \mathbb{R}^n_+$, and given an input $x \in \{x : x_i \geq 0, \sum_{i=1}^n x_i \leq K\}$, the point y attaining the optimum in computing $\psi_1^*(x)$ lies in the *interior* of the set $\{y : \psi_1(y) \leq 2K\}$. Therefore, given an input $x \in \{x : x_i \geq 0, \sum_{i=1}^n x_i \leq K\}$, we invoke the preceeding facts to conclude that the point at which the value of $\psi^*(x)$ is attained must be the same as that for $\psi_1^*(x)$. This implies $\psi^*(x) = \psi_1^*(x)$ for $x \in \{x : x_i \geq 0, \sum_{i=1}^n x_i \leq K\}$. By a result of [Lew95], this extends to $\Psi^* = \Phi$ on $\{X : X \succeq 0, \operatorname{Tr} X \leq K\}$.

We then use [JN08] and continuous differentiability at the boundary to show that Ψ is $4K$-smooth in the operator norm which in turn implies, by [KST09], that Ψ^* is $1/(4K)$-strongly convex in the nuclear norm, finishing the proof. Our full proof is in Section A.4.1. □

Chapter 3

A Faster Interior Point Method for Semidefinite Programs

In Chapter 2, we saw a faster first-order algorithm for a specific SDP (the MaxCUT SDP). In this chapter, we expand our focus to the general class of SDPs. While the MaxCUT SDP studied in Chapter 2 is in itself quite important, general SDPs constitute a fundamental class of optimization problems with important applications running the gamut of approximation algorithms, polynomial optimization, robust learning, algorithmic rounding, and adversarial deep learning. We present a faster interior point method to solve generic SDPs with variable size $n \times n$ and m constraints in time $\widetilde{O}(\sqrt{n}(mn^2 + m^\omega + n^\omega))$, where ω is the exponent of matrix multiplication and ε is the relative accuracy. Our algorithm's runtime can be naturally interpreted as follows: $\widetilde{O}(\sqrt{n}\log(1/\varepsilon))$ is our algorithm's iteration complexity, mn^2 is the input size, and $m^\omega + n^\omega$ is the time to invert the Hessian and slack matrix in each iteration. In contrast to the algorithm in Chapter 2, which was a first-order method (with a polynomial dependence of the runtime on accuracy), our focus in this chapter is on second-order methods that yield runtimes with polylogarithmic dependence on the accuracy parameter. Despite this difference, the algorithms presented in these two chapters share the principle of "robust updates".

3.1 Introduction

Semidefinite programs (SDPs) constitute a class of convex optimization problems that optimize a linear objective over the intersection of the cone of positive semidefinite matrices with an affine space. SDPs generalize linear programs and have a plethora of applications in operations research, control theory, and theoretical computer science [VB96]. Applications include improved approximation algorithms for fundamental problems (e.g., Max-Cut [GW95], coloring 3-colorable graphs [KMS94], and sparsest cut [ARV09]), quantum complexity theory [JJUW11], robust learning and estimation [CG18, CDG19, CDGW19], algorithmic discrepancy and rounding [BDG16, BG17, Ban19], and polynomial optimization [Par00]. We formally define SDPs with $n \times n$ variables and m constraints:

Definition 3.1.1 (Semidefinite program)**.** *Given symmetric*[1] *matrices* $C, A_1, \cdots, A_m \in \mathbb{R}^{n\times n}$ *and* $b_i \in \mathbb{R}$ *for all* $i \in [m]$*, solve the convex optimization problem*

$$\text{maximize } C \bullet X \text{ subject to } X \succeq 0, A_i \bullet X = b_i \text{ for all } i \in [m] \tag{3.1.1}$$

where $A \bullet B := \sum_{i,j} A_{i,j}B_{i,j}$ *is the trace product.*

High-Accuracy Algorithms. Our goal is to solve SDPs to high accuracy (i.e., with runtimes depending *logarithmically* on the accuracy parameter). We discuss below two prominent methods that achieve this goal: cutting plane methods and interior point methods.

[1] We can assume that $C, A_1, \cdots, A_m$ are symmetric, since given any $M \in \{C, A_1, \cdots, A_m\}$, we have $\sum_{i,j} M_{ij}X_{ij} = \sum_{i,j} M_{ij}X_{ji} = \sum_{i,j}(M^\top)_{ij}X_{ij}$, and therefore we can replace M with $(M + M^\top)/2$.

The cutting plane method is an iterative algorithm for searching a convex set for the optimal point. It starts with a large convex set guaranteed to contain this point and, in each iteration, queries a separation oracle within this set. Based on the output of the separation oracle, the convex set is updated to a smaller one containing the subset with the optimal point. This process is repeated until the volume of this set becomes small enough and a point sufficiently close to the optimal is found. Since Khachiyan proved [Kha80] that the ellipsoid method solves linear programs in polynomial time, cutting plane methods have played a crucial role in both discrete and continuous optimization [GLS81a, GV02].

In contrast, interior point methods turn the original constrained optimization problem into a sequence of unconstrained optimization problems parametrized by a scaling factor that ascribes relative weight to optimizing the objective versus enforcing feasibility. The solutions to these successive problems form a well-defined *central path*. Since Karmarkar's proof [Kar84] that interior point methods can solve linear programs in polynomial time, these methods have become an active research area. Their iteration complexity is proportional to the complexity parameter of the barrier function used, which, for many commonly used barriers is the square root of the dimension, as opposed to the linear dependence on the dimension in cutting plane methods.

Since cutting plane methods use less structural information than interior point methods, they are slower at solving almost all problems where interior point methods are known to apply. However, SDPs remain one of the most fundamental optimization problems where the state of the art is, in fact, the opposite: the current fastest cutting plane method of [LSW15] solves a general SDP in time $O(m(mn^2 + m^2 + n^\omega))$, while the fastest SDP solvers based on interior point methods in the work of [NN92] and [Ans00] achieve runtimes of $O(\sqrt{n}(m^2n^2 + mn^\omega + m^\omega))$ and $O((mn)^{1/4}(m^4n^2 + m^3n^\omega))$, respectively, which are slower in the most common regime of $m \in [n, n^2]$ (see Table 3.2). This apparent paradox raises the following natural question:

How fast can SDPs be solved using interior point methods?

3.1.1 Our Results

We present a faster interior point method for solving SDPs. Our main result is the following theorem, the formal version of which is presented in Theorem B.3.1.

Theorem 3.1.2 (Main result, informal)**.** *There is an interior point method that solves a general SDP with variable size $n \times n$ and m constraints in time*[2] $O^*(\sqrt{n}(mn^2 + m^\omega + n^\omega))$.

Our runtime can be roughly interpreted as follows:

- $\sqrt{n}$ is the iteration complexity of the interior point method with the log barrier function.
- mn^2 is the input size.
- m^ω is the cost of inverting the Hessian of the log barrier.
- n^ω is the cost of inverting the slack matrix.

Thus, the terms in the runtime of our algorithm arise as a natural barrier to further speeding up SDP solvers. See Section 3.2.1, Section 3.3, and Section 3.3.1 for more detail.

Table 3.1 compares our result with previous SDP solvers. The first takeaway of this table and Theorem 3.1.2 is that our interior point method always runs faster than that in [NN92] and faster than that in [NN94] and [Ans00] when $m \geq n^{1/13}$. A second consequence is that whenever $m \geq \sqrt{n}$,

[2] We use O^* to hide $n^{o(1)}$ and $\log^{O(1)}(n/\epsilon)$ factors and $\widetilde{O}$ to hide $\log^{O(1)}(n/\epsilon)$ factors, where ϵ is the accuracy parameter.

our interior point method is faster than the current fastest cutting plane method [LSW15]. We note that $n \leq m \leq n^2$ is satisfied in most SDP applications known to us, such as classic graph optimization problems, experiment design in statistics and machine learning, and sum-of-squares problems. An explicit comparison to previous algorithms for $m = n$ and $m = n^2$ is shown in Table 3.2.

Table 3.1: Summary of key high-accuracy SDP algorithms. CPM stands for cutting plane method, and IPM, interior point method. We denote by n the size of the variable matrix and by $m \leq n^2$ the number of constraints. Our runtimes hide $n^{o(1)}$, $m^{o(1)}$ and poly $\log(1/\epsilon)$ factors, where ϵ is the accuracy parameter. [Ans00] simplifies the proofs in [NN94, Section 5.5]. Neither [Ans00] nor [NN94] explicitly analyzed their runtimes, and their runtimes shown here are our best estimates.

Year	Authors	Method	Iteration Complexity	Cost Per Iteration
1976	[YN76, Sho77, Kha80]	CPM	m^2	$mn^2 + m^2 + n^\omega$
1988	[KTE88, NN89]	CPM	m	$mn^2 + m^{3.5} + n^\omega$
1989	[Vai89a]	CPM	m	$mn^2 + m^\omega + n^\omega$
1992	[NN92]	IPM	$\sqrt{n}$	$m^2n^2 + mn^\omega + m^\omega$
1994	[NN94, Ans00]	IPM	$(mn)^{1/4}$	$m^4n^2 + m^3n^\omega$
2003	[KM03]	CPM	m	$mn^2 + m^\omega + n^\omega$
2015	[LSW15, JLSW20]	CPM	m	$mn^2 + m^2 + n^\omega$
2020	Our result	IPM	$\sqrt{n}$	$mn^2 + m^\omega + n^\omega$

Table 3.2: Total runtimes for the algorithms in Table 3.1 for SDPs when $m = n$ and $m = n^2$, where n is the size of matrices, and m is the number of constraints. The runtimes shown in the table hide $n^{o(1)}$, $m^{o(1)}$ and poly $\log(1/\epsilon)$ factors, where ϵ is the accuracy parameter and assume ω to equal its currently best known upper bound of 2.373.

Year	**References**	**Method**	**Runtime**	
			$m = n$	$m = n^2$
1979	[Sho77, YN76, Kha80]	CPM	n^5	n^8
1988	[KTE88, NN89]	CPM	$n^{4.5}$	n^9
1989	[Vai89a]	CPM	n^4	$n^{6.746}$
1992	[NN92]	IPM	$n^{4.5}$	$n^{6.5}$
1994	[NN94, Ans00]	IPM	$n^{6.5}$	$n^{10.75}$
2003	[KM03]	CPM	n^4	$n^{6.746}$
2015	[LSW15, JLSW20]	CPM	n^4	n^6
2020	Our result	IPM	$n^{3.5}$	$n^{5.246}$

In the more general case where the SDP might not be dense, where $\text{nnz}(A)$ is the input size (i.e., the total number of non-zeroes in all matrices A_i for $i \in [m]$ and C), our interior point method runs faster than the current fastest cutting plane method[LSW15], which runs in time $\widetilde{O}(m \cdot (\text{nnz}(A) + m^2 + n^\omega))$.

Theorem 3.1.3 (Comparison with Cutting Plane Method)**.** *When $m \geq n$, there is an interior point method that solves an SDP with $n \times n$ matrices, m constraints, and $\text{nnz}(A)$ input size, faster than the current best cutting plane method [LSW15], over all regimes of $\text{nnz}(A)$.*

3.1.2 Related Work

Linear Programming. Linear Programming is a class of fundamental problems in convex optimization. There is a long list of work focused on fast algorithms for linear programming [Dan47, Kha80, Kar84, Vai87, Vai89b, LS14, LS15, Sid15, Lee16, CLS19, Bra20, BLSS20].

Cutting Plane Method. Cutting plane method is a class of optimization methods that iteratively refine a convex set that contains the optimal solution by querying a separation oracle. Since its introduction in the 1950s, there has been a long line of work on obtaining fast cutting plane methods [Sho77, YN76, Kha80, KTE88, NN89, Vai89a, AV95, BV02, LSW15].

First-Order SDP Algorithms. As the focus of this chapter, cutting plane methods and interior point methods solve SDPs in time that depends *logarithmically* on $1/\epsilon$, where ϵ is the accuracy parameter. A third class of algorithms, the *first-order methods*, solve SDPs at runtimes that depend *polynomially* on $1/\epsilon$. While having worse dependence on $1/\epsilon$ compared to IPM and CPM, these first-order algorithms usually have better dependence on the dimension. There is a long list of work on first-order methods for general SDP or special classes of SDP (e.g. Max-Cut SDP [AK07, GH16b, AZL17b, CDST19b, LP20, YTF+19b], positive SDPs [JY11, PT12, ALO16b, JLL+20b].)

3.2 An Overview of Our Techniques

By removing redundant constraints, we can, without loss of generality, assume $m \leq n^2$ in the primal formulation of the SDP (3.1.1). Thereafter, instead of solving the primal SDP, which has variable size $n \times n$, we solve its dual formulation, which has dimension $m \leq n^2$:

$$\text{minimize } b^\top y \text{ subject to } S = \sum_{i=1}^{m} y_i A_i - C, \text{ and } S \succeq 0. \tag{3.2.1}$$

Interior point methods solve (3.2.1) by minimizing the penalized objective function:

$$\text{minimize}_{y \in \mathbb{R}^m} f_\eta(y), \text{ where } f_\eta(y) := \eta \cdot b^\top y + \phi(y), \tag{3.2.2}$$

where $\eta > 0$ is a parameter and $\phi : \mathbb{R}^m \to \mathbb{R}$ is a *barrier* function that approaches infinity as y approaches the boundary of the feasible set $\{y \in \mathbb{R}^m : \sum_{i=1}^m y_i A_i \succeq C\}$. These methods first obtain an approximate minimizer of f_η for some small $\eta > 0$, which they then use as an initial point to minimize $f_{(1+c)\eta}$, for some constant $c > 0$, via the Newton method. This process repeats until the parameter η in (3.2.2) becomes sufficiently large, at which point the minimizer of f_η is provably close to the optimal solution of (3.2.1). The iterates y generated by this method follow a *central path*. Different choices of the barrier function ϕ yield different run times in solving (3.2.2), as we next describe.

The log barrier. Nesterov and Nemirovski [NN92] used the *log barrier* function,

$$g(y) := -\log\det\left(\sum_{i=1}^{m} y_i A_i - C\right), \tag{3.2.3}$$

in (3.2.2) and, in $O(\sqrt{n}\log(n/\epsilon))$ iterations, obtain a feasible dual solution y that satisfies $b^\top y \leq b^\top y^* + \epsilon$, where $y^* \in \mathbb{R}^m$ is the optimal solution for (3.2.1). Within each iteration, the costliest step is to compute the inverse of the Hessian of the log barrier function for the Newton step. For each $(j,k) \in [m] \times [m]$, the (j,k)-th entry of H is given by

$$H_{j,k} = \text{tr}\left[S^{-1} A_j S^{-1} A_k\right]. \tag{3.2.4}$$

The analysis of [NN92] first computes $S^{-1/2} A_j S^{-1/2}$ for all $j \in [m]$, which takes time $O^*(mn^\omega)$, and then calculates the m^2 trace products $\text{tr}\left[S^{-1} A_j S^{-1} A_k\right]$ for all $(j,k) \in [m] \times [m]$, each of which takes $O(n^2)$ time. Inverting the Hessian costs $O^*(m^\omega)$, which results in a total runtime of $O^*(\sqrt{n}(m^2 n^2 + mn^\omega + m^\omega))$.

The volumetric barrier. Vaidya [Vai89a] introduced the *volumetric barrier* for a polytope $\{x \in \mathbb{R}^n : Ax \geq c\}$, where $A \in \mathbb{R}^{m \times n}$ and $c \in \mathbb{R}^m$. Nesterov and Nemirovski [NN94] studied $V(y) = \frac{1}{2}\log\det\left(\nabla^2 g(y)\right)$, where $g(y)$ is the log barrier from Equation (3.2.3). This is the extension of the volumetric barrier to the positive semidefinite cone $\{y \in \mathbb{R}^m : \sum_{i=1}^m y_i A_i \succeq C\}$. It was shown in [NN94] that choosing $\phi(y) = \sqrt{n} \cdot V(y)$ in (3.2.2) makes the interior point method converge in $\widetilde{O}(\sqrt{m}n^{1/4})$ iterations, which is smaller than the $\widetilde{O}(\sqrt{n})$ iteration complexity of [NN92] when $m \leq \sqrt{n}$. They also studied the *combined volumetric-logarithmic barrier* $V_\rho(y) = V(y) + \rho \cdot g(y)$ and showed that taking $\phi(y) = \sqrt{n/m} \cdot V_\rho(y)$ for $\rho = (m-1)/(n-1)$ yields an iteration complexity of $\widetilde{O}((mn)^{1/4})$. When $m \leq n$, this iteration complexity is lower than $\widetilde{O}(\sqrt{n})$ of [NN92]. We refer readers to the much simpler proofs in [Ans00] for these results.

However, the volumetric barrier (and thus the combined volumetric-logarithmic barrier) leads to complicated expressions for the gradient and Hessian that make each iteration costly. For instance, the Hessian of the volumetric barrier is

$$\nabla^2 V(y) = 2Q(y) + R(y) - 2T(y),$$

where $Q(y)$, $R(y)$, and $T(y)$ are $m \times m$ matrices such that for each $(j,k) \in [m] \times [m]$,

$$\begin{aligned}
Q(y)_{j,k} &= \operatorname{tr}\left[\mathcal{A}H^{-1}\mathcal{A}^\top\left(\left(S^{-1}A_jS^{-1}A_kS^{-1}\right)\widehat{\otimes}S^{-1}\right)\right], \\
R(y)_{j,k} &= \operatorname{tr}\left[\mathcal{A}H^{-1}\mathcal{A}^\top\left(\left(S^{-1}A_jS^{-1}\right)\widehat{\otimes}\left(S^{-1}A_kS^{-1}\right)\right)\right], \\
T(y)_{j,k} &= \operatorname{tr}\left[\mathcal{A}H^{-1}\mathcal{A}^\top\left(\left(S^{-1}A_jS^{-1}\right)\widehat{\otimes}S^{-1}\right)\mathcal{A}H^{-1}\mathcal{A}^\top\left(\left(S^{-1}A_kS^{-1}\right)\widehat{\otimes}S^{-1}\right)\right].
\end{aligned} \tag{3.2.5}$$

Here, $\mathcal{A} \in \mathbb{R}^{n^2 \times m}$ is the $n^2 \times m$ matrix whose ith column is obtained by flattening A_i into a vector of length n^2, and $\widehat{\otimes}$ is the symmetric Kronecker product

$$A\widehat{\otimes}B := \frac{1}{2}(A \otimes B + B \otimes A),$$

where $\otimes$ is the Kronecker product (see Section B.1 for a formal definition). Due to the complicated formulas in Equation (3.2.5), efficient computation of Newton step in each iteration of the interior point method is difficult; in fact, each iteration runs slower than the Nesterov-Nemirovski interior point method by a factor of m^2. Since most applications of SDPs known to us have the number of constraints m be at least linear in n, the total runtime of interior point methods based on the volumetric barrier and the combined volumetric-logarithmic barrier is inevitably slow.

3.2.1 Our Techniques

Given the inefficiency of implementing the volumetric and volumetric-logarithmic barriers discussed above, this chapter uses the log barrier in Equation (3.2.3). We now describe some of our key techniques that improve the runtime of the Nesterov-Nemirovski interior point method [NN92].

Hessian computation via fast rectangular matrix multiplication. As noted earlier, the runtime bottleneck in [NN92] is in computing Equation (3.2.4), the Hessian of the log barrier. In [NN92], each of these m^2 entries is computed separately, resulting in a per iteration cost of $O(m^2n^2)$.

We show below how to speed up this computation using fast rectangular matrix multiplication.

The expression from Equation (3.2.4) can be re-written as

$$H_{j,k} = \mathrm{tr}\Big[S^{-1/2} A_j S^{-1/2} \cdot S^{-1/2} A_k S^{-1/2}\Big]. \tag{3.2.6}$$

We first compute the key quantity $S^{-1/2} A_j S^{-1/2} \in \mathbb{R}^{n \times n}$ for all $j \in [m]$ by stacking all matrices $A_j \in \mathbb{R}^{n \times n}$ into a tall matrix of size $mn \times n$, and then compute the product of $S^{-1/2} \in \mathbb{R}^{n \times n}$ with this tall matrix. This matrix product can be computed in time $\mathcal{T}_{\mathrm{mat}}(n, mn, n)$[3] using fast rectangular matrix multiplication. We then flatten each $S^{-1/2} A_j S^{-1/2}$ into a row vector of length n^2, so that the j-th row $\mathcal{B}_j = \mathsf{vec}(S^{-1/2} A_j S^{-1/2})$, and stack all m vectors to form a matrix $\mathcal{B}$ of size $m \times n^2$. It follows that the Hessian can be computed as

$$H = \mathcal{B}\mathcal{B}^\top, \tag{3.2.7}$$

which costs $\mathcal{T}_{\mathrm{mat}}(m, n^2, m)$ by applying fast rectangular matrix multiplication. Leveraging recent developments in this area [GU18], this approach improves upon the runtime in [NN92].

So far, we have reduced the per iteration cost of $O^*(m^2 n^2 + mn^\omega)$ for Hessian computation down to

$$\mathcal{T}_{\mathrm{mat}}(n, mn, n) + \mathcal{T}_{\mathrm{mat}}(m, n^2, m).$$

Low rank update on the slack matrix The fast rectangular matrix multiplication approach noted above, however, is still not very efficient, because the Hessian must be computed from scratch in each iteration of the interior point method. If there are T iterations in total, it then takes time

$$T \cdot (\mathcal{T}_{\mathrm{mat}}(n, mn, n) + \mathcal{T}_{\mathrm{mat}}(m, n^2, m)).$$

To further improve the runtime, we need to efficiently update the Hessian for the current iteration from the Hessian computed in the previous one. Generally, this is not possible, as the slack matrix $S \in \mathbb{R}^{n \times n}$ in Equation (3.2.6) might change arbitrarily in the Nesterov-Nemirovski interior point method.

To overcome this problem, we propose a new interior point method, Algorithm 3.2.1, that maintains, via Algorithm 3.2.2, an approximate slack matrix $\widetilde{S} \in \mathbb{R}^{n \times n}$, which is a spectral approximation of the true slack matrix $S \in \mathbb{R}^{n \times n}$ such that $\widetilde{S}$ admits a *low-rank update* in each iteration. Where needed, we will now use the subscript t to denote a matrix in the t-th iteration.

Our algorithm updates only the directions in which $\widetilde{S}_t$ deviates too much from S_{t+1}; the changes to S_t for the remaining directions are not propagated to $\widetilde{S}_{t+1}$. Such a selective update ensures a low-rank change in $\widetilde{S}_t$ even when S_t suffers from a full-rank update; it also guarantees the proximity of the algorithm's iterates to the central path. Specifically, for each iteration $t \in [T]$, we define the *difference matrix*

$$Z_t = S_t^{-1/2} \widetilde{S}_t S_t^{-1/2} - I \quad \in \mathbb{R}^{n \times n},$$

which captures how far the approximate slack matrix $\widetilde{S}_t$ is from the true slack matrix S_t. We maintain the invariant $\|Z_t\|_{\mathrm{op}} \leq c$ for some sufficiently small constant $c > 0$. In the $(t+1)$-th iteration when S_t gets updated to S_{t+1}, our construction of $\widetilde{S}_{t+1}$ involves a novel approach of zeroing out some of the largest eigenvalues of $|Z_t|$ to bound the rank of the update on the approximate slack matrix.

We prove that with this approach, the updates on $\widetilde{S} \in \mathbb{R}^{n \times n}$ over all $T = \widetilde{O}(\sqrt{n})$ iterations satisfy the

[3] See Section B.2 for the definition.

Algorithm 3.2.1 Main Algorithm $(n, m, \delta, \epsilon_N, C, A, b)$

Modify the SDP and obtain an initial dual solution y according to Lemma B.8.1
Initialize the step size $\eta \leftarrow 1/(n+2)$, total iteration count $T \leftarrow \frac{40}{\epsilon_N}\sqrt{n}\log\left(\frac{n}{\delta}\right)$, and true and approximate slack matrices $\widetilde{S} \leftarrow S \leftarrow \sum_{i\in[m]} y_i A_i - C$.
for iter $= 1 \to T$ **do**
    Update the central path parameter η as follows: $\eta_{\mathrm{new}} \leftarrow \eta\left(1 + \frac{\epsilon_N}{20\sqrt{n}}\right)$
    Compute the gradient: $g_{\eta_{\mathrm{new}}}(y)_j \leftarrow \eta_{\mathrm{new}} \cdot b_j - \mathrm{tr}\left[S^{-1} \cdot A_j\right]$
    Compute the Hessian: $\widetilde{H}_{j,k}(y) \leftarrow \mathrm{tr}\left[\widetilde{S}^{-1} \cdot A_j \cdot \widetilde{S}^{-1} \cdot A_k\right]$
    Update the dual variable y as follows: $\delta_y \leftarrow -\widetilde{H}(y)^{-1} g_{\eta_{\mathrm{new}}}(y)$, $y_{\mathrm{new}} \leftarrow y + \delta_y$
    Compute the true slack matrix: $S_{\mathrm{new}} \leftarrow \sum_{i\in[m]} (y_{\mathrm{new}})_i A_i - C$
    Compute the approximate slack matrix: $\widetilde{S}_{\mathrm{new}} \leftarrow \textsc{ApproxSlackUpdate}(S_{\mathrm{new}}, \widetilde{S})$
    $y \leftarrow y_{\mathrm{new}}$, $S \leftarrow S_{\mathrm{new}}$, $\widetilde{S} \leftarrow \widetilde{S}_{\mathrm{new}}$
end for
Return an approximate solution to the original SDP according to Lemma B.8.1

Algorithm 3.2.2 Approximate Slack Update$(S_{\mathrm{new}}, \widetilde{S})$

Initialize $\epsilon_S \leftarrow 0.01$, construct $Z_{\mathrm{mid}} \leftarrow S_{\mathrm{new}}^{-1/2} \cdot \widetilde{S} \cdot S_{\mathrm{new}}^{-1/2} - I$, and express $Z_{\mathrm{mid}} = U \cdot \Lambda \cdot U^\top = \sum_{i=1}^n \lambda_i u_i u_i^\top$
Let $\pi : [n] \to [n]$ be a sorting permutation such that $|\lambda_{\pi(i)}| \geq |\lambda_{\pi(i+1)}|$
if $|\lambda_{\pi(1)}| \leq \epsilon_S$ **then** ▷ $\widetilde{S}$ and S_{new} are spectrally close
    $\widetilde{S}_{\mathrm{new}} \leftarrow \widetilde{S}$
else
    $r \leftarrow 1$
    while $|\lambda_{\pi(2r)}| > \epsilon_S$ or $|\lambda_{\pi(2r)}| > (1 - 1/\log n)|\lambda_{\pi(r)}|$ **do**
        $r \leftarrow r + 1$
    end while
    $(\lambda_{\mathrm{new}})_{\pi(i)} \leftarrow \begin{cases} 0 & \text{if } i = 1, 2, \cdots, 2r; \\ \lambda_{\pi(i)} & \text{otherwise.} \end{cases}$
    $\widetilde{S}_{\mathrm{new}} \leftarrow \widetilde{S} + S_{\mathrm{new}}^{1/2} \cdot U \cdot \mathbf{diag}(\lambda_{\mathrm{new}} - \lambda) \cdot U^\top \cdot S_{\mathrm{new}}^{1/2}$
end if
return $\widetilde{S}_{\mathrm{new}}$

following *rank inequality* (see Theorem B.5.1 for the formal statement).

Theorem 3.2.1 (Rank inequality, informal version)**.** *Let $\widetilde{S}_1, \widetilde{S}_2, \cdots, \widetilde{S}_T \in \mathbb{R}^{n\times n}$ denote the sequence of approximate slack matrices generated in our interior point method. For each $t \in [T-1]$, denote by $r_t = \mathrm{rank}(\widetilde{S}_{t+1} - \widetilde{S}_t)$ the rank of the update on $\widetilde{S}_t$. Then, the sequence $r_1, r_2, \cdots, r_T$ satisfies*

$$\sum_{t=1}^{T} \sqrt{r_t} \leq \widetilde{O}(T).$$

The key component to proving Theorem 3.2.1 is our novel potential function $\Phi : \mathbb{R}^{n\times n} \to \mathbb{R}_{\geq 0}$

$$\Phi(Z) := \sum_{\ell=1}^{n} \frac{|\lambda(Z)|_{[\ell]}}{\sqrt{\ell}},$$

where $|\lambda(Z)|_{[\ell]}$ is the ℓ-th in the list of eigenvalues of $Z \in \mathbb{R}^{n\times n}$ sorted in decreasing order of their absolute values. We show an upper bound on the increase in this potential when S is updated, a lower bound on its decrease when $\widetilde{S}$ is updated, and combine the two with non-negativity of the potential to obtain Theorem 3.2.1.

Specifically, first we prove that whenever S is updated in an iteration, the potential function increases by *at most* $\widetilde{O}(1)$ (see Lemma B.5.2). The proof of this statement crucially uses the structural property of interior point method that slack matrices in consecutive steps are sufficiently close to each other. Formally, for any iteration $t \in [T]$, we show in Theorem B.4.1 that the consecutive slack matrices S_t and S_{t+1} satisfy

$$\|S_t^{-1/2} S_{t+1} S_t^{-1/2} - I\|_F = O(1). \tag{3.2.8}$$

We then combine this bound with Fact B.1.2, which relates the ℓ_2 distance between the spectrum of two matrices with the Frobenius norm of their difference. Next, when $\widetilde{S}$ is updated, we prove that our method of zeroing out the r_t largest eigenvalues of $|Z_t|$, thereby incurring a rank-r_t update to $\widetilde{S}_t$, decreases the potential by *at least* $\widetilde{O}(\sqrt{r_t})$ (see Lemma B.5.3).

Maintaining rectangular matrix multiplication for Hessian computation. Given the low-rank update on $\widetilde{S}$ described above, we show how to efficiently update the *approximate Hessian* $\widetilde{H}$ defined as

$$\widetilde{H}_{j,k} = \mathrm{tr}\left[\widetilde{S}^{-1} A_j \widetilde{S}^{-1} A_k\right] \tag{3.2.9}$$

for each entry $(j,k) \in [m] \times [m]$. The approximate slack matrix $\widetilde{S}$ being a spectral approximation of the true slack matrix S implies that the approximate Hessian $\widetilde{H}$ is also a spectral approximation of the true Hessian H (see Lemma B.4.3). This approximate Hessian therefore suffices for our algorithm to approximately follow the central path.

To efficiently update the approximate Hessian $\widetilde{H}$ in Equation (3.2.9), we notice that a rank-r update on $\widetilde{S}$ implies a rank-r update on $\widetilde{S}^{-1}$ via the Woodbury matrix identity (see Fact B.1.4). The change in $\widetilde{S}^{-1}$ can be expressed as

$$\Delta(\widetilde{S}^{-1}) = V_+ V_+^\top - V_- V_-^\top, \tag{3.2.10}$$

where $V_+, V_- \in \mathbb{R}^{n\times r}$. Plugging Equation (3.2.10) into Equation (3.2.9), we can express $\Delta\widetilde{H}_{j,k}$ as the sum of multiple terms, among the costliest of which are those of the form $\mathrm{tr}\left[\widetilde{S}^{-1} A_j V V^\top A_k\right]$, where $V \in \mathbb{R}^{n\times r}$ is either V_+ or V_-. We compute $\mathrm{tr}\left[\widetilde{S}^{-1} A_j V V^\top A_k\right]$ for all $(j,k) \in [m] \times [m]$ in time $\mathcal{T}_{\mathrm{mat}}(r, n, mn)$ by first computing $V^\top A_k$ for all $k \in [m]$ by horizontally concatenating all A_k's into a wide matrix of size $n \times mn$. We then compute the product of $\widetilde{S}^{-1/2}$ with $A_j V$ for all $j \in [m]$, which can be done in time $\mathcal{T}_{\mathrm{mat}}(n, n, mr)$, which equals $\mathcal{T}_{\mathrm{mat}}(n, mr, n)$ (see Lemma B.2.3). Finally, by flattening each $\widetilde{S}^{-1/2} A_j V$ into a vector of length nr and stacking all these vectors to form a matrix $\widetilde{\mathcal{B}} \in \mathbb{R}^{m\times nr}$ with j-th row $\widetilde{\mathcal{B}}_j = \mathsf{vec}(\widetilde{S}^{-1/2} A_j V)$, the task of computing $\mathrm{tr}\left[\widetilde{S}^{-1} A_j V V^\top A_k\right]$ for all $(j,k) \in [m] \times [m]$

reduces to computing $\widetilde{\mathcal{B}}\widetilde{\mathcal{B}}^\top$, which costs $\mathcal{T}_{\text{mat}}(m, nr, m)$.

Putting it all together. In this way, we reduce the runtime of $T \cdot (\mathcal{T}_{\text{mat}}(n, mn, n) + \mathcal{T}_{\text{mat}}(m, n^2, m))$ for computing the Hessian using fast rectangular matrix multiplication down to

$$\sum_{t=1}^{T} \left(\mathcal{T}_{\text{mat}}(r_t, n, mn) + \mathcal{T}_{\text{mat}}(n, mr_t, n) + \mathcal{T}_{\text{mat}}(m, nr_t, m)\right), \tag{3.2.11}$$

where r_t is the rank of the update on $\widetilde{S}_t$. Applying Theorem 3.2.1 with several properties of fast rectangular matrix multiplication that we prove in Section B.2 , we upper bound the runtime in (3.2.11) by

$$O^*(\sqrt{n}(mn^2 + m^\omega + n^\omega)),$$

which implies Theorem 3.1.2. In Section 3.3, we discuss bottlenecks to further improving our runtime.

3.3 Bottlenecks to Improving Our Result

In most cases, the costliest term in our runtime is the per iteration cost of mn^2, which corresponds to reading the entire input in each iteration. Our subsequent discussions therefore focus on the steps in our algorithm that require at least mn^2 time per iteration.

Slack matrix computation. When y is updated in each iteration of our interior point method, we need to compute the true slack matrix $S = \sum_{i\in[m]} y_i A_i - C$. Computing S is needed to update the approximate slack matrix $\widetilde{S}$ so that $\widetilde{S}$ remains a spectral approximation to S. As S might suffer from full-rank changes, it naturally requires mn^2 time to compute in each iteration.

Gradient computation. Recall from (3.2.2) that our interior point method follows the central path defined via the penalized objective function $\text{minimize}_{y\in\mathbb{R}^m} f_\eta(y)$ where $f_\eta(y) := \eta b^\top y + \phi(y)$ for a parameter $\eta > 0$ and $\phi(y) = -\log\det S$. In each iteration, to perform the Newton step, the gradient of the penalized objective is computed, for each coordinate $j \in [m]$, as

$$g_\eta(y)_j = \eta \cdot b_j - \text{tr}\left[S^{-1}A_j\right]. \tag{3.3.1}$$

Even if we are given S^{-1}, it still requires mn^2 time to compute Equation (3.3.1) for all $j \in [m]$.

Approximate Hessian computation. Recall from Section 3.2.1 that updating the approximate slack matrix S by rank r means the time needed to update the approximate Hessian is dominated by computing the term $\Delta_{j,k} = \text{tr}\left[\widetilde{S}^{-1/2}A_j V \cdot V^\top A_k \widetilde{S}^{-1/2}\right]$, where $V \in \mathbb{R}^{n\times r}$ is a tall, skinny matrix that comes from the spectral decomposition of $\Delta\widetilde{S}^{-1}$. Computing $\Delta_{j,k}$ for all $(j,k) \in [m]\times[m]$ requires reading at least A_j for all $j \in [m]$, which takes time mn^2.

3.3.1 LP Techniques Unlikely to Improve the SDP Runtime

The preceding discussion suggests that reading the entire input in each iteration, which costs mn^2, stands as a natural barrier to further improving the runtime of SDP solvers based on interior point methods. Recent results [CLS19, BLSS20] from the context of linear programs (LPs) suggest two potential techniques to bypass this cost: (1) showing that the Hessian (projection matrix) admits low-rank updates, and (2) speeding computation of the Hessian via sampling. We now argue that

these techniques are unlikely to improve our runtimes for SDPs and, therefore, believe that any improvements in the runtimes of general-purpose SDPs must require completely new ideas.

Showing that the Hessian admits low-rank updates. We saw in Section 3.2.1 that constructing an approximate slack matrix $\widetilde{S}$ that admits low-rank updates in each iterations leveraged the fact that the true slack matrix S changes *slowly* throughout our interior point method as described in Equation (3.2.8). One natural question that follows is whether a similar upper bound can be obtained for the Hessian. If such a result could be proved, one could maintain an approximate Hessian that admitted low-rank updates, which would speed up the approximate Hessian computation. Indeed, in the context of LPs, such a bound for the Hessian can be proved (e.g., [BLSS20, Lemma 47]).

Unfortunately, it is impossible to prove such a statement for the Hessian in the context of SDPs. To show this, it is convenient to express the Hessian using the Kronecker product (Section B.1) as

$$H = \mathcal{A}^\top \cdot (S^{-1} \otimes S^{-1}) \cdot \mathcal{A},$$

where $\mathcal{A} \in \mathbb{R}^{n^2 \times m}$ is the $n^2 \times m$ matrix whose ith column is obtained by flattening A_i into a vector of length n^2. By proper scaling, we can assume without loss of generality that the current slack matrix is $S = I$, and the slack matrix in the next iteration is $S_{\text{new}} = I + \Delta S$, which satisfies $\|\Delta S\|_F = c$ for some tiny constant $c > 0$. For the simple example of $\mathcal{A} = I$ (we are assuming here that $m = n^2$ so that $\mathcal{A}$ is a square matrix), the change in the Hessian can be approximately computed as

$$\begin{aligned}
\|H^{-1/2}\Delta H H^{-1/2}\|_F^2 &\approx \operatorname{tr}\left[((I - \Delta S) \otimes (I - \Delta S) - I \otimes I)^2\right] \\
&\approx \operatorname{tr}\left[(I \otimes \Delta S + \Delta S \otimes I)^2\right] \\
&\geq 2 \cdot \operatorname{tr}\left[I^2\right] \cdot \operatorname{tr}\left[(\Delta S)^2\right] = 2n\|\Delta S\|_F^2 \gg 1.
\end{aligned}$$

This large change indicates that we are unlikely to obtain an approximation to the Hessian that admits low-rank updates, which is a key difference between LPs and SDPs.

Sampling for faster Hessian computation. Recall from Equation (3.2.7) that the Hessian can be computed as $H = \mathcal{B} \cdot \mathcal{B}^\top$, where the jth row of $\mathcal{B} \in \mathbb{R}^{m \times n^2}$ is $\mathcal{B}_j = \mathsf{vec}(S^{-1/2} A_j S^{-1/2})$ for all $j \in [m]$. We might attempt to approximately compute H faster by sampling a subset of columns of $\mathcal{B}$ indexed by $L \subseteq [n^2]$ and compute the product for only the sampled columns. Indeed, sampling techniques have been successfully used to obtain faster LP solvers [CLS19, BLSS20].

For SDPs, however, sampling is unlikely to speed up the Hessian computation. In general, we must sample at least m columns (i.e. $|L| \geq m$) of $\mathcal{B}$ to spectrally approximate H or the computed matrix will not be full rank. However, this requires computing the entries of $S^{-1/2} A_j S^{-1/2}$ that correspond to $L \subseteq [n^2]$ for all $j \in [m]$, which requires reading all A_j's and thus still takes $O(mn^2)$ time.

Chapter 4

Decomposable Non-Smooth Convex Optimization with Nearly-Linear Gradient Oracle Complexity

Many fundamental problems in machine learning can be formulated by the convex program

$$\min_{\theta \in \mathbb{R}^d} \sum_{i=1}^{n} f_i(\theta),$$

where each f_i is a convex, Lipschitz function supported on a subset of d_i coordinates of θ. One common approach to this problem, exemplified by stochastic gradient descent, involves sampling one f_i at every iteration. This approach crucially relies on a notion of uniformity across the f_i's, formally captured by their condition number. In this chapter, we present an algorithm that minimizes the above convex formulation to ϵ-accuracy in $\widetilde{O}(\sum_{i=1}^{n} d_i \log(1/\epsilon))$ gradient computations, with no assumptions on the condition number. The previous best algorithm independent of the condition number is the standard cutting plane method, which requires $O(nd \log(1/\epsilon))$ gradient computations. As a corollary, we improve upon the evaluation oracle complexity for decomposable submodular minimization by Axiotis, Karczmarz, Mukherjee, Sankowski and Vladu (ICML 2021). Our main technical contribution is an adaptive procedure to select an f_i term at every iteration via a novel combination of cutting-plane and interior-point methods.

4.1 Introduction

Many fundamental problems in machine learning are abstractly captured by the convex optimization formulation

$$\text{minimize}_{\theta \in \mathbb{R}^d} \quad \sum_{i=1}^{n} f_i(\theta), \tag{4.1.1}$$

where each f_i is a convex, Lipschitz function. For example, in empirical risk minimization, each f_i measures the loss incurred by the i-th data point from the training set. In generalized linear models, each f_i represents a link function applied to a linear predictor evaluated at the i-th data point.

The ubiquity of (4.1.1) in the setting with smooth f_i's has spurred the development of well-known variants of stochastic gradient methods [RM51, BC03, Zha04, Bot12] such as [RSB12, SSZ13b, JZ13a, MZJ13, DBLJ14b, Mai15, AZY16, HL16a, SLRB17a]; almost universally, these algorithms leverage the "sum structure" of (4.1.1) by sampling, in each iteration, one f_i with which to make progress. These theoretical developments have in turn powered tremendous empirical success in machine learning through widely used software packages such as `libSVM` [CL11a].

In many practical applications, (4.1.1) appears with non-smooth f_i's, as well as the additional structure that each f_i depends only on a subset of the problem parameters θ. One notable example

is decomposable submodular function minimization[1] (SFM), which has proven to be expressive in diverse contexts such as determinantal point processes [KT10], MAP inference in computer vision [KLT09, VKR09, FJPZ13], hypergraph cuts [VBK20], and covering functions [SK10]. Another application is found in generalized linear models when the data is high dimensional and sparse. In this setting, f_i depends on a restricted subset of the parameters θ that correspond to the features of the data point with non-zero value. Last but not least, the case with each f_i depending on a small subset of the parameters is also called sparse separable optimization and has applications in sparse SVM and matrix completion [RRWN11].

In this work, we initiate a systematic study of algorithms for (4.1.1) without the smoothness assumption[2]. Motivated by the aforementioned applications, we introduce the additional structure that each f_i depends on a subset of the coordinates of θ. As is standard in the black-box model for studying first-order convex optimization methods, we allow sub-gradient oracle access to each f_i.

Problem 1. Let $f_1, f_2, \ldots, f_n : \mathbb{R}^d \mapsto \mathbb{R}$ be convex, L-Lipschitz, and possibly non-smooth functions, where each f_i depends on d_i coordinates of θ, and is accessible via a (sub-)gradient oracle. Define $m \stackrel{\text{def}}{=} \sum_{i=1}^n d_i$ to be the "total effective dimension" of the problem. Let $\theta^\star \stackrel{\text{def}}{=} \arg\min_{\theta\in\mathbb{R}^d} \sum_{i=1}^n f_i(\theta)$ be a minimizer of (4.1.1), and let $\theta^{(0)}$ be an initial point such that $\|\theta^{(0)} - \theta^\star\|_2 \le R$. We want to compute a vector $\theta \in \mathbb{R}^d$ satisfying

$$\sum_{i=1}^n f_i(\theta) \le \epsilon LR + \sum_{i=1}^n f_i(\theta^\star). \tag{4.1.2}$$

Prior works. To motivate a new algorithm to solve Problem 1, we first study how some well-known algorithms (presented in Table 4.1) for the related problem (4.1.1) perform on Problem 1, with detailed explanations in Section 4.1.2. We focus on the weakly-polynomial regime and therefore restrict ourselves to algorithms with polylog$(1/\epsilon)$ gradient oracle complexities. Table 4.1 summarizes the performance of all well-known algorithms applied to Problem 1. Note that the variants of gradient descent each require bounded condition number. The results of [Nes83b, AZ17] and cutting plane methods are all complemented by matching lower bounds [WS16, Nes04]. Additionally, the work on non-smooth ERM crucially requires the objective function to be a sum of a smooth ERM part and a non-smooth regularizer. There are many important problems for which the objective function cannot be split in this way, so for these problems, the techniques developed for non-smooth ERM do not apply. In comparison, our work can be understood as dealing with a more general ERM problem with fewer structural assumptions.

Even with smooth f_i's, first-order methods perform poorly when the condition number is large, or when there is a long chain of variable dependencies. These instances commonly arise in applications; an example from signal processing is

$$\text{minimize}_{\mathbf{x}} \left\{ (\mathbf{x}_1 - 1)^2 + \sum_{i=2}^{n-1} (\mathbf{x}_i - \mathbf{x}_{i+1})^2 + \mathbf{x}_n^2 \right\}, \tag{4.1.3}$$

whose variables form an $O(n)$-length chain of dependencies, and has condition number $\kappa = \Theta(n^2)$ and $\bar{\kappa} = \Theta(n^3)$. Gradient descent algorithms such as [Nes83b] and [AZ17] therefore require $\Omega(n^2)$

[1] In decomposable submodular minimization, each f_i corresponds to the Lovász extension of the individual submodular function and is therefore generally non-smooth.

[2] A function f is said to be β-*smooth* if $f(y) \le f(x) + \langle \nabla f(x), y - x\rangle + \beta/2\|y - x\|_2^2$ for all x, y and α-*strongly-convex* if $f(y) \ge f(x) + \langle \nabla f(x), y - x\rangle + \alpha/2\|y - x\|_2^2$ for all x, y. The condition number of f is defined to be $\kappa = \beta/\alpha$.

Table 4.1: Gradient oracle complexities for solving (4.1.1) to ϵ-additive accuracy. κ denotes the condition number of $\sum_i f_i$, and $\overline{\kappa}$ is a variant of the condition number defined to be the sum of smoothness of the f_i's divided by the strong convexity of $\sum_i f_i$.

Authors	Algorithm Type	Gradient Queries	Non-smooth OK?
[C+47]	Gradient Descent (GD)	$O(n\kappa \log(1/\epsilon))$	
[Nes83b]	Accelerated (Acc.) GD	$O(n\sqrt{\kappa}\log(1/\epsilon))$	
[RSB12, SSZ13b, JZ13a]	Stochastic (Stoch.) Variance-Reduced GD	$O((n+\overline{\kappa})\log(1/\epsilon))$	
[SSZ13a, LMH15, FGKS15, ZL15, AB15]	Acc. Stoch. Variance-Reduced GD	$O((n+\sqrt{n\overline{\kappa}})\log(\overline{\kappa})\log(1/\epsilon))$	
[AZ17]	Acc. Stoch. Variance-Reduced GD	$O((n+\sqrt{n\overline{\kappa}})\log(1/\epsilon))$	
[KTE88, NN89, Vai89a, BV02, LSW15]	Cutting-Plane Method (CPM)	$O(nd\log(1/\epsilon))$	✓
[LV21, DLY21]	Robust Interior-Point Method (IPM)	$O(\sum_{i=1}^n d_i^{3.5}\log(1/\epsilon))$	✓

gradient queries, despite the problem's total effective dimension being only $O(n)$.

On the other hand, cutting-plane methods (CPM) and robust interior-point methods (IPM) both trade off the dependency on condition number for worse dependencies on the problem dimension.

These significant gaps in the existing body of work motivate the following question:

Can we solve Problem 1 using a nearly-linear (in total effective dimension) number of sub-gradient oracle queries?

In this chapter, we give an affirmative answer to this question.

4.1.1 Our Results

We present an algorithm to solve Problem 1 with gradient oracle complexity nearly-linear in the total effective dimension. Our main result is the following theorem.

Theorem 4.1.1 (Main Result). *Consider Problem 1 and $\theta^{(0)}$ such that $\|\theta^\star - \theta^{(0)}\|_2 \le R$. Assuming all the f_i's are L-Lipschitz, then there is an algorithm that outputs a vector $\theta \in \mathbb{R}^d$ such that $\sum_{i=1}^n f_i(\theta) \le \sum_{i=1}^n f_i(\theta^\star) + \epsilon \cdot LR$, using $O(m\log(m/\epsilon))$ gradient oracle calls, in time $\mathrm{poly}(m\log(1/\epsilon))$.*

Intuitively, the number of gradient queries for each f_i should be thought of as $\widetilde{O}(d_i)$ in our algorithm, which nearly matches that of the standard cutting-plane method for minimizing the individual function f_i. The nearly-linear dependence on m overall is obtained by leveraging the additional structure on the f_i's and stands in stark contrast to the $O(nd)$ query complexity of CPM, which is significantly worse in the case where each $d_i \ll d$. Furthermore, we improve over the current best gradient descent algorithms in the case where the f_i's have a large condition number.

Based on the query complexity of the standard cutting-plane method, we have the following lower bound matching our algorithm's query complexity up to a $\log m$-factor:

Theorem 4.1.2. *There exist functions $f_1, \ldots, f_n : \mathbb{R}^d \mapsto \mathbb{R}$ for which a total of $\Omega(m\log(1/\epsilon))$ gradient queries are required to solve Problem 1.*

An immediate application of Theorem 4.1.1 is to decomposable submodular function minimization:

Theorem 4.1.3 (Decomposable SFM). *Let $V = \{1, 2, \ldots, n\}$, and $F : 2^V \mapsto [-1, 1]$ be given by $F(S) = \sum_{i=1}^n F_i(S \cap V_i)$, where each $F_i : 2^{V_i} \mapsto \mathbb{R}$ is a submodular function on $V_i \subseteq V$ with $|V_i| \le k$. We can find an ϵ-additive approximate minimizer of F in $O(nk^2\log(nk/\epsilon))$ evaluation oracle calls.*

Theorem 4.1.3 improves upon the evaluation oracle complexity of $\widetilde{O}(nk^6\log(1/\epsilon))$ given in [AKM+21] when the dimension k of each function F_i is large. For non-decomposable SFM, i.e. $n = 1$ and $|V_1| = k$,

the current best weakly-polynomial time SFM algorithm[3] finds an ϵ-approximate minimizer in time $O(k^2 \log(k/\epsilon))$ [LSW15]. Therefore, our result in Theorem 4.1.3 can be viewed as a generalization of the evaluation oracle complexity for non-decomposable SFM in [LSW15], and the dependence on k in Theorem 4.1.3 might be the best possible. We defer the details of decomposable SFM to Section C.1.

Limitations

Some limitations of our algorithm are as follows: When each f_i depends on the entire d-dimensional vector θ, as opposed to a subset of the coordinates of size $d_i \ll d$, our gradient complexity simply matches that of CPM. We would like to highlight, though, that our focus is in fact the regime $d_i \ll d$. When the f_i's are strongly-convex and smooth, our gradient complexity improves over Table 4.1 only when κ is large compared to d_i. Finally, note that we consider only the gradient oracle complexity in our work; our algorithm's implementation requires sampling a Hessian matrix and a gradient vector at every iteration, which incur an additional $\mathrm{poly}(m, \log(1/\epsilon))$ factor in the overall running time.

4.1.2 Technical Challenges in Prior Works

We now describe the key technical challenges that barred existing algorithms from solving Problem 1 in the desired nearly-linear gradient complexity.

Gradient descent and variants. As mentioned in Section 4.1, the family of gradient descent algorithms presented in Table 4.1 are not applicable to Problem 1 without the smoothness assumption. When the objective in Problem 1 is smooth but has a large condition number, even the optimal deterministic algorithm, Accelerated Gradient Descent (AGD) [Nes83b] can perform poorly. For example, when applied to (4.1.3), AGD updates only one coordinate in each step (thereby requiring n steps), with each step performing n gradient queries (one on each term in the problem objective), yielding a total gradient complexity of $\Omega(n^2)$ [Nes83b]. For a similar reason, the fastest randomized algorithm, Katyusha [AZ17] also incurs a gradient complexity of $\Omega(n^2)$ [WS16].

Cutting-plane methods (CPM). Given a convex function f with its set $\mathcal{S}$ of minimizers, CPM minimizes f by maintaining a convex search set $\mathcal{E}^{(k)} \supseteq \mathcal{S}$ in the k^{th} iteration, and iteratively shrinking $\mathcal{E}^{(k)}$ using the sub-gradients of f. Specifically, this is achieved by noting that for any $\mathbf{x}^{(k)}$ chosen from $\mathcal{E}^{(k)}$, if the gradient oracle indicates $\nabla f(\mathbf{x}^{(k)}) \neq 0$, (i.e. $\mathbf{x}^{(k)} \notin \mathcal{S}$), then the convexity of f guarantees $\mathcal{S} \subseteq \mathcal{H}^{(k)} \stackrel{\text{def}}{=} \left\{\mathbf{y} : \langle \nabla f(\mathbf{x}^{(k)}), \mathbf{y} - \mathbf{x}^{(k)} \rangle \leq 0\right\}$, and hence $\mathcal{S} \subseteq \mathcal{H}^{(k)} \cap \mathcal{E}^{(k)}$. The algorithm continues by choosing $\mathcal{E}^{(k+1)} \supseteq \mathcal{E}^{(k)} \cap \mathcal{H}^{(k)}$, and different choices of $\mathbf{x}^{(k)}$ and $\mathcal{E}^{(k)}$ yield different rates of shrinkage of $\mathcal{E}^{(k)}$ until a point in $\mathcal{S}$ is found.

Solving Problem 1 via the current fastest CPM [LSW15] takes $\widetilde{O}(d)$ iterations, each invoking the gradient oracle on every f_i to compute $\nabla f(\mathbf{x}^{(k)}) = \sum_{i=1}^{n} \nabla f_i(\mathbf{x}^{(k)})$. This results in $\widetilde{O}(nd)$ gradient queries overall, which can be quadratic in n when $d = \Theta(n)$ even if each f_i depends on only $d_i = O(1)$ coordinates. Similar to gradient descent and its variants, the poor performance of CPM on Problem 1 may therefore be attributed to their inability to query the right f_i required to make progress.

Interior-point methods (IPM). IPM solves the convex program $\min_{\mathbf{u} \in \mathcal{S}} \langle \mathbf{c}, \mathbf{u} \rangle$ by solving a sequence of unconstrained problems $\min_{\mathbf{u}} \Psi_t(\mathbf{u}) \stackrel{\text{def}}{=} \{t \cdot \langle \mathbf{c}, \mathbf{u} \rangle + \psi_{\mathcal{S}}(\mathbf{u})\}$ parametrized by increasing t, where $\psi_{\mathcal{S}}$ is a *self-concordant barrier* function that enforces feasibility by becoming unbounded as it approaches the boundary of the feasible set $\mathcal{S}$. The algorithm starts at $t = 0$, for which an approximate

[3] Here, we focus on the weakly-polynomial regime, where the runtime dependence on ϵ is $\log(1/\epsilon)$.

minimizer $\mathbf{x}_0^\star$ of $\psi_{\mathcal{S}}$ is known, and alternates between increasing t and updating to an approximate minimizer $\mathbf{x}_t^\star$ of the new Ψ_t via Newton's method. For a sufficiently large t, the minimizer $\mathbf{x}_t^\star$ also approximately optimizes the original problem $\min_{\mathbf{u}\in\mathcal{S}}\langle \mathbf{c}, \mathbf{u}\rangle$ with sub-optimality gap $O(\nu/t)$, where ν is the self-concordance parameter of the barrier function used.

To apply IPM to Problem 1, we may first transform (4.1.1) to $\min_{(\mathbf{u},\mathbf{z})\in\mathcal{K}} \sum_i \mathbf{z}_i$, where $\mathcal{K} = \{(\mathbf{u}, \mathbf{z}) : (\mathbf{u}_i, \mathbf{z}_i) \in \mathcal{K}_i, \forall i \in [n]\}$ and $\mathcal{K}_i = \{(\mathbf{u}_i, \mathbf{z}_i) : f_i(\mathbf{u}_i) \le \mathbf{z}_i\}$ is the feasible set. Using the universal barrier ψ_i for each $\mathcal{K}_i$ [NN94], the number of iterations of IPM is $\widetilde{O}(\sqrt{\sum_{i=1}^n d_i})$, each requiring the computation of the Hessian and gradient of ψ_i for all $i \in [n]$, leading to a total of $\widetilde{O}(n^{1.5})$ sub-gradient queries to f_i's even when all $d_i = O(1)$. Even when leveraging the recent framework of robust IPM for linear programs [LV21], the computation of each Hessian (by sampling the corresponding $\mathcal{K}_i$ [JLLV21]) yields a total sub-gradient oracle complexity of $\widetilde{O}(\sum_{i=1}^n d_i^{3.5})$, far from the complexity we seek.

4.1.3 Our Algorithmic Framework

We now give an overview of the techniques developed in this work to overcome the above barriers. First, by making identical copies of coordinates shared by different f_i and using the convex sets $\mathcal{K}_i$ to make the objective function linear, we transform (4.1.1) into a convex program over structured convex sets:

$$\begin{array}{ll} \text{minimize} & \langle \mathbf{c}, \mathbf{x}\rangle, \\ \text{subject to} & \mathbf{x}_i \in \mathcal{K}_i \subseteq \mathbb{R}^{d_i+1} \ \forall i \in [n] \\ & \mathbf{A}\mathbf{x} = \mathbf{b}. \end{array} \tag{4.1.4}$$

where $\mathbf{x}$ is the concatenation of the vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$, the objective vector $\mathbf{c}$ is 1 for the last coordinate of each $\mathbf{x}_i$ and 0 otherwise, and $\mathbf{A}\mathbf{x} = \mathbf{b}$ enforces that different copies of the same coordinates should be the same. Note that the sub-gradient oracle for f_i can be transformed equivalently to a separation oracle $\mathcal{K}_i$. We define $\mathcal{K} \stackrel{\text{def}}{=} \mathcal{K}_1 \times \mathcal{K}_2 \times \ldots \times \mathcal{K}_n$.

Main idea: combining CPM and IPM. Recall that CPM maintains a convex set which initially contains the feasible region and gradually shrinks around the minimizer, while IPM maintains a point inside the feasible region that moves toward the minimizer. Our novel idea is to combine both methods and maintain *an inner convex set $\mathcal{K}_{in,i}$ as well as an outer convex set $\mathcal{K}_{\text{out},i}$* for each $i \in [n]$, such that $\mathcal{K}_{\text{in},i} \subseteq \mathcal{K}_i \subseteq \mathcal{K}_{\text{out},i}$. We define $\mathcal{K}_{\text{in}}$ and $\mathcal{K}_{\text{out}}$ analogously to $\mathcal{K}$. When Inequality 4.3.4 and Inequality 4.3.3 are satisfied for all $i \in [n]$, we make IPM-style updates without needing to make any oracle calls. When Inequality 4.3.3 is violated for some $i \in [n]$, we query the separation oracle at the point $\mathbf{x}_{\text{out},i}^\star$ defined as the centroid of $\mathcal{K}_{\text{out},i}$ (c.f. Proposition 9). Based on the oracle's response, we iteratively either grow $\mathcal{K}_{\text{in},i}$ (and, thus, $\mathcal{K}_{\text{in}}$) outward or shrink $\mathcal{K}_{\text{out},i}$ (and, thus, $\mathcal{K}_{\text{out}}$) inward, until ultimately they approximate $\mathcal{K}$ around the optimum point.

First benefit: large change in volume. If the point $\mathbf{x}_{\text{out},i}^\star$ violates Inequality 4.3.3 for some $i \in [n]$, we query the separation oracle to see if $\mathbf{x}_{\text{out},i}^\star \in \mathcal{K}_i$ or not. If $\mathbf{x}_{\text{out},i}^\star \in \mathcal{K}_i$, then it is used to expand $\mathcal{K}_{\text{in},i}$, yielding in a large volume increase for $\mathcal{K}_{\text{in},i}$. On the other hand, if $\mathbf{x}_{\text{out},i}^\star \notin \mathcal{K}_i$, the fact that it is the centroid of $\mathcal{K}_{\text{out},i}$ results in a large volume decrease for $\mathcal{K}_{\text{out},i}$ when it is intersected with a halfspace through $\mathbf{x}_{\text{out},i}^\star$. Thus, our algorithm witnesses a large change in volume of one of $\mathcal{K}_{\text{in},i}$ and $\mathcal{K}_{\text{out},i}$, regardless of the answer from the oracle. Just like in standard CPM, this rapid change in volume is crucial to achieving the algorithm's oracle complexity.

Second benefit: making a smart choice about querying f_i. Since the algorithm maintains both an inner and outer set approximating $\mathcal{K}$, by checking if $\mathcal{K}_{\text{in},i}$ and $\mathcal{K}_{\text{out},i}$ differ significantly

(Inequality 4.3.3 essentially performs this function), we can determine if $\mathcal{K}_i$ is poorly approximated, and if so, improve the inner and outer approximations of the true feasible set. Choosing the right $\mathcal{K}_i$ translates to choosing the right f_i to make progress with at an iteration; thus, we address the central weakness of the gradient descent variants in solving (4.1.1).

4.2 Notation and Preliminaries

We lay out the notation used in this chapter as well as the definitions and prior known results that we rely on. We use lowercase boldface letters to denote (column) vectors and uppercase boldface letters to denote matrices. We use $\mathbf{x}_i$ to denote the i^{th} block of coordinates in the vector $\mathbf{x}$ (the ordering of these blocks is not important in our setup). We use $\succeq$ and $\preceq$ to denote the Loewner ordering of matrices.

We use $\langle \mathbf{x}, \mathbf{y} \rangle$ to mean the Euclidean inner product $\mathbf{x}^\top \mathbf{y}$. A subscript $\mathbf{x}$ in the inner product notation means it is induced by the Hessian of some function (which is clear from context) at $\mathbf{x}$; for example, $\langle \mathbf{u}, \mathbf{v} \rangle_{\mathbf{x}} = \mathbf{u}^\top \nabla^2_{ii} \psi(\mathbf{x}) \mathbf{v}$ with ψ inferred from context. We define the local norm of $\mathbf{v}$ at $\mathbf{x}$ analogously: $\|\mathbf{v}\|_{\mathbf{x}} = \sqrt{\langle \mathbf{v}, \nabla^2 \psi(\mathbf{x}) \cdot \mathbf{v} \rangle}$. We also define the norm $\|\mathbf{v}\|_{\mathbf{x},1} \stackrel{\text{def}}{=} \sum_{i=1}^n \|\mathbf{v}\|_{\mathbf{x}_i}$.

We use ψ to represent barrier functions and Φ to represent potential functions, with appropriate subscripts and superscripts to qualify them as needed.

4.2.1 Facts from Convex Analysis

In this section, we present some definitions and properties from convex analysis that are useful in our chapter. These results are standard and may be found in, for example, [Roc70, BBV04].

Definition 2. Let $f : \mathbb{R}^n \to \mathbb{R}$. Then the function $f^* : \mathbb{R}^n \to \mathbb{R}$ defined as

$$f^*(\mathbf{y}) = \sup_{\mathbf{x} \in \text{dom}(f)} [\langle \mathbf{x}, \mathbf{y} \rangle - f(\mathbf{x})]$$

is called the Fenchel conjugate of the function f. An immediate consequence of the definition (and by applying the appropriate convexity-preserving property) is that f^* is convex, regardless of the convexity of f. We use the superscript $*$ on functions to denote their conjugates.

Lemma 4.2.1 (Biconjugacy [Roc70])**.** *For a closed, convex function f, we have $f = f^{**}$.*

Lemma 4.2.2 ([Roc70])**.** *For a closed, convex differentiable function f, we have $\mathbf{y} = \nabla f(\mathbf{x}) \iff \mathbf{x} = \nabla f^*(\mathbf{y})$.*

Proof. This is a fact proved in standard texts in convex analysis mentioned at the start of this section; however, we present a brief proof for completeness. By definition of the gradient and Definition 2, we have the following equation (Fenchel-Young equality)

$$f(\mathbf{x}) + f^*(\mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle$$

if and only if $\mathbf{y} = \nabla f(\mathbf{x})$. Since f is a closed, convex function, we have, by Lemma 4.2.1, that $f = f^{**}$. Applying this fact to the preceding equation then gives

$$f^*(\mathbf{y}) + f^{**}(\mathbf{x}) = \langle \mathbf{x}, \mathbf{y} \rangle,$$

which implies $\mathbf{x} = \nabla f^*(\mathbf{y})$. □

Lemma 4.2.3 ([Roc70])**.** *For a strictly convex, twice-differentiable function f, we have $\nabla^2 f^*(\nabla f(\mathbf{x})) = (\nabla^2 f(\mathbf{x}))^{-1}$.*

Proof. From Lemma 4.2.2, we have $\nabla f^*(\nabla f(\mathbf{x})) = \mathbf{x}$. Differentiating both sides and using the chain rule of calculus, we obtain

$$\nabla^2 f^*(\nabla f(\mathbf{x})) \cdot \nabla^2 f(\mathbf{x}) = \mathbf{I}.$$

Since $\nabla^2 f(\mathbf{x}) > 0$,we may then infer the claimed equation. □

Definition 3 (Polar of a Set [RW09])**.** Given a set $\mathcal{S} \subseteq \mathbb{R}^n$, its polar is defined as

$$\mathcal{S}^\circ \stackrel{\text{def}}{=} \{\mathbf{y} \in \mathbb{R}^n : \langle \mathbf{y}, \mathbf{x} \rangle \leq 1,\ \forall \mathbf{x} \in \mathcal{S}\}.$$

Lemma 4.2.4 ([Roc70])**.** *Let $\mathcal{S} \subseteq \mathbb{R}^n$ be a closed, compact, convex set, and let $\mathbf{y}$ be a point. Then $(\text{conv}\,\{\mathcal{S}, \mathbf{y}\})^\circ \subseteq \mathcal{S}^\circ \cap \mathcal{H}$, where $\mathcal{H}$ is the halfspace defined by $\mathcal{H} = \{\mathbf{z} \in \mathbb{R}^n : \langle \mathbf{z}, \mathbf{y} \rangle \leq 1\}$.*

4.2.2 Background on Interior-Point Methods

Our work draws heavily upon geometric properties of self-concordant functions, which underpin the rich theory of interior-point methods. We list below the formal results needed for our analysis, and refer the reader to [NN94, Ren01a] for a detailed exposition of this function class. We begin with the definitions of self-concordant functions and self-concordant barriers:

Definition 4 (Self-concordance [NN94])**.** A function $F : Q \mapsto \mathbb{R}$ is a self-concordant function on a convex set Q if for any $\mathbf{x} \in Q$ and any $\mathbf{h}$,

$$|D^3 F(\mathbf{x})[\mathbf{h}, \mathbf{h}, \mathbf{h}]| \leq 2(D^2 F(\mathbf{x})[\mathbf{h}, \mathbf{h}])^{3/2},$$

where $D^k F(\mathbf{x})[\mathbf{h}_1, \ldots, \mathbf{h}_k]$ denotes the k-th derivative of F at $\mathbf{x}$ along the directions $\mathbf{h}_1, \ldots, \mathbf{h}_k$. We say F is a ν-self-concordant barrier if F further satisfies $\nabla F(x)^\top (\nabla^2 F(x))^{-1} \nabla F(x) \leq \nu$ for any $\mathbf{x} \in Q$.

Theorem 4.2.5 (Theorem 2.3.3 from [Ren01a])**.** *If f is a self-concordant barrier, then for all $\mathbf{x}$ and $\mathbf{y} \in dom(f)$, we have $\langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle \leq \nu$, where ν is the self-concordance of f.*

Theorem 4.2.6 (Theorem 2.3.4 from [Ren01a])**.** *If f is a ν-self-concordant barrier such that $\mathbf{x}, \mathbf{y} \in dom(f)$ satisfy $\langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle \geq 0$, then $\mathbf{y} \in \mathcal{B}_{\mathbf{x}}(\mathbf{x}, 4\nu + 1)$.*

We now state the following result from self-concordance calculus.

Theorem 4.2.7 (Theorem 3.3.1 of [Ren01a])**.** *If f is a (strongly nondegenerate) self-concordant function, then so is its Fenchel conjugate f^*.*

The following result gives a bound on the quadratic approximation of a function, with the distance between two points measured in the local norm. The convergence of Newton's method can be essentially explained by this result.

Theorem 4.2.8 (Theorem 2.2.2 of [Ren01a])**.** *If f is a self-concordant function, $\mathbf{x} \in dom(f)$, and $\mathbf{y} \in \mathcal{B}_{\mathbf{x}}(\mathbf{x}, 1)$, then*

$$f(\mathbf{y}) \leq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \frac{1}{2}\|\mathbf{y} - \mathbf{x}\|_{\mathbf{x}}^2 + \frac{\|\mathbf{y} - \mathbf{x}\|_{\mathbf{x}}^3}{3(1 - \|\mathbf{y} - \mathbf{x}\|_{\mathbf{x}})},$$

where $\|\mathbf{y} - \mathbf{x}\|_{\mathbf{x}}^2 \stackrel{def}{=} \langle \mathbf{y} - \mathbf{x}, \nabla^2 f(\mathbf{x}) \cdot (\mathbf{y} - \mathbf{x}) \rangle$.

Finally, we need the following definitions of entropic barrier and universal barrier.

Definition 5 ([BE15, Che21])**.** Given a convex body $\mathcal{K} \subseteq \mathbb{R}^n$ and some fixed $\theta \in \mathbb{R}^n$, define the function $f(\theta) = \log\left[\int_{\mathbf{x}\in\mathcal{K}} \exp\langle \mathbf{x}, \theta\rangle d\mathbf{x}\right]$. Then the Fenchel conjugate $f^* : \text{int}(\mathcal{K}) \to \mathbb{R}$ is a self-concordant barrier termed the *entropic barrier*. The entropic barrier is n-self-concordant.

Definition 6 ([NN94, LY21])**.** Given a convex body $\mathcal{K} \subseteq \mathbb{R}^n$, the *universal barrier* of $\mathcal{K}$ is defined as $\psi : \text{int}(\mathcal{K}) \to \mathbb{R}$ by

$$\psi(\mathbf{x}) = \log \text{vol}((\mathcal{K} - \mathbf{x})^\circ)$$

where $(\mathcal{K} - \mathbf{x})^\circ = \{\mathbf{y} \in \mathbb{R}^n : \mathbf{y}^\top(\mathbf{z} - \mathbf{x}) \leq 1, \forall \mathbf{z} \in \mathcal{K}\}$ is the polar set of $\mathcal{K}$ with respect to $\mathbf{x}$. The universal barrier is n-self-concordant.

4.2.3 Facts from Convex Geometry

Since our analysis is contingent on the change in the volume of convex bodies when points are added to them or when they are intersected with halfspaces, we invoke the classical result by Grünbaum several times. We, therefore, state its relevant variants next: Theorem 4.2.9 applies to log-concave distributions, and Corollary 8 is its specific case, since the indicator function of a convex set is a log-concave function [BBV04].

Theorem 4.2.9 ([Grü60, BKL$^+$20])**.** *Let f be a log-concave distribution on $\mathbb{R}^d$ with centroid $\mathbf{c}_f$. Let $\mathcal{H} = \left\{\mathbf{u} \in \mathbb{R}^d : \langle \mathbf{u}, \mathbf{v}\rangle \geq q\right\}$ be a halfspace defined by a normal vector $\mathbf{v} \in \mathbb{R}^d$. Then, $\int_{\mathcal{H}} f(\mathbf{z})d\mathbf{z} \geq \frac{1}{e} - t^+$, where $t = \frac{q - \langle \mathbf{c}_f, \mathbf{v}\rangle}{\sqrt{\mathbb{E}_{\mathbf{y}\sim f}\langle \mathbf{v}, \mathbf{y} - \mathbf{c}_f\rangle^2}}$ is the distance of the centroid to the halfspace scaled by the standard deviation along the normal vector $\mathbf{v}$ and $t^+ \stackrel{def}{=} \max\{0, t\}$.*

Remark 7. A crucial special case of Theorem 4.2.9 is that cutting a convex set through its centroid yields two parts, the smaller of which has volume at least $1/e$ times the original volume and the larger of which is at most $1 - 1/e$ times the original total volume.

Corollary 8 ([Grü60])**.** *Let $\mathcal{K}$ be a convex set with centroid μ and covariance matrix Σ. Then, for any point $\mathbf{x}$ satisfying $\|\mathbf{x} - \mu\|_{\Sigma^{-1}} \leq \eta$ and a halfspace $\mathcal{H}$ such that $\mathbf{x} \in \mathcal{H}$, we have $\text{vol}(\mathcal{K} \cap \mathcal{H}) \geq \text{vol}(\mathcal{K}) \cdot (1/e - \eta)$.*

Finally, we need the following facts.

Fact 4.2.10 (Volumes of standard objects)**.** *The volume of a q-dimensional Euclidean ball is given by $\text{vol}(\mathcal{B}_q(0, \bar{R})) = \frac{\pi^{q/2}}{\Gamma(1+q/2)}\bar{R}^q$, and the volume of a q-dimensional cone $= \frac{1}{q+1} \cdot$ volume of base $\cdot$ height.*

4.3 Our Algorithm

We begin by reducing Problem 1 to the following slightly stronger formulation (see Theorem 4.1.1 for the detailed reduction):

$$\begin{array}{ll} \text{minimize} & \langle \mathbf{c}, \mathbf{x}\rangle, \\ \text{subject to} & \mathbf{x}_i \in \mathcal{K}_i \subseteq \mathbb{R}^{d_i+1} \ \forall i \in [n] \\ & \mathbf{A}\mathbf{x} = \mathbf{b}. \end{array} \tag{4.3.1}$$

where $\mathbf{x}$ is a concatenation of vectors $\mathbf{x}_i$'s, and the $\mathcal{K}_i$'s are disjoint convex sets. This formulation decouples the overlapping support of the original f_i's by introducing additional variables tied

together through the linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$. Each $\mathcal{K}_i$ is constructed by applying a standard epigraph trick to the function f_i.

Note that we do not have a closed-form expression for $\mathcal{K}_i$. Instead, the subgradient oracle for f_i translates to a *separation oracle* for $\mathcal{K}_i$: on a point $\mathbf{z}_i$ queried by the oracle, the oracle either asserts $\mathbf{z}_i \in \mathcal{K}_i$, or returns a separating hyperplane that separates $\mathbf{z}_i$ from $\mathcal{K}_i$.

At the start of our algorithm, we have the following guarantee:

Lemma 4.3.1. *At the start of our algorithm, we are guaranteed the existence of the following.*

1. *Explicit convex sets* $\mathcal{K}_{in} \stackrel{def}{=} \mathcal{K}_{in,1} \times \mathcal{K}_{in,2} \times \cdots \times \mathcal{K}_{in,n}$ *and* $\mathcal{K}_{out} \stackrel{def}{=} \mathcal{K}_{out,1} \times \mathcal{K}_{out,2} \times \cdots \times \mathcal{K}_{out,n}$ *such that* $\mathcal{K}_{in} \subseteq \mathcal{K} \stackrel{def}{=} \mathcal{K}_1 \times \cdots \times \mathcal{K}_n \subseteq \mathcal{K}_{out}$,

2. *An initial* $\mathbf{x}_{initial} \in \mathcal{K}_{in}$ *such that* $\mathbf{A}\mathbf{x}_{initial} = \mathbf{b}$.

We show how to construct such a set $\mathcal{K}_{\text{in}}$ in Section 4.5.1 and how to find such a $\mathcal{K}_{\text{out}}$ and $\mathbf{x}_{\text{initial}}$ in Section 4.5.2.

4.3.1 Details of Our Algorithm

In this section, we explain our main algorithm (Algorithm 4.3.1).

Algorithm 4.3.1 Minimizing Decomposable Convex Function ▷ Solving Problem 4.3.1

1: **Input.** ϵ, $\mathbf{A}$, $\mathbf{b}$, $\mathbf{c}$, R, r, m , n, $\mathbf{x}$, $\mathcal{K}_{\text{in}}$, $\mathcal{K}_{\text{out}}$, and O_i for each $i \in [n]$.
2: **Initialization.** $t = \frac{m \log m}{\sqrt{n}\|\mathbf{c}\|_2 R}$ and $t_{\text{end}} = \frac{8m}{\epsilon\|\mathbf{c}\|_2 R}$, $\eta = \frac{1}{100}$, and $\mathbf{x}^\star_{\text{out}}$ (via Equation (4.3.2))
3: **while** true **do**
4: **if** $\langle \mathbf{c}, \mathbf{x} \rangle \leq \langle \mathbf{c}, \mathbf{x}^\star_{\text{out}} \rangle + \frac{4m}{t}$ **then** ▷ Either update t or end the algorithm
5: **if** $t \geq t_{\text{end}}$ **then**
6: **return** $\arg\min_{\mathbf{x}:\mathbf{x}\in\mathcal{K}_{\text{in}},\mathbf{A}\mathbf{x}=\mathbf{b}} \left\{ t\langle \mathbf{c}, \mathbf{x} \rangle + \sum_{i=1}^n \psi_{\text{in},i}(\mathbf{x}_i) \right\}$. ▷ End the algorithm
7: **end if**
8: $t \leftarrow t \cdot \left[1 + \frac{\eta}{4m}\right]$ ▷ Update t
9: Update $\mathbf{x}^\star_{\text{out}}$ and jump to Line 3 ▷ $\mathbf{x}^\star_{\text{out}}$ computed as as per Equation (4.3.2)
10: **end if**
11: **for** all $i \in [n]$ **do**
12: **if** $\langle \nabla\psi_{\text{in},i}(\mathbf{x}_i), \mathbf{x}^\star_{\text{out},i} - \mathbf{x}_i \rangle + \eta\|\mathbf{x}^\star_{\text{out},i} - \mathbf{x}_i\|_{\mathbf{x}_i} \geq 4d_i$ **then**
13: **if** $\mathbf{x}^\star_{\text{out},i} \in \mathcal{K}_i$ **then** ▷ Query O_i
14: $\mathcal{K}_{\text{in},i} = \text{conv}(\mathcal{K}_{\text{in},i}, \mathbf{x}^\star_{\text{out},i})$ ▷ Update $\mathcal{K}_{\text{in},i}$
15: **else**
16: $\mathcal{K}_{\text{out},i} = \mathcal{K}_{\text{out},i} \cap \mathcal{H}_i$, where $\mathcal{H}_i = O_i(\mathbf{x}^\star_{\text{out},i})$ ▷ Update $\mathcal{K}_{\text{out},i}$
17: **end if**
18: Update $\mathbf{x}^\star_{\text{out}}$ and jump to Line 3 ▷ $\mathbf{x}^\star_{\text{out}}$ computed as per Equation (4.3.2)
19: **end if**
20: **end for**
21: Set $\delta_{\mathbf{x}} \stackrel{\text{def}}{=} \frac{\eta}{2} \cdot \frac{\mathbf{x}^\star_{\text{out}} - \mathbf{x}}{\|\mathbf{x}^\star_{\text{out}} - \mathbf{x}\|_{\mathbf{x},1}}$,where $\|\mathbf{u}\|_{\mathbf{x},1} \stackrel{\text{def}}{=} \sum_{i=1}^n \|\mathbf{u}\|_{\mathbf{x}_i}$.
22: $\mathbf{x} \leftarrow \mathbf{x} + \delta_{\mathbf{x}}$ ▷ Move $\mathbf{x}$ towards $\mathbf{x}^\star_{\text{out}}$
23: **end while**

The inputs to Algorithm 4.3.1 are: initial sets $\mathcal{K}_{\text{in}}$ and $\mathcal{K}_{\text{out}}$ satisfying $\mathcal{K}_{\text{in}} \subseteq \mathcal{K} \subseteq \mathcal{K}_{\text{out}}$, an initial

point $\mathbf{x} \in \mathcal{K}_{\text{in}}$ satisfying $\mathbf{A}\mathbf{x} = \mathbf{b}$, a separation oracle O_i for each $\mathcal{K}_i$, the objective vector $\mathbf{c}$, and scalar parameters m, n, R, r, and ϵ. All the parameters are set in the proof of Theorem 4.4.10.

Throughout the algorithm, we maintain a central path parameter t for IPM-inspired updates, the current solution $\mathbf{x}$, and convex sets $\mathcal{K}_{\text{in},i}$ and $\mathcal{K}_{\text{out},i}$ satisfying $\mathcal{K}_{\text{in},i} \subseteq \mathcal{K}_i \subseteq \mathcal{K}_{\text{out},i}$ for each $i \in [n]$. To run IPM-style updates, we choose the entropic barrier on $\mathcal{K}_{\text{out}}$ and the universal barrier on $\mathcal{K}_{\text{in}}$.

Given the current set $\mathcal{K}_{\text{out}}$, the current t, and the entropic barrier ψ_{out} defined on $\widehat{\mathcal{K}}_{\text{out}} \stackrel{\text{def}}{=} \mathcal{K}_{\text{out}} \cap \{\mathbf{u} : \mathbf{A}\mathbf{u} = \mathbf{b}\}$, we define the point

$$\mathbf{x}^\star_{\text{out}} \stackrel{\text{def}}{=} \arg\min_{\mathbf{x} \in \widehat{\mathcal{K}}_{\text{out}}} \{t\langle \mathbf{c}, \mathbf{x}\rangle + \psi_{\text{out}}(\mathbf{x})\}. \tag{4.3.2}$$

Per the IPM paradigm, for the current value of t, this point serves as a target to "chase" when optimizing $\langle \mathbf{c}, \mathbf{x}\rangle$ over the set $\widehat{\mathcal{K}}_{\text{out}}$. Although our overall goal in Problem 4.3.1 is to optimize over $\mathcal{K} \cap \{\mathbf{u} : \mathbf{A}\mathbf{u} = \mathbf{b}\}$, we do not have an explicit closed-form expression for $\mathcal{K}$ and therefore must use its known proxies, $\mathcal{K}_{\text{in}}$ or $\mathcal{K}_{\text{out}}$, which we maintain throughout the algorithm; between these two sets, we choose $\mathcal{K}_{\text{out}}$ because $\mathcal{K}_{\text{out}} \supseteq \mathcal{K}$ ensures we do not miss a potential optimal point.

Having computed the current target $\mathbf{x}^\star_{\text{out}}$, we move the current solution $\mathbf{x}$ towards it by taking a Newton step, provided certain conditions of feasibility and minimum progress are satisfied. If either one of these conditions is violated, we update either $\mathcal{K}_{\text{in}}$, $\mathcal{K}_{\text{out}}$, or the parameter t.

Updating x. In order to move $\mathbf{x}$ towards $\mathbf{x}^\star_{\text{out}}$, we require two conditions to hold: $\mathbf{x}^\star_{\text{out}} \in \mathcal{K}_{\text{in}}$ and $\langle \mathbf{c}, \mathbf{x}\rangle \geq \langle \mathbf{c}, \mathbf{x}^\star_{\text{out}}\rangle + O(1/t)$.

The first condition implies $\mathbf{x}^\star_{\text{out}} \in \mathcal{K}$, which would in turn ensure feasibility of the resulting $\mathbf{x}$ after a Newton step. To formally check this condition, we check if the following inequality is satisfied for all $i \in [n]$ and for a fixed constant η:

$$\langle \nabla \psi_{\text{in},i}(\mathbf{x}_i), \mathbf{x}^\star_{\text{out},i} - \mathbf{x}_i\rangle + \eta \cdot \|\mathbf{x}^\star_{\text{out},i} - \mathbf{x}_i\|_{\mathbf{x}_i} \leq 4d_i. \tag{4.3.3}$$

The intuition is that since any point within the domain of a self-concordant barrier satisfies the inequalities in Theorem 4.2.5 and Theorem 4.2.6, violating Inequality 4.3.3 implies that $\mathbf{x}^\star_{\text{out},i}$ is far from $\mathcal{K}_{\text{in},i}$, and as a result, $\mathbf{x}^\star_{\text{out}}$ is not a good candidate to move $\mathbf{x}$ towards.

The second condition we impose, one of "sufficient suboptimality", ensures significant progress in the objective value can be made when updating $\mathbf{x}$. Formally, we check if

$$\langle \mathbf{c}, \mathbf{x}^\star_{\text{out}}\rangle + \frac{4m}{t} \leq \langle \mathbf{c}, \mathbf{x}\rangle. \tag{4.3.4}$$

If the inequality holds, then there is still "room for progress" to lower the value of $\langle \mathbf{c}, \mathbf{x}\rangle$ by updating $\mathbf{x}$; if the inequality is violated, it means we've "maxed out" on progress attainable with the current set of parameters and we update t instead. This inequality comes from the standard theory of interior point methods (e.g., Section .2.4.1 of [Ren01a]), wherein iterates $\mathbf{z}(\eta)$ along the central path parametrized by η satisfy $\langle \mathbf{c}, \mathbf{z}\rangle \leq \text{OPT} + \nu/\eta$, where ν is the self-concordance parameter of the barrier function. In our case, the barrier function ψ_{out} has a total self-concordance parameter of $\sum_{i=1}^n d_i = m$.

Once the two conditions hold, we move $\mathbf{x}$ towards $\mathbf{x}^\star_{\text{out}}$ in Line 22. The update step is normalized by the distance between $\mathbf{x}$ and $\mathbf{x}^\star_{\text{out}}$ measured in the local norm, which enforces $\mathbf{x} \in \mathcal{K}$ (since by the definition of self-concordance, the unit radius Dikin ball lies inside the domain of the self-concordance barrier), and also helps bound certain first-order error terms (Inequality 4.4.14 in Section 4.4.3). Our

update step (Line 22) thus plays a crucial role in both the algorithm and the analysis.

The rest of this section details the procedure for when either of these conditions is violated.

Updating the inner and outer convex sets. Suppose Inequality 4.3.3 is violated for some $i \in [n]$. Then $\mathbf{x}^\star_{\text{out},i} \notin \mathcal{K}_{\text{in},i}$, which in turn means $\mathbf{x}^\star_{\text{out}}$ *might* not be in the feasible set $\mathcal{K}$. To reestablish Inequality 4.3.3 for i, we can update one of $\mathcal{K}_{\text{in},i}$ or $\mathcal{K}_{\text{out},i}$ and recompute $\mathbf{x}^\star_{\text{out},i}$ by Equation (4.3.2).

To decide which option to take, we query $\mathcal{O}_i$ at the point $\mathbf{x}^\star_{\text{out},i}$: if the oracle indicates that $\mathbf{x}^\star_{\text{out},i} \in \mathcal{K}_i$, then we incorporate $\mathbf{x}^\star_{\text{out},i}$ into $\mathcal{K}_{\text{in},i}$ by redefining $\mathcal{K}_{\text{in},i} = \text{conv}(\mathcal{K}_{\text{in},i}, \mathbf{x}^\star_{\text{out},i})$ to be the convex hull of the current $\mathcal{K}_{\text{in},i}$ and $\mathbf{x}^\star_{\text{out},i}$ (Line 14). If, on the other hand, $\mathbf{x}^\star_{\text{out},i} \notin \mathcal{K}_i$, the oracle $\mathcal{O}_i$ will return a halfspace $\mathcal{H}_i$ satisfying $\mathcal{H}_i \supseteq \mathcal{K}_i$. Then we redefine $\mathcal{K}_{\text{out},i} = \mathcal{K}_{\text{out},i} \cap \mathcal{H}_i$ (Line 16). After processing this update of the sets, the algorithm recomputes $\mathbf{x}^\star_{\text{out}}$ and returns to the main loop since updating the sets does not necessarily imply that the new $\mathbf{x}^\star_{\text{out}}$ satisfies $\mathbf{x}^\star_{\text{out}} \in \mathcal{K}_{\text{in}}$.

This update rule for the sets is exactly where our novelty lies: *we do not arbitrarily update sets, rather, we update one only after checking the very specific condition* $\mathbf{x}^\star_{out,i} \notin \mathcal{K}_{in,i}$. Since the separation oracle is called only in this part of the algorithm, performing this check enables us to dramatically reduce the number of calls we make to the separation oracle, thereby improving our oracle complexity.

Further, this update rule shows that even when we cannot update the current $\mathbf{x}$, we make progress by using all the information from the oracles. Over the course of the algorithm, we gradually expand $\mathcal{K}_{\text{in}}$ and shrink $\mathcal{K}_{\text{out}}$, until they well-approximate $\mathcal{K}$. To formally quantify the change in volume due to the above operations, we consider the following alternative view of $\mathbf{x}^\star_{\text{out}}$.

Proposition 9 (Section 3 in [BE15]; Section 3 of [Kla06])**.** *Let $\theta \in \mathbb{R}^n$, and let p_θ be defined as $p_\theta(\mathbf{x}) \stackrel{def}{=} \exp(\langle \theta, \mathbf{x}\rangle - f(\theta))$, where $f(\theta) \stackrel{def}{=} \log\left[\int_{\mathcal{K}} \exp(\langle \theta, \mathbf{u}\rangle) d\mathbf{u}\right]$. Then,*

$$\mathbb{E}_{\mathbf{x}\sim p_\theta}[\mathbf{x}] = \arg\min_{\mathbf{x}\in \text{int}(\mathcal{K})} \{f^*(\mathbf{x}) - \langle \theta, \mathbf{x}\rangle\}.$$

By this proposition, $\mathbf{x}^\star_{\text{out}}$ defined in Equation (4.3.2) satisfies

$$\mathbf{x}^\star_{\text{out}} \stackrel{\text{def}}{=} \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[\mathbf{x}], \text{ where } \mathcal{D} \propto \exp\left\{-t\langle \mathbf{c}, \mathbf{x}\rangle - \log\left[\int_{\widehat{\mathcal{K}}_{\text{out}}} \exp(-t\langle \mathbf{c}, \mathbf{u}\rangle) d\mathbf{u}\right]\right\}, \tag{4.3.5}$$

that is, $\mathbf{x}^\star_{\text{out}}$ is the centroid of some exponential distribution over $\widehat{\mathcal{K}}_{\text{out}}$. As a result, if $\mathbf{x}^\star_{\text{out},i} \notin \mathcal{K}_i$, the hyperplane cutting $\widehat{\mathcal{K}}_{\text{out}}$ through $\mathbf{x}^\star_{\text{out}}$ will yield a large decrease in volume of $\widehat{\mathcal{K}}_{\text{out}}$, per Remark 7. Therefore, the query result in a large change in volume in either $\mathcal{K}_{\text{in}}$ or $\mathcal{K}_{\text{out}}$, allowing us to approximate $\mathcal{K}$ with a bounded number of iterations.

Updating t. If Inequality 4.3.4 is violated, then the current $\mathbf{x}$ is "as optimal as one can get" for the current parameter t. This could mean one of two things:

The first possibility is that we have already reached an approximate optimum, which we verify by checking whether $t \geq O(1/\epsilon)$ in Line 5: If true, this indicates that we have attained our desired suboptimality, and the algorithm terminates by returning

$$\mathbf{x}^{\text{ret}} = \arg\min_{\mathbf{x}:\mathbf{x}\in\mathcal{K}_{\text{in}}, \mathbf{A}\mathbf{x}=\mathbf{b}} \left\{ t\cdot\langle \mathbf{c}, \mathbf{x}\rangle + \sum_{i=1}^{n} \psi_{\text{in},i}(\mathbf{x}_i)\right\}.$$

The point $\mathbf{x}^{\text{ret}}$ is feasible because it is in $\mathcal{K}_{\text{in}}$ by definition, and the suboptimality of $O(1/t_{\text{end}}) = O(\epsilon)$ ensures it is an approximate optimum for the original problem.

The second possibility is that we need to increase t to set the next "target suboptimality". The value of t is increased by a scaling factor of $1 + O(1/m)$ in Line 8. This scaling factor ensures, like in the standard IPM framework, that the next optimum is not too far from the current one. In particular, our choice of $O(1/m)$ comes from having to choose $t = O(1/\nu)$, and in our case, the self-concordance parameter is $\nu = \sum_{i=1}^{n} d_i = m$. Following the update to t, we recompute $\mathbf{x}^{\star}_{\text{out}}$ by Equation (4.3.2). Since $\langle \mathbf{c}, \mathbf{x} \rangle > \langle \mathbf{c}, \mathbf{x}^{\star}_{\text{out}} \rangle + O(1/t)$ is not guaranteed with the new t and $\mathbf{x}^{\star}_{\text{out}}$, the algorithm jumps back to the start of the main loop.

4.4 Our Analysis

To analyze Algorithm 4.3.1, we define the following potential function that captures the changes in $\mathcal{K}_{\text{in},i}$, $\mathcal{K}_{\text{out},i}$, t, and $\mathbf{x}$ in each iteration:

$$\Phi \stackrel{\text{def}}{=} \underbrace{t\langle \mathbf{c}, \mathbf{x} \rangle + \log\left[\int_{\widehat{\mathcal{K}}_{\text{out}}} \exp(-t\langle \mathbf{c}, \mathbf{u} \rangle) d\mathbf{u}\right]}_{\text{entropic terms}} + \underbrace{\sum_{i \in [n]} \psi_{\text{in},i}(\mathbf{x}_i)}_{\text{universal terms}}, \tag{4.4.1}$$

where $\log\left[\int_{\widehat{\mathcal{K}}_{\text{out}}} \exp(-t\langle \mathbf{c}, \mathbf{u} \rangle) d\mathbf{u}\right]$ is related to the entropic barrier on $\widehat{\mathcal{K}}_{\text{out}}$ (see Section 4.4.1) and $\psi_{\text{in},i}$ is the universal barrier on $\mathcal{K}_i$. In the subsequent sections, we study the changes in each of these potential functions along with obtaining bounds on the initial and final potentials and combine them to bound the algorithm's separation oracle complexity.

4.4.1 Potential Change due to the Entropic Barrier

In this section, we study the changes in the entropic terms of Equation (4.4.1) upon updating the outer convex set $\widehat{\mathcal{K}}_{\text{out}}$ as well as t. These two changes are lumped together in this section because both updates affect the term $\log\left[\int_{\widehat{\mathcal{K}}_{\text{out}}} \exp(-t \cdot \langle \mathbf{c}, \mathbf{x} \rangle)\, d\mathbf{x}\right]$, albeit in different ways: the update in $\widehat{\mathcal{K}}_{\text{out}}$ affects it via Grünbaum's Theorem; the update in t affects it via the fact that, by duality with respect to the entropic barrier (Definition 5), $\log\left[\int_{\mathbf{x} \in \widehat{\mathcal{K}}_{\text{out}}} \exp(\langle \mathbf{x}, \theta \rangle) d\mathbf{x}\right]$ is also self-concordant. We detail these two potential changes below.

Lemma 4.4.1 (Potential analysis for outer set)**.** *Let* $\widehat{\mathcal{K}}_{out} \stackrel{def}{=} \{\mathbf{x} : \mathbf{x}_i \in \mathcal{K}_{\text{out},i} \cap \{\mathbf{y} : \mathbf{A}\mathbf{y} = \mathbf{b}\}\}$*, and let* $\Phi = t\langle \mathbf{c}, \mathbf{x} \rangle + \log\left[\int_{\widehat{\mathcal{K}}_{out}} \exp(-t\langle \mathbf{c}, \mathbf{u} \rangle) d\mathbf{u}\right] + \sum_{i \in [n]} \psi_{in,i}(\mathbf{x}_i)$*. Let* $\mathcal{H}_i$ *be the halfspace generated by the separation oracle* $\boldsymbol{O}_i$ *queried at* $\mathbf{x}^{\star}_{out,i}$ *as shown in Line 16 of Algorithm 4.3.1. Then the new potential* $\Phi^{(new)} = t\langle \mathbf{c}, \mathbf{x} \rangle + \log\left[\int_{\widehat{\mathcal{K}}_{out} \cap \mathcal{H}_i} \exp(-t\langle \mathbf{c}, \mathbf{u} \rangle) d\mathbf{u}\right] + \sum_{i \in [n]} \psi_{in,i}(\mathbf{x}_i)$ *is bounded from above as follows.*

$$\Phi^{(new)} \leq \Phi + \log(1 - 1/e).$$

Proof. The change in potential is given by

$$\Phi^{(\text{new})} - \Phi = \log\left[\frac{\int_{\widehat{\mathcal{K}}_{\text{out}} \cap \mathcal{H}_i} \exp(-t \cdot \langle \mathbf{c}, \mathbf{x} \rangle)\, d\mathbf{x}}{\int_{\widehat{\mathcal{K}}_{\text{out}}} \exp(-t \cdot \langle \mathbf{c}, \mathbf{x} \rangle)\, d\mathbf{x}}\right].$$

We now apply Theorem 4.2.9 to the right hand side, with the function $f(\mathbf{x}) = \exp(-t \cdot \langle \mathbf{c}, \mathbf{x} \rangle - A(t\mathbf{c}))$,

where $A(\theta) = \log\left[\int_{\widehat{\mathcal{K}}_{\text{out}}} \exp(-\langle\theta, \mathbf{x}\rangle) d\mathbf{x}\right]$. Noting that each halfspace $\mathcal{H}_i$ passes directly through $\mathbf{x}^\star_{\text{out},i}$, where $\mathbf{x}^\star_{\text{out}}$ is the centroid of $\widehat{\mathcal{K}}_{\text{out}}$ with respect to f (by the definition of $\mathbf{x}^\star_{\text{out}}$ in Equation (4.3.5)), Remark 7 applies and gives the claimed volume change. □

To capture the change in potential due to the update in t, we recall the alternative perspective to the function $\log\left[\int_{\widehat{\mathcal{K}}_{\text{out}}} \exp(-t\langle\mathbf{c}, \mathbf{x}\rangle) d\mathbf{x}\right]$ given by Definition 5 and derive the following technical result.

Lemma 4.4.2. *Consider a ν-self-concordant barrier $\psi : int(\mathcal{K}) \to \mathbb{R}$ over the interior of a convex set $\mathcal{K} \subseteq \mathbb{R}^d$. Define*

$$\zeta_t^\psi \stackrel{def}{=} \min_{\mathbf{x}} \left[t \cdot \langle\mathbf{c}, \mathbf{x}\rangle + \psi(\mathbf{x})\right] \text{ and } \mathbf{x}_t \stackrel{def}{=} \arg\min_{\mathbf{x}} \left[t \cdot \langle\mathbf{c}, \mathbf{x}\rangle + \psi(\mathbf{x})\right]. \tag{4.4.2}$$

Then for $0 \le h \le \frac{1}{3\sqrt{\nu}}$, we have

$$\zeta_t^\psi + th \cdot \langle\mathbf{x}_t, \mathbf{c}\rangle \ge \zeta_{t(1+h)}^\psi \ge \zeta_t^\psi + ht \cdot \langle\mathbf{c}, \mathbf{x}_t\rangle - h^2\nu.$$

Proof. The first inequality holds for any function ψ by invoking the definitions of $\zeta_{t(1+h)}^\psi$, ζ_t^ψ, and $\mathbf{x}_t$ from Equation (4.4.2):

$$\zeta_{t(1+h)}^\psi = \min_{\mathbf{x}} \left[t(1+h) \cdot \langle\mathbf{x}, \mathbf{c}\rangle + \psi(\mathbf{x})\right] \le t(1+h) \cdot \langle\mathbf{x}_t, \mathbf{c}\rangle + \psi(\mathbf{x}_t) = \zeta_t^\psi + th \cdot \langle\mathbf{x}_t, \mathbf{c}\rangle.$$

We now prove the second inequality of the lemma. This one specifically uses the self-concordance of ψ. Observe first, by definition,

$$\zeta_t^\psi = -\psi^*(-t\mathbf{c}). \tag{4.4.3}$$

Since ψ is a self-concordant barrier (and hence, a self-concordant function), Theorem 4.2.7 implies that ψ^* is a self-concordant function as well. Then, by applying Theorem 4.2.8 to ψ^* under the assumption $\|-th\mathbf{c}\|_{-t\mathbf{c}} \le 1$ yields

$$\psi^*(-t\mathbf{c} - th\mathbf{c}) \le \psi^*(-t\mathbf{c}) + \langle\nabla\psi^*(-t\mathbf{c}), -th\mathbf{c}\rangle + \left[\frac{1}{2}\|-th\mathbf{c}\|_{-t\mathbf{c}}^2 + \frac{\|-th\mathbf{c}\|_{-t\mathbf{c}}^3}{3(1-\|-th\mathbf{c}\|_{-t\mathbf{c}})}\right]. \tag{4.4.4}$$

By applying the first-order optimality condition to the definition of $\mathbf{x}_t$ in Equation (4.4.2), we see that

$$\nabla\psi(\mathbf{x}_t) = -t\mathbf{c}. \tag{4.4.5}$$

Next, evaluating $a \stackrel{\text{def}}{=} \|-th\mathbf{c}\|_{-t\mathbf{c}}$ to check the assumption $\|-th\mathbf{c}\|_{-t\mathbf{c}} \le 1$, we get

$$\begin{aligned} a^2 = h^2\langle -t\mathbf{c}, \nabla^2\psi^*(-t\mathbf{c}) \cdot (-t\mathbf{c})\rangle &= h^2\langle\nabla\psi(\mathbf{x}_t), \nabla^2\psi^*(\nabla\psi(\mathbf{x}_t)) \cdot \nabla\psi(\mathbf{x}_t)\rangle \\ &= h^2\langle\nabla\psi(\mathbf{x}_t), (\nabla^2\psi(\mathbf{x}_t))^{-1} \cdot \nabla\psi(\mathbf{x}_t)\rangle \\ &\le h^2\nu \end{aligned}$$

where we used Equation (4.4.5) and Lemma 4.2.3, in the first two equations and Definition 4 and the complexity value of ψ in the last step. Our range of h proves that $a \le 1$, which is what we need

for Inequality 4.4.4 to hold. We continue our computation to get

$$\left[\frac{1}{2}\|-th\mathbf{c}\|_{-t\mathbf{c}}^2+\frac{\|-th\mathbf{c}\|_{-t\mathbf{c}}^3}{3(1-\|-th\mathbf{c}\|_{-t\mathbf{c}})}\right]\leq\frac{1}{2}h^2\nu+\frac{1}{3}h^3\nu^{3/2}\leq\frac{1}{2}h^2\nu+\frac{1}{3}h^2\nu\leq h^2\nu. \tag{4.4.6}$$

Applying Lemma 4.2.2 to Equation (4.4.5) gives

$$\nabla\psi^*(-t\mathbf{c})=\mathbf{x}_t. \tag{4.4.7}$$

Plugging Equation (4.4.7) and Inequality 4.4.6 into the first and second-order terms, respectively, of Inequality 4.4.4 gives

$$\psi^*(-t\mathbf{c}-th\mathbf{c})\leq\psi^*(-t\mathbf{c})+\langle\mathbf{x}_t,-th\mathbf{c}\rangle+h^2\nu.$$

Plugging in Equation (4.4.3) gives the desired inequality and completes the proof. □

To finally compute the potential change due to t, we need the following result about the self-concordance parameter of the entropic barrier. While [BE15] prove that this barrier on a set in $\mathbb{R}^d$ is $(1+\epsilon_d)d$-self-concordant, the recent work of [Che21] remarkably improves this complexity to exactly d.

Theorem 4.4.3 ([Che21]). *The entropic barrier on any convex body $\mathcal{K}\subseteq\mathbb{R}^d$ is a d-self-concordant barrier.*

We may now compute the potential change due to change in t in Line 8.

Lemma 4.4.4. *When t is updated to $t\cdot\left[1+\frac{\eta}{4m}\right]$ in Line 8 of Algorithm 4.3.1, the potential Φ Equation (4.4.1) increases to $\Phi^{(new)}$ as follows:*

$$\Phi^{(new)}\leq\Phi+\eta+\eta^2.$$

Proof. Recall that the barrier function we use for the set $\widehat{\mathcal{K}}_{\text{out}}$ is the entropic barrier ψ_{out}. By Equation (4.4.2) and the definition of conjugate, we have

$$-\zeta_t^{\psi_{\text{out}}}=\max_{\mathbf{v}}\left[\langle-t\mathbf{c},\mathbf{v}\rangle-\psi_{\text{out}}(\mathbf{v})\right]=\psi_{\text{out}}^*(-t\mathbf{c}).$$

Applying Definition 5, taking the conjugate on both sides of the preceding equation, and using Lemma 4.2.1 then gives

$$-\zeta_t^{\psi_{\text{out}}}=\log\left[\int_{\widehat{\mathcal{K}}_{\text{out}}}\exp(-t\cdot\langle\mathbf{c},\mathbf{u}\rangle)\,d\mathbf{u}\right]. \tag{4.4.8}$$

From Equation (4.4.1), the change in potential by changing t to $t\cdot(1+h)$ for some $h>0$ may be expressed as

$$\Phi^{(\text{new})}-\Phi=\log\left[\int_{\widehat{\mathcal{K}}_{\text{out}}}\exp\langle-t(1+h)\mathbf{c},\mathbf{v}\rangle d\mathbf{v}\right]-\log\left[\int_{\widehat{\mathcal{K}}_{\text{out}}}\exp\langle-t\mathbf{c},\mathbf{v}\rangle d\mathbf{v}\right]+\langle th\cdot\mathbf{c},\mathbf{x}\rangle.$$

By applying $h=\frac{\eta}{4m}$ and $\nu=m$ (via a direct application of Theorem 4.4.3), we have $h=\frac{\eta}{4m}\leq\frac{1}{\sqrt{m}}=\frac{1}{\sqrt{\nu}}$, and so we may now apply Equation (4.4.8) and Lemma 4.4.2 in the preceding equation to obtain the following bound.

$$\Phi^{(\text{new})}-\Phi\leq th\langle\mathbf{c},\mathbf{x}\rangle-th\langle\mathbf{c},\mathbf{x}_t\rangle+h^2\nu.$$

From Equation (4.3.2) and Equation (4.4.2), we see that $\mathbf{x}_t$ for the entropic barrier satisfies the

equation $\mathbf{x}_t = \mathbf{x}^\star_{\text{out}}$, and applying the guarantee $\langle \mathbf{c}, \mathbf{x} \rangle \leq \langle \mathbf{c}, \mathbf{x}^\star_{\text{out}} \rangle + \frac{4m}{t}$ to this inequality, we obtain

$$\Phi^{(\text{new})} - \Phi \leq th \cdot \frac{4m}{t} + h^2 \nu = \eta + \left(\frac{\eta}{4m}\right)^2 \nu \leq \eta + \eta^2.$$

□

4.4.2 Potential Change due to the Universal Barrier

We now study the change in volume on growing the inner convex set $\mathcal{K}_{\text{in},i}$ in Line 14. As mentioned in Section 4.3, our barrier of choice on this set is the universal barrier introduced in [NN94] (see also [Gül97]). This barrier was constructed to demonstrate that *any* convex body in $\mathbb{R}^n$ admits an $O(n)$-self-concordant barrier, and its complexity parameter was improved to exactly n in [LY21].

Conceptually, we choose the universal barrier for the inner set because the operation we perform on the inner set (i.e., generating its convex hull with an external point $\mathbf{x}^\star_{\text{out}}$) is dual to the operation of intersecting the outer set with the separating halfspace containing $\mathbf{x}^\star_{\text{out}}$ (see Lemma 4.2.4), which suggests the use of a barrier dual to the entropic barrier used on the outer set. As explained in [BE15], for the special case of convex cones, the universal barrier is precisely one such barrier.

We now state a technical property of the universal barrier, which we use in the potential argument.

Lemma 4.4.5 ([LY21, Lemma 1], [NN94, Gül97])**.** *Given a convex set $\mathcal{K} \in \mathbb{R}^d$ and $\mathbf{x} \in \mathcal{K}$, let $\psi_{\mathcal{K}}(\mathbf{x}) \stackrel{def}{=} \log \text{vol}(\mathcal{K} - \mathbf{x})^\circ$ be the universal barrier defined on $\mathcal{K}$ with respect to $\mathbf{x}$. Let $\mu \in \mathbb{R}^d$ be the center of gravity and $\Sigma \in \mathbb{R}^{d \times d}$ be the covariance matrix of the body $(\mathcal{K} - \mathbf{x})^\circ$, where $(\mathcal{K} - \mathbf{x})^\circ = \{\mathbf{y} \in \mathbb{R}^n : \mathbf{y}^\top (\mathbf{z} - \mathbf{x}) \leq 1, \forall \mathbf{z} \in \mathcal{K}\}$ is the polar set of $\mathcal{K}$ with respect to $\mathbf{x}$. Then, we have that*[4]

$$\nabla \psi_{\mathcal{K}}(\mathbf{x}) = (d+1)\mu, \ \nabla^2 \psi_{\mathcal{K}}(\mathbf{x}) = (d+1)(d+2)\Sigma + (d+1)\mu\mu^\top.$$

Lemma 4.4.6. *Given a convex set $\mathcal{K} \subseteq \mathbb{R}^d$ and a point $\mathbf{x} \in \mathcal{K}$. Let $\psi_{\mathcal{K}} \stackrel{def}{=} \log \text{vol}(\mathcal{K} - \mathbf{x})^\circ$ be the universal barrier defined on $\mathcal{K}$ with respect to $\mathbf{x}$. Let $\eta \leq 1/4$ and $\mathbf{y} \notin \mathcal{K}$ be a point satisfying the following condition*

$$\langle \nabla \psi_{\mathcal{K}}(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \eta \|\mathbf{y} - \mathbf{x}\|_{\mathbf{x}} \geq 4d, \tag{4.4.9}$$

where the local norm is with respect to $\psi_{\mathcal{K}}(\mathbf{x})$. Construct the new set $\text{conv}\{\mathcal{K}, \mathbf{y}\}$. Then, the value of the universal barrier defined on this new set with respect to $\mathbf{x}$ satisfies the following inequality.

$$\psi_{\mathcal{K},new}(\mathbf{x}) \stackrel{def}{=} \psi_{\text{conv}\{\mathcal{K},\mathbf{y}\}}(\mathbf{x}) = \log \text{vol}(\text{conv}\{\mathcal{K}, \mathbf{y}\} - \mathbf{x})^\circ \leq \psi_{\mathcal{K}}(\mathbf{x}) + \log(1 - 1/e + \eta).$$

Proof. Denote $\mathbf{y} - \mathbf{x} = \mathbf{v}$. By Lemma 4.2.4, we have that

$$(\text{conv}\{\mathcal{K}, \mathbf{y}\} - \mathbf{x})^\circ \subseteq (\mathcal{K} - \mathbf{x})^\circ \cap \mathcal{H},$$

where $\mathcal{H} = \{\mathbf{z} \in \mathbb{R}^n : \langle \mathbf{z}, \mathbf{v} \rangle \leq 1\}$. Therefore, our strategy to computing the deviation of $\psi_{\text{conv}\{\mathcal{K},\mathbf{y}\}}(\mathbf{x}) = \log \text{vol}(\text{conv}\{\mathcal{K}, \mathbf{y}\} - \mathbf{x})^\circ$ from $\psi_{\mathcal{K}}(\mathbf{x})$ is to instead compute the change in $\text{vol}[(\mathcal{K} - \mathbf{x})^\circ \cap \mathcal{H}]$ from $\text{vol}(\mathcal{K} - \mathbf{x})^\circ$, for which it is immediate that one may apply an appropriate form of Grünbaum's Theorem.

[4] The expression for $\nabla \psi_{\mathcal{K}}(\mathbf{x})$ in [NN94] has a typo in the sign.

Let μ be the center of gravity of the body $(\mathcal{K}-\mathbf{x})^\circ$. If $\mu \in \partial\mathcal{H}$, then Corollary 8 (with $\eta = 0$) gives

$$\mathrm{vol}\left[(\mathcal{K}-\mathbf{x})^\circ \cap \mathcal{H}\right] \le \mathrm{vol}(\mathcal{K}-\mathbf{x})^\circ \cdot (1-1/e),$$

and taking the logarithm on both sides gives the claimed bound. We now consider the case in which $\mu \in \mathcal{H}$, and the variance matrix of the body $(\mathcal{K}-\mathbf{x})^\circ$ is Σ. Consider the point

$$\mathbf{z} = \mu + \frac{1-\langle \mathbf{v}, \mu\rangle}{4\|\mathbf{v}\|_\Sigma^2} \cdot \Sigma\mathbf{v}.$$

Since we are assuming $\mu \in \mathcal{H}$, it means $\langle \mathbf{v}, \mu\rangle \le 1$; using this, we can check that the point $\mathbf{z}$ defined above satisfies $\langle \mathbf{v}, \mathbf{z}\rangle \le 1$, which implies $\mathbf{z} \in \mathcal{H}$. We show that $\mathbf{z}$ is sufficiently close to μ, so that even though $\mu \in \mathcal{H}$, the subset of $(\mathcal{K}-\mathbf{x})^\circ$ cut out by the halfspace $\mathcal{H}$ is sufficiently large. By applying Lemma 4.4.5 to express $\|\mathbf{v}\|_\mathbf{x}^2$ as $\|\mathbf{v}\|_\mathbf{x}^2 = (d+1)(d+2)\|\mathbf{v}\|_\Sigma^2 + (d+1)\langle \mathbf{v}, \mu\rangle^2$, we may compute the following quantity.

$$\|\mathbf{z}-\mu\|_{\Sigma^{-1}} = \sqrt{(d+1)(d+2)} \cdot \frac{1-\langle \mathbf{v}, \mu\rangle}{4\sqrt{\frac{1}{2}\|\mathbf{v}\|_\mathbf{x}^2 + \frac{1}{2}\|\mathbf{v}\|_\mathbf{x}^2 - (d+1)\langle \mathbf{v}, \mu\rangle^2}}. \tag{4.4.10}$$

Applying the expression for gradient from Lemma 4.4.5 in Equation (4.4.9), we have

$$\eta\|\mathbf{v}\|_\mathbf{x} \ge 4d - (d+1)\langle \mathbf{v}, \mu\rangle. \tag{4.4.11}$$

We observe that $4d \ge (d+1)\langle \mathbf{v}, \mu\rangle$ (trivially true if $\langle \mathbf{v}, \mu\rangle \le 0$ and otherwise, true because $d \ge 1$ and $\langle \mathbf{v}, \mu\rangle \le 1$). Therefore, we can square both sides to see $\eta^2\|\mathbf{v}\|_\mathbf{x}^2 \ge 16d^2 - 8d(d+1)\langle \mathbf{v}, \mu\rangle + (d+1)(\langle \mathbf{v}, \mu\rangle)^2$. Using the facts that $\eta \le 1/4$ and that $\mu \in \mathcal{H}$ implies $\langle \mathbf{v}, \mu\rangle \le 1$, we can conclude from this inequality that $\frac{1}{2}\|\mathbf{v}\|_\mathbf{x}^2 \ge (d+1)\langle \mathbf{v}, \mu\rangle^2$. Plugging this in Equation (4.4.10) gives

$$\|\mathbf{z}-\mu\|_{\Sigma^{-1}} \le \sqrt{(d+1)(d+2)} \cdot \frac{1-\langle \mathbf{v}, \mu\rangle}{4\sqrt{\frac{1}{2}\|\mathbf{v}\|_\mathbf{x}^2}} \le d\frac{1-\langle \mathbf{v}, \mu\rangle}{\|\mathbf{v}\|_\mathbf{x}} \le d \cdot \frac{1-\langle \mathbf{v}, \mu\rangle}{d(1-\langle \mathbf{v}, \mu\rangle)/\eta} \le \eta,$$

where the final step is by Equation (4.4.11). At this point, Corollary 8 applies, giving us the claimed potential change. □

4.4.3 Potential Change for the Update of x

In this section, we quantify the amount of progress made in Line 21 of Algorithm 4.3.1 by computing the change in the potential Φ as defined in Equation (4.4.1).

Lemma 4.4.7. *Consider the potential Φ Equation (4.4.1) and the update step $\delta_\mathbf{x} = \frac{\eta}{2} \cdot \frac{\mathbf{x}^\star_{out}-\mathbf{x}}{\|\mathbf{x}^\star_{out}-\mathbf{x}\|_{\mathbf{x},1}}$ as in Line 21. Assume the guarantees in Inequality 4.3.3 and Inequality 4.3.4. Then the potential Φ incurs the following minimum decrease.*

$$\Phi^{(new)} \le \Phi - \frac{\eta^2}{4}.$$

Proof. Taking the gradient of Φ with respect to $\mathbf{x}$ and rearranging the terms gives

$$t\mathbf{c} = \nabla_\mathbf{x}\Phi - \sum_{i=1}^n \nabla\psi_{\mathrm{in},i}(\mathbf{x}_i), \tag{4.4.12}$$

where we are overloading notation in $\nabla\psi_{\text{in},i}(\mathbf{x}_i)$ to mean the d-dimensional vector equalling the appropriate entries at the d_i coordinates corresponding to $\mathbf{x}_i$ and zero elsewhere. By applying the expression for $t\mathbf{c}$ from the preceding equation, we get

$$\begin{aligned}\Phi^{(\text{new})} - \Phi &= t\langle \mathbf{c}, \mathbf{x} + \delta_{\mathbf{x}}\rangle + \sum_{i=1}^{n} \psi_{\text{in},i}(\mathbf{x}_i + \delta_{\mathbf{x},i}) - t\langle \mathbf{c}, \mathbf{x}\rangle - \sum_{i=1}^{n} \psi_{\text{in},i}(\mathbf{x}_i) \\ &= \langle \nabla_{\mathbf{x}}\Phi, \delta_{\mathbf{x}}\rangle + \sum_{i=1}^{n} \underbrace{\left[\psi_{\text{in},i}(\mathbf{x}_i + \delta_{\mathbf{x},i}) - \psi_{\text{in},i}(\mathbf{x}_i) - \langle \nabla\psi_{\text{in},i}(\mathbf{x}_i), \delta_{\mathbf{x},i}\rangle\right]}_{q_{\psi_{\text{in},i}}(\mathbf{x}_i)}. \end{aligned} \tag{4.4.13}$$

Note that in substituting Equation (4.4.12) above, we crucially use that $\mathbf{x}_i$ are all disjoint vectors whose coordinates completely cover those of $\mathbf{x}$. The term $q_{\psi_{\text{in},i}}(\mathbf{x}_i)$ in Equation (4.4.13) the error due to first-order approximation of $\psi_{\text{in},i}$ around $\mathbf{x}_i$. Since each of the $\psi_{\text{in},i}(\mathbf{x}_i)$ is a self-concordant function and $\|\delta_{\mathbf{x},i}\|_{\mathbf{x}_i} \leq \|\delta_{\mathbf{x}}\|_{\mathbf{x},1} \leq \eta \leq 1/4$, Theorem 4.2.8 applies and gives

$$\psi_{\text{in},i}(\mathbf{x}_i + \delta_{\mathbf{x},i}) - \psi_{\text{in},i}(\mathbf{x}_i) - \langle \nabla\psi_{\text{in},i}(\mathbf{x}_i), \delta_{\mathbf{x},i}\rangle \leq \|\delta_{\mathbf{x},i}\|_{\mathbf{x}_i}^2. \tag{4.4.14}$$

Plugging in Inequality 4.4.14 into Equation (4.4.13), we get

$$\Phi^{(\text{new})} - \Phi \leq \langle \nabla_{\mathbf{x}}\Phi, \delta_{\mathbf{x}}\rangle + \sum_{i=1}^{n} \|\delta_{\mathbf{x},i}\|_{\mathbf{x}_i}^2. \tag{4.4.15}$$

We now bound the two terms on the right hand side one at a time. Using the definition of $\delta_{\mathbf{x}}$ (as given in the statement of the lemma) and of $\nabla_{\mathbf{x}}\Phi$ from Equation (4.4.12) gives

$$\begin{aligned}\langle \nabla_{\mathbf{x}}\Phi, \delta_{\mathbf{x}}\rangle &= \frac{\eta}{2}\frac{1}{\|\mathbf{x}^\star_{\text{out}} - \mathbf{x}\|_{\mathbf{x},1}}\langle \nabla_{\mathbf{x}}\Phi, \mathbf{x}^\star_{\text{out}} - \mathbf{x}\rangle \\ &= \frac{\eta}{2}\frac{1}{\|\mathbf{x}^\star_{\text{out}} - \mathbf{x}\|_{\mathbf{x},1}}\left[\langle t\mathbf{c}, \mathbf{x}^\star_{\text{out}} - \mathbf{x}\rangle + \sum_{i=1}^{n}\langle \nabla\psi_{\text{in},i}(\mathbf{x}_i), \mathbf{x}^\star_{\text{out},i} - \mathbf{x}_i\rangle\right] \\ &\leq \frac{\eta}{2}\frac{1}{\|\mathbf{x}^\star_{\text{out}} - \mathbf{x}\|_{\mathbf{x},1}}\left[\langle t\mathbf{c}, \mathbf{x}^\star_{\text{out}} - \mathbf{x}\rangle + \sum_{i=1}^{n}\left(4d_i - \eta\|\mathbf{x}^\star_{\text{out},i} - \mathbf{x}_i\|_{\mathbf{x}_i}\right)\right] \\ &= \frac{\eta}{2}\frac{1}{\|\mathbf{x}^\star_{\text{out}} - \mathbf{x}\|_{\mathbf{x},1}}\left[\langle t\mathbf{c}, \mathbf{x}^\star_{\text{out}} - \mathbf{x}\rangle + 4m - \eta\|\mathbf{x}^\star_{\text{out}} - \mathbf{x}\|_{\mathbf{x},1}\right] \\ &\leq \frac{\eta}{2}\frac{1}{\|\mathbf{x}^\star_{\text{out}} - \mathbf{x}\|_{\mathbf{x},1}}\cdot\left(-\eta\|\mathbf{x}^\star_{\text{out}} - \mathbf{x}\|_{\mathbf{x},1}\right) \\ &= -\eta^2/2. \end{aligned} \tag{4.4.16}$$

where the third step follows from Inequality 4.3.3, the fourth step follows from $\sum_{i=1}^{n} d_i = m$, and the fifth step follows from Inequality 4.3.4. To bound the second term, we note from Line 21 that $\delta_{\mathbf{x},i} = \frac{\eta}{2}\cdot\frac{\mathbf{x}^\star_{\text{out},i} - \mathbf{x}_i}{\|\mathbf{x}^\star_{\text{out}} - \mathbf{x}\|_{\mathbf{x},1}}$, which implies

$$\sum_{i=1}^{n} \|\delta_{\mathbf{x},i}\|_{\mathbf{x}_i}^2 \leq \eta^2/4. \tag{4.4.17}$$

Plugging in Inequality 4.4.16 and Equation (4.4.17) into Inequality 4.4.15 finishes the proof. □

4.4.4 Total Oracle Complexity

Before we bound the total oracle complexity of the algorithm, we first bound the total potential change throughout the algorithm.

Lemma 4.4.8. *Consider the potential function* $\Phi = t\langle \mathbf{c}, \mathbf{x}\rangle + \log\left[\int_{\widehat{\mathcal{K}}_{out}} \exp(-t\langle \mathbf{c}, \mathbf{u}\rangle)d\mathbf{u}\right] + \sum_{i\in[n]} \psi_{in,i}(\mathbf{x}_i)$ *as defined in Equation (4.4.1) associated with Algorithm 4.3.1. Let* Φ_{init} *be the potential at* $t = t_{init}$ *of this algorithm, and let* Φ_{end} *be the potential at* $t = t_{end}$*. Suppose at* $t = t_{init}$ *in Algorithm 4.3.1, we have* $\mathcal{B}_m(\mathbf{x}, \bar{r}) \subseteq \mathcal{K}_{in}$ *with* $\bar{r} = r/\operatorname{poly}(m)$ *and* $\mathcal{K}_{out} \subseteq \mathcal{B}_m(0, \bar{R})$ *with* $\bar{R} = O(\sqrt{n}R)$*. Then we have, under the assumptions of Theorem 4.4.10, that*

$$\Phi_{init} - \Phi_{end} \leq O\left(m \log\left(\frac{mR}{\epsilon r}\right)\right).$$

Proof. For this proof, we introduce the following notation: let $\mathrm{vol}_{\mathbf{A}}(\cdot)$ denote the volume restricted to the subspace $\{\mathbf{x} : \mathbf{A}\mathbf{x} = \mathbf{b}\}$. We also invoke Fact 4.2.10. We now bound the change in the potential term by term, starting with the entropic terms

$$t\langle \mathbf{c}, \mathbf{x}\rangle + \log\left[\int_{\widehat{\mathcal{K}}_{\text{out}}} \exp(-t\langle \mathbf{c}, \mathbf{u}\rangle)d\mathbf{u}\right] \tag{4.4.18}$$

at $t = t_{\text{init}}$ and a lower bound on it at $t = t_{\text{end}}$. We start with bounding Equation (4.4.18) evaluated at $t = t_{\text{end}} = \frac{8m}{\epsilon\|\mathbf{c}\|_2 R}$.

Let $\bar{\mathbf{x}} = \arg\min_{\mathbf{x}\in\widehat{\mathcal{K}}_{\text{out}}} \langle \mathbf{c}, \mathbf{x}\rangle$ and $\alpha = \langle \mathbf{c}, \bar{\mathbf{x}}\rangle$. By optimality of $\bar{\mathbf{x}}$, we know that $\bar{\mathbf{x}} \in \partial\widehat{\mathcal{K}}_{\text{out}}$. Denote $\mathcal{B}_A(\mathbf{z}, \bar{r})$ to be $\mathcal{B}(\mathbf{z}, \bar{r})$ restricted to the subspace $\{\mathbf{x} : \mathbf{A}\mathbf{x} = \mathbf{b}\}$. Note that $\widehat{\mathcal{K}}_{\text{out}} \supseteq \mathcal{B}_{\mathbf{A}}(\mathbf{z}, \bar{r})$. Consider the cone $\mathcal{C}$ and halfspace $\mathcal{H}$ defined by

$$\mathcal{C} = \bar{\mathbf{x}} + \{\lambda\mathbf{y} : \lambda > 0, \mathbf{y} \in \mathcal{B}_{\mathbf{A}}(\mathbf{z} - \bar{\mathbf{x}}, \bar{r})\} \text{ and } \mathcal{H} \stackrel{\text{def}}{=} \left\{\mathbf{x} : \langle \mathbf{c}, \mathbf{x}\rangle \leq \alpha + \frac{1}{t_{\text{end}}}\right\}.$$

Then, by a similarity argument, we note that $\mathcal{C} \cap \mathcal{H}$ contains a cone with height $\frac{1}{t_{\text{end}}\|c\|_2}$ and base radius $\frac{\bar{r}}{\bar{R}t_{\text{end}}\|c\|_2}$, which means

$$\mathrm{vol}_{\mathbf{A}}(\mathcal{C} \cap \mathcal{H}) \geq \frac{1}{m - \operatorname{rank}(\mathbf{A})} \cdot \frac{1}{t_{\text{end}}\|c\|_2} \cdot \left(\frac{\bar{r}}{\bar{R}t_{\text{end}}\|c\|_2}\right)^{m-\operatorname{rank}(\mathbf{A})-1} \cdot \mathrm{vol}(\mathcal{B}_{m-\operatorname{rank}(\mathbf{A})-1}(0, 1)).$$

Then, we have

$$\begin{aligned}
\log\left[\int_{\widehat{\mathcal{K}}_{\text{out}}} \exp(-t_{\text{end}}\langle \mathbf{c}, \mathbf{u}\rangle) d\mathbf{u}\right] + t_{\text{end}}\langle \mathbf{c}, \mathbf{x}\rangle &\geq \log\left[\int_{\widehat{\mathcal{K}}_{\text{out}}} \exp(-t_{\text{end}}\langle \mathbf{c}, \mathbf{u}\rangle) d\mathbf{u}\right] + t_{\text{end}} \min_{\mathbf{x}\in\widehat{\mathcal{K}}_{\text{out}}} \langle \mathbf{c}, \mathbf{x}\rangle \\
&\geq \log\left[\int_{C\cap\mathcal{H}} \exp(-t_{\text{end}}\langle \mathbf{c}, \mathbf{u}\rangle) d\mathbf{u}\right] + t_{\text{end}}\alpha \\
&\geq \log\left[\int_{C\cap\mathcal{H}} \exp(-t_{\text{end}}\alpha - 1) d\mathbf{u}\right] + t_{\text{end}}\alpha \\
&= \log\left[\frac{1}{e}\cdot \text{vol}_{\mathbf{A}}(C\cap\mathcal{H})\exp(-t_{\text{end}}\alpha)\right] + t_{\text{end}}\alpha \\
&= \log\left[\text{vol}_{\mathbf{A}}(C\cap\mathcal{H})\cdot\frac{1}{e}\right] \\
&\geq -(m - \text{rank}(\mathbf{A}) - 1)\cdot\log(\bar{R}t_{\text{end}}\|c\|_2/\bar{r})) \\
&\quad + \log\Big(\text{vol}(\mathcal{B}_{m-\text{rank}(\mathbf{A})-1}(0,1))\Big) \\
&\quad - \log(m - \text{rank}(\mathbf{A})) - \log(t_{\text{end}}\|\mathbf{c}\|_2) - 1. \qquad (4.4.19)
\end{aligned}$$

Next, to bound Equation (4.4.18) at $t = t_{\text{init}}$, we may express these terms as follows.

$$\begin{aligned}
\log\left[\int_{\widehat{\mathcal{K}}_{\text{out}}} \exp(-t_{\text{init}}\cdot\langle \mathbf{c}, \mathbf{u}\rangle) d\mathbf{u}\right] + t_{\text{init}}\cdot\langle \mathbf{c}, \mathbf{x}\rangle &\leq \log\left[\text{vol}_{\mathbf{A}}(\widehat{\mathcal{K}}_{\text{out}})\right] + t_{\text{init}}\cdot \max_{\mathbf{u}\in\widehat{\mathcal{K}}_{\text{out}}} \langle \mathbf{c}, \mathbf{x}-\mathbf{u}\rangle \\
&\leq \log\Big(\text{vol}(\mathcal{B}_{m-\text{rank}(\mathbf{A})}(0,\bar{R}))\Big) + t_{\text{init}}\cdot 2\bar{R}\|\mathbf{c}\|_2 \\
&\leq \log\Big(\text{vol}(\mathcal{B}_{m-\text{rank}(\mathbf{A})}(0,1))\Big) \\
&\quad + (m - \text{rank}(\mathbf{A}))\log\bar{R} + O(m\log m), \qquad (4.4.20)
\end{aligned}$$

where the second step is by $\widehat{\mathcal{K}}_{\text{out}} \subseteq \mathcal{K}_{\text{out}} \subseteq \mathcal{B}_{\sum_{i\in[n]} d_i}(0,\bar{R})$ (here, the second inclusion is by assumption), and the third step is by $\text{vol}(\mathcal{B}_q(0,\bar{R})) = \frac{\pi^{q/2}}{\Gamma(1+q/2)}\bar{R}^q$ and our choice of $t_{\text{init}} \stackrel{\text{def}}{=} \frac{m\log m}{\sqrt{n}\|\mathbf{c}\|_2 R}$.

We now compute the change in the entropic barrier $\sum_{i\in[n]}\psi_{\text{in},i}(\mathbf{x}_i)$, where

$$\psi_{\text{in},i}(\mathbf{x}_i) = \log\text{vol}(\mathcal{K}^\circ_{\text{in},i}(\mathbf{x}_i)).$$

Define $\mathcal{B}_d(0,r)$ to be the d-dimensional Euclidean ball centred at the origin and with radius r. We note by the radius assumption of Theorem 4.4.10 that $\mathcal{K}_{\text{in},i} \subseteq \mathcal{K}_i \subseteq \mathcal{B}_{d_i}(0,\bar{R})$ throughout the algorithm. By the assumption made in this lemma's statement, we have that at the start of Algorithm 4.3.1, $\mathcal{K}_{\text{in},i} \supseteq \mathcal{B}_{d_i}(\mathbf{x},\bar{r})$. These give us the following bounds.

$$\psi^{\text{end}}_{\text{in},i}(\mathbf{x}_i) \geq \log\Big(\text{vol}(\mathcal{B}^\circ_{d_i}(0,\bar{R}))\Big) \text{ and } \psi^{\text{init}}_{\text{in},i}(\mathbf{x}_i) \leq \log\Big(\text{vol}(\mathcal{B}^\circ_{d_i}(\mathbf{x}_i,\bar{r}))\Big).$$

Applying the fact that $\text{vol}(\mathcal{B}_d(0,r)) \propto r^d$ and summing over all $i \in [n]$ gives

$$\begin{aligned}
\sum_{i\in[n]}\left[\psi^{\text{init}}_{\text{in},i}(\mathbf{x}_i) - \psi^{\text{end}}_{\text{in},i}(\mathbf{x}_i)\right] &\leq \sum_{i\in[n]}\log\left(\frac{\text{vol}(\mathcal{B}_{d_i}(\mathbf{x}_i, 1/\bar{r}))}{\text{vol}(\mathcal{B}_{d_i}(0,1/\bar{R}))}\right) \\
&= \sum_{i\in[n]} d_i\log(\bar{R}/\bar{r}) = m\log(\bar{R}/\bar{r}). \qquad (4.4.21)
\end{aligned}$$

Combining Inequality 4.4.20, Inequality 4.4.19, and Inequality 4.4.21, we have

$$\begin{aligned}\Phi_{\text{init}} - \Phi_{\text{end}} &\leq m\log(mR/r) + \left[\log\left(\text{vol}(\mathcal{B}_{m-\text{rank}(\mathbf{A})}(0,1))\right) + (m-\text{rank}(\mathbf{A}))\log\bar{R} + O(m\log m)\right] \\ &\quad + (m-\text{rank}(\mathbf{A})-1)\cdot\log(\bar{R}t_{\text{end}}\|c\|_2/\bar{r}) - \log\left(\text{vol}(\mathcal{B}_{m-\text{rank}(\mathbf{A})-1}(0,1))\right) \\ &\quad + \log(m-\text{rank}(\mathbf{A})) + \log(t_{\text{end}}\|\mathbf{c}\|_2) + 1 \\ &\leq m\log(mR/\epsilon r) + O(m\log m) + O((m-\text{rank}(\mathbf{A}))\log(mR/\epsilon r)) \\ &\leq O(m\log(mR/\epsilon r)).\end{aligned}$$

□

Lemma 4.4.9. *[Total oracle complexity] Suppose the inputs $\mathcal{K}_{in}$ and $\mathcal{K}_{out}$ to Algorithm 4.3.1 satisfy $\mathcal{K}_{out} \subseteq \mathcal{B}_m(0,\bar{R})$ with $\bar{R} = O(\sqrt{n}R)$ and $\mathcal{K}_{in} \supseteq \mathcal{B}(\mathbf{z},\bar{r})$ with $\bar{r} = r/\text{poly}(m)$. Then, when Algorithm 4.3.1 terminates at $t \geq t_{end}$, it outputs a solution $\mathbf{x}$ that satisfies*

$$\mathbf{c}^\top\mathbf{x} \leq \min_{\mathbf{x}\in\mathcal{K}, A\mathbf{x}=\mathbf{b}} \mathbf{c}^\top\mathbf{x} + \epsilon\cdot\|\mathbf{c}\|_2 R$$

using at most $\mathcal{N}_{sep} = O\left(m\log\left(\frac{mR}{\epsilon r}\right)\right)$ separation oracle calls.

Proof. Let $\mathcal{N}_{\text{t}}$ be the number of times t is updated; $\mathcal{N}_{\text{in}}$ the number of times $\mathcal{K}_{\text{in}}$ is updated; $\mathcal{N}_{\text{out}}$ the number of times $\mathcal{K}_{\text{out}}$ is updated; $\mathcal{N}_{\mathbf{x}}$ the number of times $\mathbf{x}$ is updated, and $\mathcal{N}_{\text{total}}$ the total number of iterations of the `while` loop before termination of Algorithm 4.3.1. Then, combining Lemma 4.4.1, Lemma 4.4.4, Lemma 4.4.6, and Lemma 4.4.7 gives

$$\Phi_{\text{end}} \leq \Phi_{\text{init}} + \mathcal{N}_{\text{out}}\cdot\log(1-1/e) + \mathcal{N}_{\text{t}}\cdot(\eta+\eta^2) + \mathcal{N}_{\text{in}}\cdot\log(1-1/e+\eta) + \mathcal{N}_{\mathbf{x}}\cdot\left(-\frac{\eta^2}{4}\right). \tag{4.4.22}$$

The initialization step of Algorithm 4.3.1 chooses $\eta = 1/100$, $t_{\text{end}} = \frac{8m}{\epsilon\|\mathbf{c}\|_2 R}$, and $t_{\text{init}} = \frac{m\log(m)}{\sqrt{n}\|\mathbf{c}\|_2 R}$, and we always update t by a multiplicative factor of $1+\frac{\eta}{4m}$ (see Line 8); therefore, we have

$$\mathcal{N}_{\text{t}} = O(m\log(mR/(\epsilon r)).$$

From Algorithm 4.3.1, the only times the separation oracle is invoked is when updating $\mathcal{K}_{\text{in}}$ or $\mathcal{K}_{\text{out}}$ in Line 14 and Line 16, respectively. Therefore, the total separation oracle complexity is $\mathcal{N}_{\text{sep}} = \mathcal{N}_{\text{in}} + \mathcal{N}_{\text{out}}$. Therefore, we have

$$\mathcal{N}_{\text{sep}} = \mathcal{N}_{\text{in}} + \mathcal{N}_{\text{out}} \leq O(1)\cdot[\Phi_{\text{init}} - \Phi_{\text{end}} + \mathcal{N}_{\text{t}}] = O(m\log(mR/(\epsilon r))$$

This gives the claimed separation oracle complexity.

We now prove the guarantee on approximation. Let $\mathbf{x}_{\text{output}}$ be the output of Algorithm 4.3.1 and $\mathbf{x}$ be the point which entered Line 4 right before termination. Note that the termination of Algorithm 4.3.1 implies, by Line 4, that

$$\mathbf{c}^\top\mathbf{x}_{\text{output}} \leq \mathbf{c}^\top\mathbf{x} + \frac{\nu}{t_{\text{end}}} \leq \mathbf{c}^\top\mathbf{x}^\star_{\text{out}} + \frac{4(n+m)}{t_{\text{end}}} \leq \min_{\mathbf{x}\in\mathcal{K}, A\mathbf{x}=\mathbf{b}} \mathbf{c}^\top\mathbf{x} + \epsilon\cdot\|\mathbf{c}\|_2\cdot R$$

where the first step is by the second inequality in Lemma 4.5.7 (using the universal barrier) and the last step follows by our choice of t_{end} and the definition of $\mathbf{x}^\star_{\text{out}}$ and $\mathcal{K}_{\text{out}} \supseteq \mathcal{K}$. □

Theorem 4.4.10 (Main theorem of Problem 4.3.1)**.** *Given the convex program*

$$\begin{array}{ll} \textit{minimize} & \langle \mathbf{c}, \mathbf{x} \rangle, \\ \textit{subject to} & \mathbf{x}_i \in \mathcal{K}_i \subseteq \mathbb{R}^{d_i+1} \forall i \in [n], \\ & \mathbf{A}\mathbf{x} = \mathbf{b}. \end{array}$$

Denote $\mathcal{K} = \mathcal{K}_1 \times \mathcal{K}_2 \times \ldots \times \mathcal{K}_n$*. Assuming we have*

- *outer radius R: For any* $\mathbf{x}_i \in \mathcal{K}_i$*, we have* $\|\mathbf{x}_i\|_2 \leq R$*, and*
- *inner radius r: There exists a* $\mathbf{z} \in \mathbb{R}^d$ *such that* $\mathbf{A}\mathbf{z} = \mathbf{b}$ *and* $\mathcal{B}(\mathbf{z}, r) \subset \mathcal{K}$*,*

then, for any $0 < \epsilon < \frac{1}{2}$*, we can find a point* $\mathbf{x} \in \mathcal{K}$ *satisfying* $\mathbf{A}\mathbf{x} = \mathbf{b}$ *and*

$$\langle \mathbf{c}, \mathbf{x} \rangle \leq \min_{\substack{\mathbf{x}_i \in \mathcal{K}_i \subseteq \mathbb{R}^{d_i+1} \forall i \in [n], \\ \mathbf{A}\mathbf{x} = \mathbf{b}}} \langle \mathbf{c}, \mathbf{x} \rangle + \epsilon \cdot \|\mathbf{c}\|_2 \cdot R,$$

in $O(\mathrm{poly}(m \log(mR/\epsilon r)))$ *time and using* $O(m \log(mR/(\epsilon r))$ *gradient oracle calls, where* $m = \sum_{i=1}^n d_i$*.*

Proof. We apply Theorem 4.5.1 for each $\mathcal{K}_i$ separately to find a solution $\mathbf{z}_i$. Then $\mathbf{z} = (\mathbf{z}_1, \ldots, \mathbf{z}_n) \in \mathbb{R}^{m+n}$ satisfies $\mathcal{B}_{m+n}(\mathbf{z}, \bar{r}) \subset \mathcal{K}$ with $\bar{r} = \frac{r}{6m^{3.5}}$. Then, we modified convex problem as in Definition 10 with $s = 2^{16} \frac{m^{2.5}R}{r\epsilon}$ and obtaining the following:

$$\begin{array}{ll} \text{minimize} & \langle \bar{\mathbf{c}}, \bar{\mathbf{x}} \rangle \\ \text{subject to} & \bar{\mathbf{A}}\bar{\mathbf{x}} = \bar{\mathbf{b}}, \\ & \bar{\mathbf{x}} \in \bar{\mathcal{K}} \stackrel{\text{def}}{=} \mathcal{K} \times \mathbb{R}^{m+n}_{\geq 0} \times \mathbb{R}^{m+n}_{\geq 0} \end{array} \tag{4.4.23}$$

with

$$\bar{\mathbf{A}} = [\mathbf{A} \mid \mathbf{A} \mid -\mathbf{A}], \bar{\mathbf{b}} = \mathbf{b}, \bar{\mathbf{c}} = (\mathbf{c}, \frac{\|\mathbf{c}\|_2 s}{\sqrt{m+n}} \cdot \mathbf{1}, \frac{\|\mathbf{c}\|_2 s}{\sqrt{m+n}} \cdot \mathbf{1})^\top$$

We solve the linear system $\mathbf{A}\mathbf{y} = \mathbf{b} - \mathbf{A}\mathbf{z}$ for $\mathbf{y}$. Then, we construct the initial $\bar{\mathbf{x}}$ by set $\bar{\mathbf{x}}^{(1)} = \mathbf{z}$,

$$\bar{\mathbf{x}}_i^{(2)} = \begin{cases} \mathbf{y}_i & \text{if } \mathbf{y}_i \geq 0, \\ 0 & \text{otherwise.} \end{cases} \quad \text{and} \quad \bar{\mathbf{x}}_i^{(3)} = \begin{cases} -\mathbf{y}_i & \text{if } \mathbf{y}_i < 0, \\ 0 & \text{otherwise.} \end{cases}$$

Then, we run Algorithm 4.3.1 on the Problem 4.4.23, with initial $\bar{\mathbf{x}}$ set above, $\bar{m} = 3(m+n), \bar{n} = n + 2, \bar{\epsilon} = \frac{\epsilon}{6\sqrt{ns}}, \overline{\mathcal{K}}_{\text{in}} = \{\mathbf{x}^{(1)} \in B(\mathbf{z}, \bar{r}), (\mathbf{x}^{(2)}, \mathbf{x}^{(3)}) \in \mathbb{R}^{2n}_{\geq 0}\}$ and $\widehat{\mathcal{K}}_{\text{out}} = \mathcal{B}_{\bar{m}}(\mathbf{0}, \sqrt{n}R)$.
By our choice of t_{end}, we have

$$\bar{t}_{\text{end}} = \frac{8\bar{m}}{\bar{\epsilon}\|\bar{\mathbf{c}}\|_2 \bar{R}} \leq \frac{48m}{\epsilon\|\mathbf{c}\|_2 R}.$$

First, we check the condition that $s \geq 48\bar{\nu}\bar{t}_{\text{end}} \sqrt{m+n} \frac{R^2}{r} \|\mathbf{c}\|_2$, we note that

$$48\bar{\nu}\bar{t}_{\text{end}} \sqrt{m+n} \frac{R^2}{r} \|\mathbf{c}\|_2 \leq 27648 \frac{m^{2.5}R}{\epsilon r} \leq 2^{16} \frac{m^{2.5}R}{r\epsilon} = s.$$

Let $\bar{\mathbf{x}}_{output} = (\mathbf{x}^{(1)}_{output}, \mathbf{x}^{(2)}_{output}, \mathbf{x}^{(3)}_{output})$ be the output of Algorithm 4.3.1. Then, let $\mathbf{x}_{output} = \mathbf{x}^{(1)}_{output} +$

$\mathbf{x}^{(2)}_{output} - \mathbf{x}^{(3)}_{output}$ as defined in Theorem 4.5.5. By Lemma 4.4.9, we have

$$\min_{\mathbf{x}\in\mathcal{P}_{\text{in}}} \bar{\mathbf{c}}^\top \bar{\mathbf{x}} \le \min_{\mathbf{x}\in\mathcal{P}} \mathbf{c}^\top \bar{\mathbf{x}} + \gamma$$

where $\gamma = \bar{\epsilon} \cdot \|\bar{\mathbf{c}}\|_2 \cdot \bar{R}$. Applying (3) of Theorem 4.5.5, we have

$$\mathbf{c}^\top \mathbf{x}_{output} \le \frac{\bar{\nu}+1}{\bar{t}_{\text{end}}} + \gamma + \min_{x\in\mathcal{K}, A\mathbf{x}=\mathbf{b}} \mathbf{c}^\top \mathbf{x} \le \min_{x\in\mathcal{K}, A\mathbf{x}=\mathbf{b}} \mathbf{c}^\top \mathbf{x} + \epsilon \cdot \|\mathbf{c}\|_2 \cdot R.$$

The last inequality follows by our choice of $\bar{\epsilon}$ and $\bar{t}_{\text{end}}$, we have $\gamma \le \frac{\epsilon}{2}\|\mathbf{c}\|_2 R$ and $\frac{\bar{\nu}+1}{\bar{t}_{\text{end}}} \le \frac{\epsilon}{2}\|\mathbf{c}\|_2 R$. Plug this $\bar{\epsilon}$ in Lemma 4.4.9, it gives the claimed oracle complexity.

□

Theorem 4.1.1 (Main Result)**.** *Consider Problem 1 and $\theta^{(0)}$ such that $\|\theta^\star - \theta^{(0)}\|_2 \le R$. Assuming all the f_i's are L-Lipschitz, then there is an algorithm that outputs a vector $\theta \in \mathbb{R}^d$ such that $\sum_{i=1}^n f_i(\theta) \le \sum_{i=1}^n f_i(\theta^\star) + \epsilon \cdot LR$, using $O(m \log(m/\epsilon))$ gradient oracle calls, in time* $\text{poly}(m \log(1/\epsilon))$.

Proof. First, we reformulate (4.1.1) using a change of variables and the epigraph trick. Suppose each f_i depends on d_i coordinates of θ given by $\{i_1, \ldots, i_{d_i}\} \subseteq [d]$. Then, symbolically define $\mathbf{x}_i = [x^{(i)}_{i_1}; x^{(i)}_{i_2}; \ldots; x^{(i)}_{i_{d_i}}] \in \mathbb{R}^{d_i}$ for each $i \in [n]$. Since each f_i is convex and supported on d_i variables, its epigraph is convex and $d_i + 1$ dimensional. So we may define the convex set

$$\mathcal{K}_i^{\text{unbounded}} = \left\{(\mathbf{x}_i, z_i) \in \mathbb{R}^{d_i+1} : f_i(\mathbf{x}_i) \le L z_i\right\}.$$

Finally, we add linear constraints of the form $x^{(i)}_k = x^{(j)}_k$ for all i, j, k where f_i and f_j both depend on θ_k. We denote these by the matrix constraint $\mathbf{A}\mathbf{x} = \mathbf{b}$. Then, Problem 1 is equivalent to

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^n L z_i \\ \text{subject to} & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & (\mathbf{x}_i, z_i) \in \mathcal{K}_i^{\text{unbounded}} \text{ for each } i \in [n]. \end{array} \tag{4.4.24}$$

Since we are given $\theta^{(0)}$ satisfying $\|\theta^{(0)} - \theta^*\|_2 \le R$, we define $\mathbf{x}_i^{(0)} = [\theta^{(0)}_{i_1}; \ldots, \theta^{(0)}_{i_{d_i}}]$ and $z_i^{(0)} = f_i(\theta^{(0)})/L$. Then, we can restrict the search space $\mathcal{K}_i^{\text{unbounded}}$ to

$$\mathcal{K}_i = \mathcal{K}_i^{\text{unbounded}} \cap \{(\mathbf{x}_i, z_i) \in \mathbb{R}^{d_i+1} : \|\mathbf{x}_i - \mathbf{x}_i^{(0)}\|_2 \le R \text{ and } z_i^{(0)} - 2R \le z_i \le z_i^{(0)} + 2R\}.$$

It's easy to check that $\mathcal{K}_i$ is contained in a ball of radius $5R$ centered at $(\mathbf{x}_i^{(0)}, z_i^{(0)})$, and contains a ball of radius R centered at $(\mathbf{x}_i^{(0)}, z_i^{(0)})$. The subgradient oracle for f_i translates to a separation oracle for $\mathcal{K}_i$. Then, we apply Theorem 4.4.10 to (4.4.24) with $\mathcal{K}_i^{\text{unbounded}}$ replaced by $\mathcal{K}_i$ to get the error guarantee and oracle complexity directly. □

Finally, we have the matching lower bound.

Theorem 4.1.2. *There exist functions $f_1, \ldots, f_n : \mathbb{R}^d \mapsto \mathbb{R}$ for which a total of $\Omega(m \log(1/\epsilon))$ gradient queries are required to solve Problem 1.*

Proof. [Nes04] shows that for any d_i, there exists $f_i : \mathbb{R}^{d_i} \mapsto \mathbb{R}$ for which $\Omega(d_i \log(1/\epsilon))$ total gradient queries are required. We define $f_1, \ldots, f_n$ to be such functions on disjoint coordinates of θ. It follows that $\Omega(\sum_{i=1}^n d_i \log(1/\epsilon)) = \Omega(m \log(1/\epsilon))$ gradient queries are required in total. □

4.5 Initialization

4.5.1 Constructing an initial $\mathcal{K}_{\text{in},i}$

In this section, we discuss how to construct an initial set $\mathcal{K}_{\text{in},i}$ to serve as an input to Algorithm 4.3.1. In particular, we will prove the following theorem.

Theorem 4.5.1. *Suppose we are given separation oracle access to a convex set $\mathcal{K}$ that satisfies $\mathcal{B}(\mathbf{z}, r) \subseteq \mathcal{K} \subseteq \mathcal{B}(\mathbf{0}, R)$ for some $\mathbf{z} \in \mathbb{R}^d$. Then, Algorithm 4.5.1, in $O(d \log(R/r))$ separation oracle calls to $\mathcal{K}$, outputs a point $\mathbf{x}$ such that $\mathcal{B}\left(\mathbf{x}, \frac{r}{6d^{3.5}}\right) \subseteq \mathcal{K}$.*

Algorithm 4.5.1 Inner Ball Finding

$\mathcal{K}_{\text{out}} \leftarrow B(0, R)$
while true do
  Let $\mathbf{v}$ be the center of gravity of $\mathcal{K}_{\text{out}}$
  Sample $\mathbf{u}$ from $B(\mathbf{v}, r/(6d))$ uniformly
  if $\mathbf{u} \in \mathcal{K}$ **then**
    Let $S = \{\mathbf{v} \pm \frac{r}{6d^3}\mathbf{e}_i : i \in [d]\}$
    if $S \subset \mathcal{K}$ **then**
      return the inscribed ball of conv(S)
    end if
  end if
  Let $\mathcal{K}_{\text{out}} \leftarrow \mathcal{K}_{\text{out}} \cap \mathcal{H}$ where $\mathcal{H} = \mathcal{O}(\mathbf{u})$
end while

Before we prove the preceding theorem, we need the following facts about the self-concordant barrier and convex sets.

Theorem 4.5.2 ([Nes04, Theorem 4.2.6])**.** *Let $\psi : int(\mathcal{K}) \to \mathbb{R}$ be a ν-self-concordant barrier with the minimizer $\mathbf{x}_\psi^\star$. Then, for any $\mathbf{x} \in int(\mathcal{K})$ we have:*

$$\|\mathbf{x}_\psi^\star - \mathbf{x}\|_{\mathbf{x}_\psi^\star} \leq \nu + 2\sqrt{\nu}.$$

On the other hand, for any $\mathbf{x} \in \mathbb{R}^d$ such that $\|\mathbf{x} - \mathbf{x}_\psi^\star\|_{\mathbf{x}_\psi^\star} \leq 1$, we have $x \in int(\mathcal{K})$.

Theorem 4.5.3 ([KLS95b, Theorem 4.1])**.** *Let $\mathcal{K} \subseteq \mathbb{R}^d$ be a convex set with center of gravity μ and covariance matrix Σ. Then,*

$$\{\mathbf{x} : \|\mathbf{x} - \mu\|_{\Sigma^{-1}} \leq \sqrt{(d+2)/d}\} \subseteq \mathcal{K} \subseteq \{\mathbf{x} : \|\mathbf{x} - \mu\|_{\Sigma^{-1}} \leq \sqrt{d(d+2)}\}.$$

Theorem 4.5.4 ([BGVV14, Section 1.4.2])**.** *Let $\mathcal{K}$ be a convex set with $\mathcal{K} \subset B(\mathbf{u}, R)$ for some R. Let $\mathcal{K}_{-\delta} = \{\mathbf{x} : B(\mathbf{x}, \delta) \subset \mathcal{K}\}$. Then, we have*

$$\text{vol}\mathcal{K}_{-\delta} \geq \text{vol}\mathcal{K} - (1 - (1 - \frac{\delta}{R})^d) \cdot \text{vol}B(\mathbf{u}, R)$$

Proof of Theorem 4.5.1. We note that by the description of the Algorithm 4.5.1, the returned ball is the inscribed ball of $\text{conv}(S)$ and we have $\mathbf{v} \in \mathcal{K}$ for each $\mathbf{v} \in S$. Then, we must have $\text{conv}(S) \subseteq \mathcal{K}$. We note that $\text{conv}(S)$ is a ℓ_1 ball with ℓ_1 radius $\frac{r}{6d^3}$, then the inscribed ball has ℓ_2 radius $\frac{r}{6d^{3.5}}$.

First, we prove the sample complexity of the algorithm above. We use $\mathcal{K}_t$ to denote the $\mathcal{K}_{\text{out}}$ at the t-th iteration. We first observe that throughout the algorithm, $\mathcal{K}_t$ is obtained by intersection of halfspaces and $B(0, R)$. This implies

$$B(\mathbf{z}, r) \subseteq \mathcal{K} \subseteq \mathcal{K}_t \qquad \forall t.$$

Since $\mathcal{K}_t$ contains a ball of radius r, let A_t be the covariance matrix of $\mathcal{K}_t$. By Theorem 5.3.7, we have

$$A_t \succeq \frac{r^2}{d(d+2)} I.$$

Let $\mathcal{H}_t$ be the halfspace returned by the oracle at iteration t. We note that $\mathbf{u}$ is sampled uniform from $B(\mathbf{v}, r/(6d))$, so we have

$$\|\mathbf{v} - \mathbf{u}\|_{A^{-1}} \leq \frac{\sqrt{d(d+2)}}{r} \cdot \frac{r}{6d} \leq \frac{1}{3}.$$

Apply the inequality above to Corollary 8, we have

$$\text{vol}(\mathcal{K}_t) \leq (1 - 1/e + 1/3)^t \text{vol}(\mathcal{K}_0) \leq (1 - 1/30)^t \text{vol}(B(0, R)).$$

Then, since $B(\mathbf{z}, r) \subseteq \mathcal{K}_t$ for all the t, this implies the algorithm at most takes $O(d \log(R/r))$ many iterations.

Now, we consider the number of oracle calls within each iterations. There are three possible cases to consider:

1. $\mathbf{u} \in \mathcal{K}_{-\delta}$ with $\delta = \frac{r}{6d^3}$ (see the definition of $\mathcal{K}_{-\delta}$ in Theorem 4.5.4). In this case, we have $S \subset \mathcal{K}$ and this is the last iteration. We can pay this $O(d)$ oracle calls for the last iteration.

2. $\mathbf{u} \in \mathcal{K} \backslash \mathcal{K}_{-\delta}$.

   Since $\mathbf{u}$ is uniformly sampled from $B(\mathbf{v}, r/(6d))$, Theorem 4.5.4 shows that $\mathbf{u} \in \mathcal{K} \backslash \mathcal{K}_{-\delta}$ with probability at most

   $$1 - (1 - \frac{\delta}{r/(6d)})^d \leq \frac{1}{d}.$$

   Hence, this case only happens with probability only at most $1/d$. Since the cost of checking $S \subset \mathcal{K}$ takes $O(d)$ oracle calls. The expected calls for this case is only $O(1)$.

3. $\mathbf{u} \notin K$. The cost is just 1 call.

Combining all the cases, the expected calls is $O(1)$ per iteration.

□

4.5.2 Initial point reduction

In this section, we will show how to obtain an initial feasible point for the algorithm.

Definition 10. Given a convex program $\min_{\mathbf{Ax}=\mathbf{b},\mathbf{x}\in\mathcal{K}\subseteq\mathbb{R}^d} \mathbf{c}^\top\mathbf{x}$ and some $s>0$, we define $\mathbf{c}_1=\mathbf{c},\mathbf{c}_2=\mathbf{c}_3=\frac{s\|\mathbf{c}\|_2}{\sqrt{d}}\cdot\mathbf{1}$ and $\mathcal{P}=\{\mathbf{x}^{(1)}\in\mathcal{K},(\mathbf{x}^{(2)},\mathbf{x}^{(3)})\in\mathbb{R}^{2d}_{\geq 0}:\mathbf{A}(\mathbf{x}^{(1)}+\mathbf{x}^{(2)}-\mathbf{x}^{(3)})=\mathbf{b}\}$. We then define the *modified convex program* by

$$\min_{(\mathbf{x}^{(1)},\mathbf{x}^{(2)},\mathbf{x}^{(3)})\in\mathcal{P}} \mathbf{c}_1^\top\mathbf{x}^{(1)}+\mathbf{c}_2^\top\mathbf{x}^{(2)}+\mathbf{c}_3^\top\mathbf{x}^{(3)}.$$

We denote $(\mathbf{c}_1,\mathbf{c}_2,\mathbf{c}_3)$ by $\bar{\mathbf{c}}$.

Theorem 4.5.5. *Given a convex program $\min_{\mathbf{Ax}=\mathbf{b},\mathbf{x}\in\mathcal{K}\subseteq\mathbb{R}^d}\mathbf{c}^\top\mathbf{x}$ with outer radius R and some convex set $\mathcal{K}_{in}$ with $\mathcal{K}_{in}\subseteq\mathcal{K}$ and inner radius r. For any modified convex program as in Definition 10 with $s\geq 48\nu t\sqrt{d}\cdot\frac{R}{r}\cdot\|c\|_2 R$. For an arbitrary $t\in\mathbb{R}_{\geq 0}$, we define the function*

$$f_t(\mathbf{x}^{(1)},\mathbf{x}^{(2)},\mathbf{x}^{(3)})=t(\mathbf{c}_1^\top\mathbf{x}^{(1)}+\mathbf{c}_2^\top\mathbf{x}^{(2)}+\mathbf{c}_3^\top\mathbf{x}^{(3)})+\psi_{\mathcal{P}_{in}}(\mathbf{x}^{(1)},\mathbf{x}^{(2)},\mathbf{x}^{(3)})$$

where $\psi_{\mathcal{P}_{in}}$ is some ν self-concordant barrier for the set

$$\mathcal{P}_{in}=\{\mathbf{x}^{(1)}\in\mathcal{K}_{in},(\mathbf{x}^{(2)},\mathbf{x}^{(3)})\in\mathbb{R}^{2d}_{\geq 0}:\mathbf{A}(\mathbf{x}^{(1)}+\mathbf{x}^{(2)}-\mathbf{x}^{(3)})=\mathbf{b}\}.$$

Given $\bar{\mathbf{x}}_t\overset{def}{=}(\mathbf{x}_t^{(1)},\mathbf{x}_t^{(2)},\mathbf{x}_t^{(3)})=\arg\min_{(\mathbf{x}^{(1)},\mathbf{x}^{(2)},\mathbf{x}^{(3)})\in\mathcal{P}_{in}} f_t(\mathbf{x}^{(1)},\mathbf{x}^{(2)},\mathbf{x}^{(3)})$, we denote $\mathbf{x}_{in}=\mathbf{x}_t^{(1)}+\mathbf{x}_t^{(2)}-\mathbf{x}_t^{(3)}$. Suppose $\min_{\bar{\mathbf{x}}\in\mathcal{P}_{in}}\bar{\mathbf{c}}^\top\bar{\mathbf{x}}\leq\min_{\bar{\mathbf{x}}\in\mathcal{P}}\bar{\mathbf{c}}^\top\bar{\mathbf{x}}+\gamma$, we have the following

1. $\mathbf{A}\mathbf{x}_{in}=\mathbf{b}$,
2. $\mathbf{x}_{in}\in\mathcal{K}_{in}$,
3. $\mathbf{c}^\top\mathbf{x}_{in}\leq\min_{\mathbf{x}\in\mathcal{K},\mathbf{Ax}=\mathbf{b}}\mathbf{c}^\top\mathbf{x}+\frac{\nu+1}{t}+\gamma$.

First, we show that $\mathbf{x}_t^{(1)}$ is not too close to the boundary. Before we proceed, we need the following lemmas.

Lemma 4.5.6 (Theorem 4.2.5 [Nes04])**.** *Let ψ be a ν-self-concordant barrier. Then, for any $\mathbf{x}\in dom(\psi)$ and $\mathbf{y}\in dom(\psi)$ such that*

$$\langle\psi'(\mathbf{x}),\mathbf{y}-\mathbf{x}\rangle\geq 0,$$

we have

$$\|\mathbf{y}-\mathbf{x}\|_{\mathbf{x}}\leq\nu+2\sqrt{\nu}.$$

Lemma 4.5.7 (Theorem 2 of [ZLY22])**.** *Given a convex set [5] Ω with a ν-self-concordant barrier ψ_Ω and inner radius r. Let $\mathbf{x}_t=\arg\min_{\mathbf{x}} t\cdot\mathbf{c}^\top\mathbf{x}+\psi_\Omega(\mathbf{x})$. Then, for any $t>0$,*

$$\min\left\{\frac{1}{2t},\frac{r\|c\|_2}{4\nu+4\sqrt{\nu}}\right\}\leq\mathbf{c}^\top\mathbf{x}_t-\mathbf{c}^\top\mathbf{x}_\infty\leq\frac{\nu}{t}.$$

Consider the optimization problem restricted in the subspace $\{(\mathbf{x}^{(1)},\mathbf{x}^{(2)},\mathbf{x}^{(3)}):A(\mathbf{x}^{(1)}+\mathbf{x}^{(2)}-\mathbf{x}^{(3)})=\mathbf{b}\}$, as a direct corollary of theorem above we have the following:

Corollary 11. *Let $\bar{\mathbf{x}}_t$ be as the same as defined in Theorem 4.5.5. For $t\geq\frac{4\nu}{r\|c\|_2}$, we have dist$(\mathbf{x}_t^{(1)},\mathbf{x}_\infty^{(1)})\geq\frac{1}{2t\|\mathbf{c}\|_2}$.*

Now, we are ready to show dist$(\mathbf{x}_t^{(1)},\partial\mathcal{K}_{\text{in}})$ is not too small.

[5] The original theorem is stated only for polytopes, but their proof works for general convex sets.

Theorem 4.5.8. *Let $\bar{\mathbf{x}}_t$ be the same as defined in Theorem 4.5.5. For $t \geq \frac{4\nu}{r\|c\|_2}$, we have dist$(\mathbf{x}_t^{(1)}, \partial\mathcal{K}_{in}) \geq \frac{r}{12\nu t\|\mathbf{c}\|_2 R}$.*

Proof. We consider the domain restricted in the subspace $\{(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)}) : \mathbf{A}(\mathbf{x}^{(1)} + \mathbf{x}^{(2)} - \mathbf{x}^{(3)}) = \mathbf{b}\}$. By the optimality of $\bar{\mathbf{x}}_t$ and Lemma 4.5.6, we have

$$\mathcal{K}_{\mathcal{H}} \subseteq \{\mathbf{x} : \|\mathbf{x} - \mathbf{x}_t^{(1)}\|_{\mathbf{x}_t^{(1)}} \leq \nu + 2\sqrt{\nu}\},$$

where $\mathcal{H} = \{\mathbf{x} : \mathbf{c}^\top(\mathbf{x}_t^{(1)} - \mathbf{x}) \geq 0\}$ and $\mathcal{K}_{\mathcal{H}} \stackrel{\text{def}}{=} \mathcal{H} \cap \mathcal{K}_{\text{in}}$.

Recall that $\mathcal{K}_{\text{in}}$ contains a ball of radius r, we denote it by B. We note that conv$(\mathbf{x}_\infty^{(1)}, B)$ is a union of a ball and a convex cone C with diameter at most $2R$. We observe that the set conv$(\mathbf{x}_\infty^{(1)}, B) \cap \mathcal{H}$ contains a ball of radius at least $\frac{r}{4t\|\mathbf{c}\|_2 R}$ since dist$(\mathbf{x}_\infty^{(1)}, \partial\mathcal{H}) \geq \frac{1}{2t\|\mathbf{c}\|_2}$.

We note that

$$\text{conv}(\mathbf{x}_\infty^{(1)}, B) \cap \mathcal{H} \subseteq \mathcal{K}_{\text{in}} \subseteq \{\mathbf{x} : \|\mathbf{x} - \mathbf{x}_t^{(1)}\|_{\mathbf{x}_t^{(1)}} \leq \nu + 2\sqrt{\nu}\},$$

this implies $\{\mathbf{x} : \|\mathbf{x} - \mathbf{x}_t^{(1)}\|_{\mathbf{x}_t^{(1)}} \leq \nu + 2\sqrt{\nu}\}$ contains a ball of radius at least $\frac{r}{4t\|\mathbf{c}\|_2 R}$, and then by Theorem 4.5.2, we have $B(\mathbf{x}_t^{(1)}, \frac{r}{4(\nu+2\sqrt{\nu})t\|\mathbf{c}\|_2 R}) \subseteq \mathcal{K}_{\text{in}}$. □

Lemma 4.5.9. *Let $(\mathbf{x}_t^{(1)}, \mathbf{x}_t^{(2)}, \mathbf{x}_t^{(3)}) \in \mathbb{R}^{3d}$ be the same as defined in Theorem 4.5.5. If $t > \frac{\nu}{\|\mathbf{c}\|_2 R}$, then we have $\|\mathbf{x}_t^{(2)} - \mathbf{x}_t^{(3)}\|_2 \leq \frac{4\sqrt{d}}{s} R$.*

Proof. Let $\mathbf{x}_{\text{in}}^\star = \arg\min_{\mathbf{x}\in\mathcal{K}_{\text{in}}, \mathbf{A}\mathbf{x}=\mathbf{b}} \mathbf{c}^\top\mathbf{x}$ and $\bar{\mathbf{x}}_{\text{in}}^\star = \arg\min_{\mathbf{x}\in\mathcal{P}_{\text{in}}} \bar{\mathbf{c}}^\top\bar{\mathbf{x}}$. Since $\mathbf{x}^\star \in \mathcal{B}(0, R)$, we have $\mathbf{c}^\top\mathbf{x}_{\text{in}}^\star \leq \|\mathbf{c}\|_2 R$. Note that $(\mathbf{x}_{\text{in}}^\star, \mathbf{0}, \mathbf{0}) \in \mathcal{P}_{\text{in}}$, this means we have $\bar{\mathbf{c}}^\top\bar{\mathbf{x}}_{\text{in}}^\star \leq \mathbf{c}^\top\mathbf{x}_{\text{in}}^\star \leq \|\mathbf{c}\|_2 R$. Combining this with the second inequality in Lemma 4.5.7, we get

$$\bar{\mathbf{c}}^\top\bar{\mathbf{x}}_t \leq \bar{\mathbf{c}}^\top\bar{\mathbf{x}}_{\text{in}}^\star + \frac{\nu}{t} \leq \|\mathbf{c}\|_2 R + \frac{\nu}{t} \leq 2\|\mathbf{c}\|_2 R.$$

We further note that

$$\mathbf{c}_2^\top\mathbf{x}_t^{(2)} \leq \bar{\mathbf{c}}^\top\bar{\mathbf{x}}_t \leq 2\|\mathbf{c}\| R.$$

This shows

$$\max\{\|\mathbf{x}_t^{(2)}\|_2, \|\mathbf{x}_t^{(3)}\|_2\} \leq \frac{2\sqrt{d}\|\mathbf{c}\|_2 R}{\|\mathbf{c}\|_2 s} \leq \frac{2\sqrt{d}R}{s}.$$

Hence, we have the claimed bound. □

Now, we are ready to prove Theorem 4.5.5.

Proof of Theorem 4.5.5. We note that $\mathbf{x}_{\text{in}}$ satisfies (1), directly follows by definition of $\mathcal{P}$. By assumption, we have $s \geq 48\nu t\sqrt{d} \cdot \frac{R}{r} \cdot \|c\|_2 R$; using this in Lemma 4.5.9, we have $\|\mathbf{x}_t^{(2)} - \mathbf{x}_t^{(3)}\|_2 \leq \frac{r}{12\nu t\|\mathbf{c}\|_2 R}$. This means $\mathbf{x}_{\text{in}} = \mathbf{x}_t^{(1)} + \mathbf{x}_t^{(2)} - \mathbf{x}_t^{(3)} \in \mathcal{K}_{\text{in}}$ since dist$(\mathbf{x}_t^{(1)}, \partial\mathcal{K}_{\text{in}}) \geq \frac{r}{12\nu t\|\mathbf{c}\|_2 R}$. Now, we show $\mathbf{c}^\top\mathbf{x}_{\text{in}}$ is close to $\mathbf{c}^\top\mathbf{x}^\star$.

Let $\mathbf{x}^\star = \arg\min_{\mathbf{x}\in\mathcal{K}, \mathbf{A}\mathbf{x}=\mathbf{b}} \mathbf{c}^\top\mathbf{x}$ and $\bar{\mathbf{x}}^\star = \arg\min_{\mathbf{x}\in\mathcal{P}} \bar{\mathbf{c}}^\top\bar{\mathbf{x}}$. By Lemma 4.5.7, we have

$$\bar{\mathbf{c}}^\top\bar{\mathbf{x}}_t - \frac{\nu}{t} \leq \bar{\mathbf{c}}^\top\bar{\mathbf{x}}_{\text{in}}^\star \leq \bar{\mathbf{c}}^\top\bar{\mathbf{x}}^\star + \gamma \leq \mathbf{c}^\top\mathbf{x}^\star + \gamma.$$

This implies

$$\mathbf{c}^\top \mathbf{x}_t^{(1)} \leq \bar{\mathbf{c}}^\top \bar{\mathbf{x}}_t \leq \mathbf{c}^\top \mathbf{x}^\star + \frac{\nu}{t} + \gamma.$$

We have

$$\mathbf{c}^\top \mathbf{x}_{\text{in}} = \mathbf{c}^\top (\mathbf{x}_t^{(1)} + \mathbf{x}_t^{(2)} - \mathbf{x}_t^{(3)}) \leq \mathbf{c}^\top \mathbf{x}^\star + \frac{\nu}{t} + \frac{4}{s}\|\mathbf{c}\|_2 R \leq \mathbf{c}^\top \mathbf{x}^\star + \frac{\nu+1}{t} + \gamma.$$

□

Chapter 5

A Gradient Sampling Algorithm for Lipschitz Functions in High and Low Dimensions

Zhang et al. introduced a novel modification of Goldstein's classical subgradient method, with an efficiency guarantee of $O(\varepsilon^{-4})$ for minimizing Lipschitz functions. Their work, however, makes use of a nonstandard subgradient oracle model and requires the function to be directionally differentiable. In this chapter, we show that both of these assumptions can be dropped by simply adding a small random perturbation in each step of their algorithm. The resulting method works on any Lipschitz function whose value and gradient can be evaluated at points of differentiability. We additionally present a new cutting plane algorithm that achieves better efficiency in low dimensions: $O(d\varepsilon^{-3})$ for Lipschitz functions and $O(d\varepsilon^{-2})$ for those that are weakly convex.

5.1 Introduction

The subgradient method [SKR85] is a classical procedure for minimizing a nonsmooth Lipschitz function f on $\mathbb{R}^d$. Starting from an initial iterate x_0, the method computes

$$x_{t+1} = x_t - \alpha_t v_t \text{ where } v_t \in \partial f(x_t). \tag{5.1.1}$$

Here, the positive sequence $\{\alpha_t\}_{t\geq 0}$ is user-specified, and the set ∂f is the *Clarke subdifferential* [Cla90, RW09],

$$\partial f(x) = \operatorname{conv}\left\{\lim_{i\to\infty} \nabla f(x_i) : x_i \to x,\ x_i \in \mathbf{dom}(\nabla f)\right\}.$$

In classical circumstances, the subdifferential reduces to familiar objects: for example, when f is C^1-smooth at x, the subdifferential $\partial f(x)$ comprises of only the gradient $\nabla f(x)$, while for convex functions, it reduces to the subdifferential in the sense of convex analysis.

The limit points $\bar{x}$ of the subgradient method are known to be first-order critical, meaning $0 \in \partial f(\bar{x})$, for functions f that are weakly convex — a broad class of functions first introduced in English in [Nur73]: a function f is ρ-weakly convex if the quadratically perturbed function $x \mapsto f(x) + \frac{\rho}{2}\|x\|^2$ is convex. In particular, convex and smooth functions are weakly convex [DD19]. Going beyond asymptotic guarantees, finite-time complexity estimates are known for smooth, convex, or weakly convex problems [GL13, RHS+16, JGN+17, AZ18, CDHS18, DDMP18, FLLZ18, ZXG18].

Modern machine learning, however, has witnessed the emergence of problems far beyond the weakly convex problem class. Indeed, tremendous empirical success has been recently powered by industry-backed solvers, such as Google's TensorFlow and Facebook's PyTorch, which routinely train nonsmooth nonconvex deep networks via (stochastic) subgradient methods. Despite a vast body of work on the asymptotic convergence of subgradient methods for nonsmooth nonconvex

problems [BHS05, Kiw07, MMM18, DDKL20, BP21], no finite-time nonasymptotic convergence rates were known outside the weakly convex setting until recently, with Zhang, Lin, Jegelka, Sra, and Jadbabaie [ZLSJ20] making a big leap forward towards this goal.

In particular, restricting themselves to the class of Lipschitz and directionally differentiable functions, [ZLSJ20] developed an efficient algorithm motivated by Goldstein's conceptual subgradient method [Gol77]. Moreover, this was recently complemented by [KS21] with lower bounds for finding *near*-approximate-stationary points for nonconvex nonsmooth functions.

While a significant breakthrough in both result and technique, one limitation of [ZLSJ20] is that their complexity guarantees and algorithm use a nonstandard first-order oracle whose validity is unclear in examples. To elaborate, their algorithm requires the following oracle access: given $x, u \in \mathbb{R}^d$ solve the *auxiliary convex feasibility problem*:

$$\text{find } g \in \partial f(x) \text{ subject to } \langle g, u \rangle = f'(x, u). \tag{5.1.2}$$

The first issue with this oracle is that no general recipe exists for representing the full subdifferential $\partial f(x)$ analytically, and evaluating even an arbitrary element of the subdifferential can be highly non-trivial [BLO02, KB13]. Moreover, $\partial f(x)$ could be a very complicated set, e.g., for a deep ReLU neural network, the subdifferential is a polyhedron with a potentially huge number of facets, making the complexity of (5.1.2) unclear.

Further, [ZLSJ20] claim that for a composition of directionally differentiable functions with a closed-form directional derivative for each function, we can find the desired g by the chain rule. While the chain rule does compute the directional derivative $f'(x, u)$, to the best of our knowledge, this does not translate to solving (5.1.2). This is owing to the crucial fact that the chain rule (and sum rule) can easily fail for the computation of the subdifferential[1] (although these are indeed valid for the directional derivative of a composition of directionally differentiable functions). We believe that this could potentially render the oracle of [ZLSJ20] computationally intractable.

Finally, we are unaware of other optimization algorithms imposing this oracle model. Therefore, at face value, the convergence guarantees of [ZLSJ20] are not comparable to those of others.

5.1.1 Our Results

Weakly convex optimization via a standard oracle. Our first contribution is to recover the complexity result of [ZLSJ20] under a much weaker assumption: specifically, we *replace the non-standard assumption in (5.1.2) with a standard first-order oracle model*. We show (Section 5.2) that a simple, yet critical, modification of the algorithm of [ZLSJ20], wherein one simply adds a small random perturbation in each iteration, works for any Lipschitz function assuming only an oracle that can compute gradients and function values at almost every point of $\mathbb{R}^d$ in the sense of Lebesgue measure. In particular, such oracles arise from automatic differentiation schemes routinely used in deep learning [BP20, BP21]. Our end result is a randomized algorithm for minimizing any L-Lipschitz function that outputs a (δ, ϵ)-stationary point (Definition 5.2.1) after using at most $\widetilde{O}\left(\frac{\Delta L^2}{\epsilon^3 \delta} \log(1/\gamma)\right)$ [2] gradient and function evaluations. Here Δ is the initial function gap and γ is the failure probability.

[1] We provide a simple example to demonstrate this claim: Consider the function $f(x, y) = f_1(x, y) + f_2(x, y)$ with $f_1(x, y) = |x|$ and $f_2(x, y) = -|x|$. Choose the direction $u = (0, 1)$, and let $z = (0, 0)$. Then, $f_1'(z, u) = f_2'(z, u) = 0$, and $\partial f_1(z) = \partial f_2(z) = [-1, 1] \times 0$. Therefore, to satisfy the oracle (5.1.2), for f_1, we may choose the subgradient $v_1 = (-1, 0)$, and for f_2, we may choose the subgradient $v_2 = (-1, 0)$ since $\langle v_1, u \rangle = 0 = \langle v_2, u \rangle$. However, $v_1 + v_2 = (-2, 0)$, which is not a subgradient of $f = 0$ at z.

[2] Throughout the chapter, we use $\widetilde{O}(\cdot)$ to hide poly-logarithmic factors in L, δ, Δ, and ϵ.

In light of the above modifications, our algorithm is implementable in the many important settings like deep neural networks that [ZLSJ20] is not. Along the way, we also simplify their proof techniques by providing a geometric viewpoint of the algorithm.

We would like to highlight the concurrent work of Tian, Zhou, and So [TZS22], which obtains this part of our result with a very similar technique. Additionally, we mention that such perturbation techniques have been widely used in the convex optimization literature, e.g., [Ber73, FKM04, DBW12].

Improved complexity in low dimensions. Having obtained the result of [ZLSJ20] within the standard first-order oracle model, we then proceed to investigate the following question.

*Can we improve the efficiency of the algorithm in **low dimensions**?*

In addition to being natural from the viewpoint of complexity theory, this question is well-grounded in applications. For instance, numerous problems in control theory involve minimization of highly irregular functions of a small number of variables. We refer the reader to the survey [BCL+20, Section 6] for an extensive list of examples, including Chebyshev approximation by exponential sums, spectral and pseudospectral abscissa minimization, maximization of the "distance to instability", and fixed-order controller design by static output feedback. We note that for many of these problems, the gradient sampling method of [BCL+20] is often used. Despite its ubiquity in applications, the gradient sampling method does not have finite-time efficiency guarantees. The algorithms we present here offer an alternative approach with a complete complexity theory.

The second contribution of this chapter is *an affirmative answer to the highlighted question*. We present a novel algorithm that uses $\widetilde{O}\left(\frac{\Delta L d}{\epsilon^2 \delta} \log(1/\gamma)\right)$ calls to our (weaker) oracle. Thus we are able to trade off the factor $L\epsilon^{-1}$ with d. Further, if the function is ρ-weakly convex, the complexity improves to $\widetilde{O}\left(\frac{\Delta d}{\epsilon \delta} \log(\rho)\right)$, which matches the complexity in $\delta = \epsilon$ of gradient descent for smooth minimization. Strikingly, the dependence on the weak convexity constant ρ is only logarithmic.

To put this contribution in perspective, assume for now $\delta = \epsilon$: then, our algorithm's dependence on ϵ in the case of Lipschitz, weakly convex functions is likely optimal in low dimensions, following a conjecture by Bubeck and Mikulincer [BM20] on the optimality of gradient descent for *smooth optimization* in dimension $d = \log\left(\frac{1}{\epsilon}\right)$ (thus matching the lower bound by Carmon, Duchi, Hinder, and Sidford [CDHS20]). Aside from possible optimality, the logarithmic dependence on smoothness/weak convexity exhibited by our iteration complexity is a significant improvement over the prior result of either $O(1/\epsilon^4)$ by [ZLSJ20] or Nemirovski and Yudin's rate of $O(1/\epsilon^2)$ with a polynomial dependence on smoothness. In the process, we also show that the minimal-norm element of the Goldstein-subdifferential in low dimensions can be found in time $O(\log(1/\epsilon))$, thus settling a question open since the 70s.

Techniques. The main idea for our improved dependence on ϵ in low dimensions is outlined next. The algorithm of [ZLSJ20] comprises of an outer loop with $O\left(\frac{\Delta}{\epsilon\delta}\right)$ iterations, each performing either a decrease in the function value or an ingenious random sampling step to update the descent direction. Our observation, central to improving the ε dependence, is that the violation of the descent condition can be transformed into a gradient oracle for the problem of finding a minimal norm element of the Goldstein subdifferential. This gradient oracle may then be used within a cutting plane method, which achieves better ε dependence at the price of a dimension factor (Section 5.3).

Limitations. One limitation of our work is that our second contribution does not immediately extend to the stochastic setting. We consider this to be an interesting open problem to resolve.

Notation. Throughout, we let $\mathbb{R}^d$ denote a d-dimensional Euclidean space equipped with a dot product $\langle \cdot, \cdot \rangle$ and the Euclidean norm $\|x\|_2 = \sqrt{\langle x, x \rangle}$. The symbol $\mathbb{B}_r(x)$ denotes an open Euclidean ball of radius $r > 0$ around a point x. Throughout, we fix a function $f \colon \mathbb{R}^d \to \mathbb{R}$ that is L-Lipschitz, and let $\mathbf{dom}(\nabla f)$ denote the set of points where f is differentiable—a full Lebesgue measure set by Rademacher's theorem. The symbol $f'(x,u) \stackrel{\text{def}}{=} \lim_{\tau \downarrow 0} \tau^{-1}(f(x+\tau u) - f(x))$ denotes the directional derivative of f at x in direction u, whenever the limit exists.

5.2 Interpolated Normalized Gradient Descent (INGD)

In this section, we describe the results in [ZLSJ20] and our modified subgradient method that achieves finite-time guarantees in obtaining (δ, ϵ)-stationarity for an L-Lipschitz function $f : \mathbb{R}^d \to \mathbb{R}$. The main construction we use is the Goldstein subdifferential [Gol77].

Definition 5.2.1 (Goldstein subdifferential)**.** *Consider a locally Lipschitz function $f \colon \mathbb{R}^d \to \mathbb{R}$, a point $x \in \mathbb{R}^d$, and a parameter $\delta > 0$. The* Goldstein subdifferential *of f at x is the set*

$$\partial_\delta f(x) \stackrel{def}{=} \operatorname{conv}\Big(\bigcup_{y \in \mathbb{B}_\delta(x)} \partial f(y) \Big).$$

A point x is called (δ, ϵ)-stationary if $\operatorname{dist}(0, \partial_\delta f(x)) \leq \epsilon$.

Thus, the Goldstein subdifferential of f at x is the convex hull of all Clarke subgradients at points in a δ-ball around x. Famously, [Gol77] showed that one can significantly decrease the value of f by taking a step in the direction of the minimal norm element of $\partial_\delta f(x)$. Throughout the rest of the section, we fix $\delta \in (0,1)$ and use the notation

$$\widehat{g} \stackrel{\text{def}}{=} g/\|g\|_2 \text{ for any nonzero vector } g \in \mathbb{R}^d. \tag{5.2.1}$$

Theorem 5.2.2 ([Gol77])**.** *Fix a point x, and let g be a minimal norm element of $\partial_\delta f(x)$. Then as long as $g \neq 0$, we have $f\left(x - \delta \widehat{g}\right) \leq f(x) - \delta \|g\|_2$.*

Theorem 5.2.2 immediately motivates the following conceptual descent algorithm:

$$x_{t+1} = x_t - \delta \widehat{g_t}, \text{ where } g_t \in \operatorname{argmin}_{g \in \partial_\delta f(x)} \|g\|_2. \tag{5.2.2}$$

In particular, Theorem 5.2.2 guarantees that, defining $\Delta \stackrel{\text{def}}{=} f(x_0) - \min f$, the *approximate stationarity condition*

$$\min_{t=1,\ldots,T} \|g_t\|_2 \leq \epsilon \text{ holds after } T = O\left(\frac{\Delta}{\delta \epsilon}\right) \text{ iterations of (5.2.2).}$$

We display this algorithmic framework in Algorithm 5.2.1.

Algorithm 5.2.1 Interpolated Normalized Gradient Descent (`INGD`(x_0, T))

Initial x_0, counter T
for $t = 0, \ldots, T-1$ **do**
  $g =$ `MinNorm`(x_t) ▷ Computational complexity $\widetilde{O}(L^2/\epsilon^2)$
  Set $x_{t+1} = x_t - \delta \widehat{g}$ ▷ We define $\widehat{g}$ in (5.2.1)
end for
Return x_T

The difficulty one encounters in trying to analyze Algorithm 5.2.1 is in the step $\texttt{MinNorm}(x_t)$: Evaluating the minimal norm element of $\partial_\delta f(x)$ is impossible in general, and therefore the descent method described in (5.2.2) cannot be applied directly. Nonetheless it serves as a guiding principle for implementable algorithms. Notably, the gradient sampling algorithm [BLO05] in each iteration forms polyhedral approximations K_t of $\partial_\delta f(x_t)$ by sampling gradients in the ball $\mathbb{B}_\delta(x)$ and computes search directions $g_t \in \operatorname{argmin}_{g \in K_t} \|g\|_2$. These gradient sampling algorithms, however, have only asymptotic convergence guarantees [BCL+20].

The recent paper [ZLSJ20], operating in the framework of the conceptual Goldstein descent algorithm (Algorithm 5.2.1), remarkably shows that for any $x \in \mathbb{R}^d$ one *can* find an *approximate* minimal norm element of $\partial_\delta f(x)$ using a number of subgradient computations that is independent of the dimension. We display [ZLSJ20]'s algorithm to find a minimal norm element of the Goldstein subdifferential in Algorithm 5.2.2.

Algorithm 5.2.2 $\texttt{MinNorm}(x, \delta, \epsilon)$ of [ZLSJ20]

Input. x, $\delta > 0$, $\epsilon > 0$, and access to the oracle $\mathbb{O}(x, d)$ defined in Assumption 1 of [ZLSJ20].
Let $k = 0$, $g_0 = \mathbb{O}(x, \mathbf{0})$.
while $\|g_k\|_2 > \epsilon$ and $\frac{\delta}{4}\|g_k\|_2 \geq f(x) - f\left(x - \delta\widehat{g_k}\right)$ **do**
    Choose y_k uniformly at random from the segment $[x, x - \delta\widehat{g_k}]$.
    $u_k = \mathbb{O}(y_k, -g_k)$.
    $g_{k+1} = \operatorname{argmin}_{z \in [g_k, u_k)]} \|z\|_2$.
    $k = k + 1$.
end while
Return g_k.

The key idea underlying Algorithm 5.2.2 is as follows. Suppose that we have a trial vector $g \in \partial_\delta f(x)$ (not necessarily a minimal norm element) satisfying

$$f\left(x - \delta\widehat{g}\right) \geq f(x) - \frac{\delta}{2}\|g\|_2. \tag{5.2.3}$$

That is, the decrease in function value is not as large as guaranteed by Theorem 5.2.2 for the true minimal norm subgradient. One would like to now find a vector $u \in \partial_\delta f(x)$ so that the norm of some convex combination $(1 - \lambda)g + \lambda u$ is smaller than that of g. A short computation shows that this is sure to be the case for all small $\lambda > 0$ as long as $\langle u, g\rangle \leq \|g\|_2^2$. The task therefore reduces to:

$$\text{find some } u \in \partial_\delta f(x) \quad \text{satisfying} \quad \langle u, g\rangle \leq \|g\|_2^2.$$

The ingenious idea of [ZLSJ20] is a randomized procedure for establishing exactly that in expectation. Namely, suppose for the moment that f happens to be differentiable along the segment $[x, x - \delta\widehat{g}]$; we will revisit this assumption shortly. Then the fundamental theorem of calculus, in conjunction with (5.2.3), yields

$$\frac{1}{2}\|g\|_2 \geq \frac{f(x) - f\left(x - \delta\widehat{g}\right)}{\delta} = \frac{1}{\delta}\int_0^\delta \langle \nabla f(x - \tau\widehat{g}), \widehat{g}\rangle \, d\tau. \tag{5.2.4}$$

Consequently, a point y chosen uniformly at random in the segment $[x, x - \delta\widehat{g}]$ satisfies

$$\mathbb{E}\langle \nabla f(y), g\rangle \leq \frac{1}{2}\|g\|_2^2. \tag{5.2.5}$$

Therefore the vector $u = \nabla f(y)$ can act as the subgradient we seek. Indeed, the following lemma shows that, in expectation, the minimal norm element of $[g, u]$ is significantly shorter than g. The proof is extracted from that of [ZLSJ20, Theorem 8].

Lemma 5.2.3 ([ZLSJ20]). *Fix a vector $g \in \mathbb{R}^d$, and let $u \in \mathbb{R}^d$ be a random vector satisfying $\mathbb{E}\langle u, g\rangle < \frac{1}{2}\|g\|_2^2$. Suppose moreover that the inequality $\|g\|_2, \|u\|_2 \le L$ holds for some $L < \infty$. Then the minimal-norm vector z in the segment $[g, u]$ satisfies:*

$$\mathbb{E}\|z\|_2^2 \le \|g\|_2^2 - \frac{\|g\|_2^4}{16L^2}.$$

Proof. Applying $\mathbb{E}\langle u, g\rangle \le \frac{1}{2}\|g\|_2^2$ and $\|g\|_2, \|u\|_2 \le L$, we have, for any $\lambda \in (0, 1)$,

$$\begin{aligned}\mathbb{E}\|z\|_2^2 \le \mathbb{E}\|g + \lambda(u - g)\|_2^2 &= \|g\|_2^2 + 2\lambda\mathbb{E}\langle g, u - g\rangle + \lambda^2\mathbb{E}\|u - g\|_2^2 \\ &\le \|g\|_2^2 - \lambda\|g\|_2^2 + 4\lambda^2L^2.\end{aligned}$$

Plugging in the value $\lambda = \frac{\|g\|_2^2}{8L^2} \in (0, 1)$ minimizes the right hand side and completes the proof. □

The last technical difficulty to overcome is the requirement that f be differentiable along the line segment $[g, u]$. This assumption is crucially used to obtain (5.2.4) and (5.2.5). To cope with this problem, [ZLSJ20] introduce extra assumptions on the function f to be minimized and assume a nonstandard oracle access to subgradients.

5.2.1 Our Algorithm for Computing the Minimal Norm Element.

Our first contribution is to show, using Lemma 5.2.4, that no extra assumptions are needed if one slightly perturbs g.

Lemma 5.2.4. *Let $f\colon \mathbb{R}^d \to \mathbb{R}$ be a Lipschitz function, and fix a point $x \in \mathbb{R}^d$. Then there exists a set $\mathcal{D} \subset \mathbb{R}^d$ of full Lebesgue measure such that for every $y \in \mathcal{D}$, the line spanned by x and y intersects $\mathbf{dom}(\nabla f)$ in a full Lebesgue measure set in $\mathbb{R}$. Then, for every $y \in \mathcal{D}$ and all $\tau \in \mathbb{R}$, we have*

$$f(x + \tau(y - x)) - f(x) = \int_0^\tau \langle \nabla f(x + s(y - x)), y - x\rangle \, ds.$$

Proof. Without loss of generality, we may assume $x = 0$ and $f(x) = 0$. Rademacher's theorem guarantees that $\mathbf{dom}(\nabla f)$ has full Lebesgue measure in $\mathbb{R}^d$. Fubini's theorem then directly implies that there exists a set $Q \subset \mathbb{S}^{d-1}$ of full Lebesgue measure within the sphere $\mathbb{S}^{d-1}$ such that for every $y \in Q$, the intersection $\mathbb{R}_+\{y\} \cap (\mathbf{dom}(\nabla f))^c$ is Lebesgue null in $\mathbb{R}$. It follows immediately that the set $\mathcal{D} = \{\tau y : \tau > 0, y \in Q\}$ has full Lebesgue measure in $\mathbb{R}^d$. Fix now a point $y \in \mathcal{D}$ and any $\tau \in \mathbb{R}_+$. Since f is Lipschitz, it is absolutely continuous on any line segment and therefore

$$f(x + \tau(y - x)) - f(x) = \int_0^\tau f'(x + s(y - x), y - x) \, ds = \int_0^\tau \langle \nabla f(x + s(y - x)), y - x\rangle \, ds.$$

□

We now have all the ingredients to present a modification of the algorithm from [ZLSJ20], which, under a standard first-order oracle model, either significantly decreases the objective value or finds an approximate minimal norm element of $\partial_\delta f$. As one can see comparing with Algorithm 5.2.2, the difference is in how we compute y_k and the associated assumptions [ZLSJ20] needs to impose.

Algorithm 5.2.3 `MinNorm`(x, δ, ϵ)

Input. x, $\delta > 0$, and $\epsilon > 0$.
Let $k = 0$, $g_0 = \nabla f(\zeta_0)$ where $\zeta_0 \sim \mathbb{B}_\delta(x)$.
while $\|g_k\|_2 > \epsilon$ and $\frac{\delta}{4}\|g_k\|_2 \geq f(x) - f\left(x - \delta\widehat{g_k}\right)$ **do**
    Choose any r satisfying $0 < r < \|g_k\|_2 \cdot \sqrt{1 - (1 - \frac{\|g_k\|_2^2}{128L^2})^2}$.
    Sample ζ_k uniformly from $\mathbb{B}_r(g_k)$.
    Choose y_k uniformly at random from the segment $[x, x - \delta\widehat{\zeta_k}]$.
    Set $u_k = \nabla f(y_k)$.
    $g_{k+1} = \operatorname{argmin}_{z \in [g_k, u_k]} \|z\|_2$.
    $k = k + 1$.
end while
Return g_k.

The following theorem establishes the efficiency of Algorithm 5.2.3, and its proof is similar to that of [ZLSJ20, Lemma 13]. For completeness, we include the full proof in Section D.1.

Theorem 5.2.5. *Let $\{g_k\}$ be generated by* `MinNorm`(x, δ, ϵ). *Fix an index $k \geq 0$, and define the stopping time $\tau \stackrel{def}{=} \inf\{k\colon f(x - \delta\widehat{g_k}) < f(x) - \delta\|g_k\|_2/4$ or $\|g_k\|_2 \leq \epsilon\}$. Then, we have*

$$\mathbb{E}\left[\|g_k\|_2^2 1_{\tau > k}\right] \leq \frac{16L^2}{16 + k}.$$

An immediate consequence of Theorem 5.2.5 is that `MinNorm`(x, δ, ϵ) terminates with high-probability.

Corollary 12. `MinNorm`(x, δ, ϵ) *terminates in at most $\left\lceil \frac{64L^2}{\epsilon^2} \right\rceil \cdot \lceil 2\log(1/\gamma) \rceil$ iterations with probability at least $1 - \gamma$, where we define the stopping time $\tau \stackrel{def}{=} \inf\{k\colon f(x - \delta\widehat{g_k}) < f(x) - \delta\|g_k\|_2/4$ or $\|g_k\|_2 \leq \epsilon\}$.*

Incorporating Algorithm 5.2.3 in Algorithm 5.2.1 yields convergence guarantees summarized in Theorem 5.2.6, whose proof is identical to that of [ZLSJ20, Theorem 8].

Theorem 5.2.6. *Fix an initial point $x_0 \in \mathbb{R}^d$, and define $\Delta = f(x_0) - \inf_x f(x)$. Set the number of iterations $T = \frac{4\Delta}{\delta\epsilon}$. Then, with probability $1 - \gamma$, the point $x_T =$* `INGD`(x_0, T) *satisfies dist$(0, \partial_\delta f(x_T)) \leq \epsilon$ in a total of at most*

$$\left\lceil \frac{4\Delta}{\delta\epsilon} \right\rceil \cdot \left\lceil \frac{64L^2}{\epsilon^2} \right\rceil \cdot \left\lceil 2\log\left(\frac{4\Delta}{\gamma\delta\epsilon}\right) \right\rceil \qquad \textit{function-value and gradient evaluations.}$$

Discussion. In summary, the complexity of finding a point x satisfying $\operatorname{dist}(0, \partial_\delta f(x)) \leq \epsilon$ is at most $O\left(\frac{\Delta L^2}{\delta\epsilon^3}\log\left(\frac{4\Delta}{\gamma\delta\epsilon}\right)\right)$ with probability $1 - \gamma$. Using the identity $\partial f(x) = \limsup_{\delta \to 0} \partial_\delta f(x)$, this result also provides a strategy for finding a Clarke stationary point, albeit with no complexity guarantee. It is thus natural to ask whether one may efficiently find some point x for which there exists $y \in \mathbb{B}_\delta(x)$ satisfying $\operatorname{dist}(0, \partial f(y)) \leq \epsilon$. This is exactly the guarantee of subgradient methods on weakly convex functions in [DD19]. [Sha20] shows that for general Lipschitz functions, the number of subgradient computations required to achieve this goal by any algorithm scales with the dimension of the ambient space. The perturbation technique of this section similarly applies to the stochastic algorithm of [ZLSJ20, Algorithm 2], yielding a method that matches their complexity estimate. This stochastic setting recently saw tremendous progress in the work of [CMO23], who

obtained the optimal rate of $O(\Delta L^2/\delta\epsilon^2)$ via a clever reduction to online learning. We also remark that a simple, but clever, modification of Algorithm 5.2.3 combined with Algorithm D.2.1 was done in [KS22] to obtain a matching complexity $(O\left(\frac{\Delta L^2 \log(H\delta/\epsilon)}{\delta\epsilon^3}\right))$ for finding a (δ,ϵ)-stationary point of an L-Lipschitz, H-smooth function via a *deterministic* algorithm (among several other results).

5.3 Faster INGD in Low Dimensions

In this section, we describe our modification of Algorithm 5.2.3 ("INGD") for obtaining improved runtimes in the low-dimensional setting. Our modified algorithm hinges on computations similar to (5.2.3), (5.2.4), and (5.2.5) except for the constants involved, and hence we explicitly state this setup. Given a vector $g \in \partial_\delta f(x)$, we say it satisfies the *descent condition* at x if

$$f(x - \delta\widehat{g}) \le f(x) - \frac{\delta\epsilon}{3}. \tag{5.3.1}$$

Recall that Lemma 5.2.4 shows that for almost all g, we have

$$f(x) - f(x - \delta\widehat{g}) = \int_0^1 \langle \nabla f(x - t\delta\widehat{g}), \widehat{g}\rangle \, dt = \delta \cdot \mathbb{E}_{z\sim \mathrm{Unif}[x-\delta\widehat{g},x]} \langle \nabla f(z), \widehat{g}\rangle.$$

Hence, when g does *not* satisfy the descent condition (5.3.1), we can output a random vector $u \in \partial_\delta f(x)$ such that

$$\mathbb{E}\langle u, g\rangle \le \frac{\epsilon}{3}\|g\|_2. \tag{5.3.2}$$

Then, an arbitrary vector g either satisfies (5.3.1) or can be used to output a random vector u satisfying (5.3.2). As described in Corollary 12, Algorithm 5.2.3 achieves this goal in $\widetilde{O}(L^2/\epsilon^2)$ iterations.

In this section, we improve upon this oracle complexity by applying cutting plane methods (which we review shortly) to design Algorithm 5.3.1, which finds a better descent direction in $\widetilde{O}(Ld/\epsilon)$ oracle calls for L-Lipschitz functions and $O(d\log(L/\epsilon)\log(\delta\rho/\epsilon))$ oracle calls for ρ-weakly convex functions.

Brief overview of cutting-plane methods. We first provide a brief relevant overview of cutting-plane methods here and refer the reader to standard textbooks in optimization for a more in-depth exposition. Given a convex function f with its set $\mathcal{S}$ of minimizers, a cutting-plane method (CPM) minimizes f by maintaining a convex search set $\mathcal{E}^{(k)} \supseteq \mathcal{S}$ in the k^{th} iteration and iteratively shrinking $\mathcal{E}^{(k)}$ guided by the subgradients of f that act as "separation oracles" for the set $\mathcal{S}$. Specifically, this is achieved by noting that for any $x^{(k)}$ chosen from $\mathcal{E}^{(k)}$, if the gradient oracle indicates $\nabla f(x^{(k)}) \neq 0$, (i.e. $x^{(k)} \notin \mathcal{S}$), then the convexity of f guarantees $\mathcal{S} \subseteq \mathcal{H}^{(k)} : \left\{y : \langle \nabla f(x^{(k)}), y - x^{(k)}\rangle \le 0\right\}$, and hence $\mathcal{S} \subseteq \mathcal{H}^{(k)} \cap \mathcal{E}^{(k)}$. The algorithm continues by choosing $\mathcal{E}^{(k+1)} \supseteq \mathcal{E}^{(k)} \cap \mathcal{H}^{(k)}$, and different choices of $x^{(k)}$ and $\mathcal{E}^{(k)}$ yield different rates of shrinkage of $\mathcal{E}^{(k)}$ until a point in $\mathcal{S}$ is found.

In light of this description, the minimization of a convex function over a constrained convex set via this cutting-plane method requires, at each iteration, merely a subgradient of the function. Our novel insight is that a lack of function decrease implies we have roughly such a subgradient, which we may then use in a cutting-plane method for computing the minimum norm element of the subdifferential faster in low dimensions (with improved complexity for weakly convex functions).

Setting the stage for our algorithm. In Section D.2, we demonstrate how to remove the expectation in (5.3.2) and turn the inequality into a high probability statement. For now, we assume the existence

of an oracle $\mathscr{O}$ as in Definition 5.3.1.

Definition 5.3.1 (Inner Product Oracle)**.** *Given a vector $g \in \partial_\delta f(x)$ that does not satisfy the descent condition (5.3.1), the inner product oracle $\mathscr{O}(g)$ outputs a vector $u \in \partial_\delta f(x)$ such that*

$$\langle u, g\rangle \le \frac{\epsilon}{2}\|g\|_2.$$

We defer the proof of the lemma below to Section D.2.

Lemma 5.3.2. *Fix $x \in \mathbb{R}^d$ and a unit vector $\widehat{g} \in \mathbb{R}^d$ such that f is differentiable almost everywhere on the line segment $[x, y]$, where $y \stackrel{def}{=} x - \delta\widehat{g}$. Suppose that $z \in \mathbb{R}^d$ sampled uniformly from $[x, y]$ satisfies $\mathbb{E}_z\langle\nabla f(z), \widehat{g}\rangle \le \frac{\epsilon}{3}$. Then we can find $\bar{z} \in \mathbb{R}^d$ using at most $O(\frac{L}{\epsilon}\log(1/\gamma))$ gradient evaluations of f, such that with probability at least $1-\gamma$ the estimate $\langle\nabla f(\bar{z}), \widehat{g}\rangle \le \frac{\epsilon}{2}$ holds. Moreover, if f is ρ-weakly convex, we can find $\bar{z} \in \mathbb{R}^d$ such that $\langle\nabla f(\bar{z}), \widehat{g}\rangle \le \frac{\epsilon}{2}$ using only $O(\log(\delta\rho/\epsilon))$ function evaluations of f.*

Our key insight is that this oracle is almost identical to the gradient oracle of the minimal norm element problem

$$\min_{g\in\partial_\delta f(x)} \|g\|_2.$$

Therefore, we can use it in the cutting plane method to find an approximate minimal norm element of $\partial_\delta f$. When there is no element of $\partial_\delta f$ with norm less than ϵ, our algorithm will instead find a vector that satisfies the descent condition. The main result of this section is the following theorem.

Theorem 5.3.3. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be an L-Lipschitz function. Fix an initial point $x_0 \in \mathbb{R}^d$, and let $\Delta \stackrel{def}{=} f(x_0) - \inf_x f(x)$. Then, there exists an algorithm that outputs a point $x \in \mathbb{R}^d$ satisfying $dist(0, \partial_\delta f(x)) \le \epsilon$ and, with probability at least $1-\gamma$, uses at most*

$$O\left(\frac{\Delta L d}{\delta\epsilon^2}\cdot\log(L/\epsilon)\cdot\log(1/\gamma)\right) \qquad \textit{function value/gradient evaluations.}$$

If f is ρ-weakly convex, the analogous statement holds with probability one and with the improved efficiency estimate $O\left(\frac{\Delta d}{\delta\epsilon}\log(L/\epsilon)\cdot\log(\delta\rho/\epsilon)\right)$ of function value/gradient evaluations.

5.3.1 Finding a Minimal Norm Element

In this section, we show, via Algorithm 5.3.1, how to find an approximate minimal norm element of $\partial_\delta f(x)$. Instead of directly working with the minimal norm problem, we note that, by Cauchy-Schwarz inequality and the Minimax Theorem, for any closed convex set Q, we have

$$\min_{g\in Q}\|g\|_2 = \min_{g\in Q}\left[\max_{\|v\|_2\le 1}\langle g, v\rangle\right] = \max_{\|v\|_2\le 1}\left[\min_{g\in Q}\langle g, v\rangle\right] = \max_{\|v\|_2\le 1}\phi_Q(v), \tag{5.3.3}$$

where $\phi_Q(v) \stackrel{\text{def}}{=} \min_{g\in Q}\langle g, v\rangle$, and Lemma 5.3.4 formally connects the problem of finding the minimal norm element with that of maximizing ϕ_Q. The key observation in this section (Lemma 5.3.5) is that the inner product oracle $\mathscr{O}$ is a separation oracle for the (dual) problem $\max_{\|v\|_2\le 1}\phi_Q(v)$ with $Q = \partial_\delta f(x)$ and hence can be used in cutting plane methods.

Lemma 5.3.4. *Let $Q \subset \mathbb{R}^d$ be a closed convex set that does not contain the origin. Let g_Q^* be a minimizer of*

$\min_{g \in Q} \|g\|_2$. *Then, the vector* $v_Q^* = g_Q^*/\|g_Q^*\|_2$ *satisfies*

$$\langle v_Q^*, g \rangle \geq \|g_Q^*\|_2 \qquad \text{for all } g \in Q.$$

and $v_Q^* = \arg\max_{\|v\|_2 \leq 1} \phi_Q(v)$.

Proof. We omit the subscript Q to simplify notation. Since, by definition, g^* minimizes $\|g\|_2$ over all $g \in Q$, we have

$$\langle g^*, g \rangle \geq \|g^*\|_2^2 \text{ for all } g \in Q,$$

and the inequality is tight for $g = g^*$. Using this fact and $\phi(v^*) = \min_{g \in Q} \langle g, \frac{g^*}{\|g^*\|_2} \rangle$ gives

$$\phi(v^*) = \|g^*\|_2 = \min_{g \in Q} \|g\|_2 = \min_{g \in Q} \max_{v : \|v\|_2 \leq 1} \langle g, v \rangle = \max_{\|v\|_2 \leq 1} \min_{g \in Q} \langle g, v \rangle = \max_{v : \|v\|_2 \leq 1} \phi(v),$$

where we used Sion's minimax theorem in the second to last step. This completes the proof. □

Using this lemma, we can show that $\mathcal{O}$ is a separation oracle.

Lemma 5.3.5. *Consider a vector* $g \in \partial f_\delta(x)$ *that does not satisfy the descent condition* (5.3.1)*, and let the output of querying the oracle at* g *be* $u \in \mathcal{O}(g)$*. Suppose that* $dist(0, \partial_\delta f(x)) \geq \frac{\epsilon}{2}$*. Let* g^* *be the minimal-norm element of* $\partial_\delta f(x)$*. Then the normalized vector* $v^* \stackrel{def}{=} g^*/\|g^*\|_2$ *satisfies the inclusion:*

$$v^* \in \left\{ w \in \mathbb{R}^d : \langle u, \widehat{g} - w \rangle \leq 0 \right\}.$$

Proof. Set $Q = \partial_\delta f(x)$. By using $\langle u, \widehat{g} \rangle \leq \frac{\epsilon}{2}$ (the guarantee of $\mathcal{O}$ per Definition 5.3.1) and $\langle u, v^* \rangle \geq \|g^*\|_2$ (from Lemma 5.3.4), we have $\langle u, \widehat{g} - v^* \rangle = \langle u, \widehat{g} \rangle - \langle u, v^* \rangle \leq \frac{\epsilon}{2} - \|g^*\|_2 \leq 0$. □

Thus Lemma 5.3.5 states that if x is not a $(\delta, \frac{\epsilon}{2})$-stationary point of f, then the oracle $\mathcal{O}$ produces a halfspace $\mathcal{H}_v$ that separates $\widehat{g}$ from v^*. Since $\mathcal{O}$ is a separation oracle, we can combine it with any cutting plane method to find v^*. For concreteness, we use the center of gravity method and display our algorithm in Algorithm 5.3.1. We note that Ω_k is an intersection of a ball and some half spaces, hence we can compute its center of gravity in polynomial time by taking an average of the empirical samples from this convex set. While we use a simple cutting-plane method, any algorithm in this class may be used; our focus is merely on minimizing the oracle complexity. Further note that in our algorithm, we use a point ζ_k close to the true center of gravity of Ω_k, and therefore, we invoke a result about the *perturbed* center of gravity method.

Theorem 5.3.6 (Theorem 3 of [BV04a]; see also [Grü60])**.** *Let* K *be a convex set with center of gravity* μ *and covariance matrix* A*. For any halfspace* H *that contains some point* x *with* $\|x - \mu\|_{A^{-1}} \leq t$*, we have*

$$\text{vol}(K \cap H) \leq (1 - 1/e + t)\text{vol}(K).$$

Theorem 5.3.7 (Theorem 4.1 of [KLS95a])**.** *Let* K *be a convex set in* $\mathbb{R}^d$ *with center of gravity* μ *and covariance matrix* A*. Then,*

$$K \subset \left\{ x : \|x - \mu\|_{A^{-1}} \leq \sqrt{d(d+2)} \right\}.$$

We now have all the tools to show correctness and iteration complexity of Algorithm 5.3.1.

Algorithm 5.3.1 `MinNormCG`(x)

Initialize center point x.
Set $k = 0$, the search region $\Omega_0 = \mathbb{B}_2(0)$, the set of gradients $Q_0 = \{\nabla f(x)\}$, and r satisfying $0 < r < \epsilon/(32dL)$
while $\min_{g \in Q_k} \|g\|_2 > \epsilon$ **do**
    Let v_k be the center of gravity of Ω_k.
    if v_k satisfies the descent condition (5.3.1) at x **then**
        Return v_k
    end if
    Sample ζ_k uniformly from $\mathbb{B}_r(v_k)$
    $u_k \leftarrow \mathscr{O}(\zeta_k)$
    $\Omega_{k+1} = \Omega_k \cap \{w : \langle u_k, \zeta_k - w \rangle \leq 0\}$.
    $Q_{k+1} = \mathrm{conv}(Q_k \cup \{u_k\})$
    $k = k + 1$
end while
Return $\arg\min_{g \in Q_k} \|g\|_2$.

Theorem 5.3.8. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be an L-Lipschitz function. Then Algorithm 5.3.1 returns a vector $v \in \partial_\delta f(x)$ that either satisfies the descent condition (5.3.1) at x or satisfies $\|v\|_2 \leq \epsilon$ in*

$$\lceil 8d \log(8L/\epsilon) \rceil \text{ calls to } \mathscr{O}.$$

Proof. By the description of Algorithm 5.3.1, it returns either a vector v satisfying the descent condition or $g \in \partial_\delta f(x)$ with $\|g\|_2 \leq \epsilon$. We now obtain the algorithm's claimed iteration complexity. Consider an iteration k such that Ω_k *does* contain a ball of radius $\frac{\epsilon}{4L}$. Let A_k be the covariance matrix of convex set Ω_k. By Theorem 5.3.7, we have

$$A_k \succeq \left(\frac{\epsilon}{8dL}\right)^2 I.$$

Applying this result to the observation that in Algorithm 5.3.1 ζ_k is sampled uniformly from $\mathbb{B}_r(v_k)$ gives

$$\|v_k - \zeta_k\|_{A_k^{-1}} \leq r \cdot \frac{8dL}{\epsilon} \leq \frac{1}{4}. \tag{5.3.4}$$

Recall from Algorithm 5.3.1 and the preceding notation that Ω_k has center of gravity v_k and covariance matrix A_k. Further, the halfspace $\{w : \langle u_k, \zeta_k - w \rangle \leq 0\}$ in Algorithm 5.3.1 contains the point ζ_k satisfying (5.3.4). Given these statements, since Algorithm 5.3.1 sets $\Omega_{k+1} = \Omega_k \cap \{w : \langle u_k, \zeta_k - w \rangle\}$, we may invoke Theorem 5.3.6 to obtain

$$\mathrm{vol}(\Omega_k) \leq (1 - 1/e + 1/4)^k \mathrm{vol}(\mathbb{B}_2(0)) \leq (1 - 1/10)^k \mathrm{vol}(\mathbb{B}_2(0)). \tag{5.3.5}$$

We claim that Algorithm 5.3.1 takes at most $T + 1$ steps where $T = d \log_{(1-\frac{1}{10})}(\epsilon/(8L))$. For the sake of contradiction, suppose that this statement is false. Then, applying (5.3.5) with $k = T + 1$ gives

$$\mathrm{vol}(\Omega_{T+1}) \leq \left(\frac{\epsilon}{4L}\right)^d \mathrm{vol}(\mathbb{B}_1(0)). \tag{5.3.6}$$

On the other hand, Algorithm 5.3.1 generates points $u_i = \mathscr{O}(\zeta_i)$ in the i-th call to $\mathscr{O}$ and the

set $Q_i = \operatorname{conv}\{u_1, u_2, \cdots, u_i\}$. Since we assume that the algorithm takes more than $T+1$ steps, we have $\min_{g \in Q_{T+1}} \|g\|_2 \geq \epsilon$. Using this and $u_i \in Q_{T+1}$, Lemma 5.3.5 lets us conclude that $v^*_{Q_{T+1}} \in \left\{w \in \mathbb{R}^d : \langle u_i, \zeta_i - w\rangle \leq 0\right\}$ for all $i \in [T+1]$. Since Ω_{T+1} is the intersection of the unit ball and these halfspaces, we have

$$v^*_{Q_{T+1}} \in \Omega_{T+1}.$$

Per (5.3.6), Ω_{T+1} does not contain a ball of radius $\frac{\epsilon}{4L}$, and therefore we may conclude that

$$\text{there exists a point } \widetilde{v} \in \mathbb{B}_{\frac{\epsilon}{2L}}(v^*_{Q_{T+1}}) \text{ such that } \widetilde{v} \notin \Omega_{T+1}.$$

Since $\widetilde{v} \in \mathbb{B}_2(0)$, the fact $\widetilde{v} \notin \Omega_{T+1}$ must be true due to one of the halfspaces generated in Algorithm 5.3.1. In other words, there must exist some $i \in [T+1]$ with

$$\langle u_i, \zeta_i - \widetilde{v}\rangle > 0.$$

By the guarantee of $\mathcal{O}$, we have $\langle u_i, \zeta_i\rangle \leq \frac{\epsilon}{2}$, and hence

$$\langle u_i, \widetilde{v}\rangle = \langle u_i, v_i\rangle - \langle u, v_i - \widetilde{v}\rangle < \frac{\epsilon}{2}. \tag{5.3.7}$$

By applying $\widetilde{v} \in \mathbb{B}_{\frac{\epsilon}{2L}}(v^*_{Q_{T+1}})$, $u_i \in \partial_\delta f(x)$, L-Lipschitzness of f, and Lemma 5.3.4, we have

$$\langle u_i, \widetilde{v}\rangle \geq \langle u_i, v^*_{Q_{T+1}}\rangle - \frac{\epsilon}{2L}\|u_i\|_2 \geq \langle u_i, v^*_{Q_{T+1}}\rangle - \frac{\epsilon}{2} \geq \|g^*_{Q_{T+1}}\|_2 - \frac{\epsilon}{2}. \tag{5.3.8}$$

Combining (5.3.7) and (5.3.8) yields that $\min_{g \in Q_{T+1}} \|g\|_2 = \|g^*_{Q_{T+1}}\|_2 < \epsilon$. This contradicts the assumption that the algorithm takes more than $T+1$ steps and concludes the proof. □

Now, we are ready to prove the main theorem.

Proof of Theorem 5.3.3. We note that the outer loop in Algorithm 5.2.1 runs at most $O(\frac{\Delta}{\delta\epsilon})$ times because we decrease the objective by $\Omega(\delta\epsilon)$ every step. Combining this with Theorem 5.3.8 and Lemma 5.3.2, we have that with probability $1-\gamma$, the oracle complexity for L-Lipschitz function is

$$\left\lceil \frac{4\Delta}{\delta\epsilon} \right\rceil \cdot \lceil 8d \log(8L/\epsilon)) \rceil \cdot \left\lceil \frac{36L}{\epsilon} \right\rceil \cdot \left\lceil 2\log\left(\frac{4\Delta}{\gamma\delta\epsilon}\right) \right\rceil = O\left(\frac{\Delta L d}{\delta\epsilon^2} \cdot \log(L/\epsilon) \cdot \log(1/\gamma)\right)$$

and for L-Lipschitz and ρ-weakly convex function is $O\left(\frac{\Delta d}{\delta\epsilon} \log(L/\epsilon) \cdot \log(\delta\rho/\epsilon)\right)$. □

Chapter 6

A Fast Scale-Invariant Algorithm for Non-negative Least Squares with Non-negative Data

Nonnegative (linear) least square problems are a fundamental class of problems that is well-studied in statistical learning and for which solvers have been implemented in many of the standard programming languages used within the machine learning community. The existing off-the-shelf solvers view the non-negativity constraint in these problems as an obstacle and, compared to unconstrained least squares, perform additional effort to address it. However, in many of the typical applications, the data itself is nonnegative as well, and we show that the nonnegativity in this case makes the problem easier. In particular, while the worst-case dimension-independent oracle complexity for unconstrained least squares problems necessarily scales with one of the data matrix constants (typically the spectral norm) and these problems are solved to additive error, we show that nonnegative least squares problems with nonnegative data are solvable to multiplicative error and with complexity independent of any matrix constants. The algorithm we introduce is accelerated and based on a primal-dual perspective. We further show how to provably obtain linear convergence using adaptive restart coupled with our method and demonstrate its effectiveness on large-scale data via numerical experiments.

6.1 Introduction

Nonnegative least squares (NNLS) problems, defined by $\min_{\mathbf{x}\geq\mathbf{0}} \frac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2$, where $\mathbf{A}\in\mathbb{R}^{m\times n}$ and $\mathbf{b}\in\mathbb{R}^m$, are fundamental problems and have been studied for decades in optimization and statistical learning [LH95, BJ97, KSD13], with various off-the-shelf solvers available in standard packages of Python (as `optimize.nnls` in the SciPy package), Julia (as `nnls.jl`), and MATLAB (as `lsqnonneg`). Within machine learning, NNLS problems arise whenever having negative labels is not meaningful, for example, when representing prices, age, pixel intensities, chemical concentrations, or frequency counts. NNLS is also widely used as a subroutine in nonnegative matrix factorization [CZPA09, Gil14, KSK13] to extract sparse features in applications like clustering, collaborative filtering, and community detection.

From a statistical perspective, NNLS problems can be shown to possess a regularization property that enforces sparsity similar to LASSO [Tib96], while being comparatively simpler, without the need to tune a regularization parameter or perform cross-validation [SH14, BEZ08, FK14, KJ18, WXT11, SJC19]. From an algorithmic standpoint, the nonnegativity constraint in NNLS problems is typically viewed as an obstacle: most NNLS algorithms perform additional work to handle it, and the problem is considered harder than unconstrained least squares.

However, in many important applications of NNLS, such as text mining [BBL+07], functional MRI [AR04, JHD18], EEG data analysis [MMSBVS08], pulse oximetry [JP87, WPTP88], observa-

tional astronomy [IFAB90], and those traditionally addressed using nonnegative matrix factorization [CZPA09], *the data is also nonnegative*. We argue in this chapter that when the data for NNLS is nonnegative, it is in fact possible to obtain *stronger* guarantees than for traditional least squares.

Our Contributions. We study NNLS problems with (element-wise) nonnegative data matrix $\mathbf{A}$, to which we refer as the NNLS+ problems, through the lens of the (equivalent) quadratic problems:

$$\min_{\mathbf{x}\geq\mathbf{0}} \left\{ \bar{f}(\mathbf{x}) \stackrel{\text{def}}{=} \frac{1}{2}\|\mathbf{A}\mathbf{x}\|_2^2 - \mathbf{c}^\top \mathbf{x} \right\}, \tag{P}$$

where $\mathbf{c} = \mathbf{A}^\top \mathbf{b}$ may be assumed element-wise positive. This assumption is without loss of generality since if $c_j < 0$ for some j, then $\nabla_j \bar{f}(\mathbf{x}) \geq 0$, implying that the j^{th} coordinate of the optimal solution is zero[1]. Hence, we could fix $x_j = 0$ and optimize over only the remaining coordinates.

We further assume that the matrix $\mathbf{A}$ is non-degenerate: none of its rows or columns has all of its elements equal to zero. This assumption is without loss of generality because (1) if such a row exists, we could remove it without affecting the objective, and (2) if the j^{th} column had all elements equal to zero, the optimal value of (P) would be $-\infty$, obtained for $\mathbf{x}$ with $x_j \to \infty$. Having established our assumptions and setup, we now proceed to state our contributions, which are three-fold.

(1) A scale-invariant, ϵ-multiplicative algorithm. We design an algorithm based on coordinate descent that, in total cost $O(\frac{\text{nnz}(\mathbf{A})}{\sqrt{\epsilon}})$, constructs an *$\epsilon$-multiplicative* approximate solution to (P). Our algorithm capitalizes on structural properties of (P) that arise as a result of the nonnegativity of $\mathbf{A}$.

Theorem 6.1.1 (Informal; see Theorem 6.3.5)**.** *Given a matrix $\mathbf{A} \in \mathbb{R}_+^{m\times n}$ and $\epsilon > 0$, define $f(\mathbf{x}) = \frac{1}{2}\|\mathbf{A}\mathbf{x}\|_2^2 - \mathbf{c}^\top\mathbf{x}$ and $\mathbf{x}^\star \in \operatorname{argmin}_{\mathbf{x}\geq\mathbf{0}} f(\mathbf{x})$. Then, there exists an algorithm that in $K = O(n\log n + \frac{n}{\sqrt{\epsilon}})$ iterations and $O\Big(nnz(\mathbf{A})\Big(\log n + \frac{1}{\sqrt{\epsilon}}\Big)\Big)$ arithmetic operations returns $\widetilde{\mathbf{x}}_K \in \mathbb{R}_+^n$ such that $\mathbb{E}\left[\langle \nabla f(\widetilde{\mathbf{x}}_K), \widetilde{\mathbf{x}}_K - \mathbf{x}^\star\rangle\right] \leq \epsilon |f(\mathbf{x}^\star)|$.*

The application of our structural observations on (P) to Theorem 4.6 of [DO19] enables the recovery of our guarantee on the optimality gap; however, we provide a guarantee on the primal-dual gap, and this is *stronger* than the one on the optimality gap stated in Theorem 6.1.1. What is significant about Theorem 6.1.1 is the *invariance* of the computational complexity to the scale of $\mathbf{A}$—it does not depend on any matrix constants. This cost stands in stark contrast to that of traditional least squares, where the dependence of (oracle) complexity on matrix constants (specifically, the spectral norm of $\mathbf{A}$ in the Euclidean case) is *unavoidable* [Nem92, NY83a], and multiplicative approximation is *not possible in general*.[2] In general, scale-invariance is a crucial feature in problems with data matrices, since a dependence on the width implies that the algorithm is technically not polynomial-time. This feature has, in fact, been an object of extensive study in the long line of works on packing and covering linear programs [Wan17, AZO19] and its variants such as a fair packing [DFO20a]. Conceptually, our algorithm is a new acceleration technique inspired by VRPDA^2 [SWD21].

(2) Linear convergence with restart. By incorporating adaptive restart in (P), we improve the guarantee of Theorem 6.1.1 to one with linear convergence (with $\log(1/\epsilon)$ complexity). Thus, we establish the first theoretical guarantee for NNLS+ that simultaneously satisfies the properties of being *scale-invariant, accelerated, and linearly-convergent*.

[1] To see this, note that by first-order optimality condition $\nabla \bar{f}(\mathbf{x}^\star)^\top(\mathbf{x} - \mathbf{x}^\star) \geq 0$ for all $\mathbf{x} \geq \mathbf{0}$. Choosing $\mathbf{x}$ with $\mathbf{x}_i = \mathbf{x}_i^\star$ for all $i \neq j$ and $\mathbf{x}_j = 0$ in the first-order optimality condition gives $\mathbf{x}_j^\star = 0$.

[2] To see why, consider a case in which the optimal objective value equals zero. Then any problem with a multiplicative guarantee of the form in Theorem 6.1.1 would necessarily return an optimal solution.

Theorem 6.1.2 (Informal; see Theorem 6.4.1)**.** *Consider the setup of Theorem 6.1.1. Then, there is an algorithm that in expected* $O(\mathrm{nnz}(\mathbf{A})(\log n + \frac{\sqrt{n}}{\mu})\log(\frac{1}{\varepsilon}))$ *arithmetic operations returns* $\widetilde{\mathbf{x}}_K \in \mathbb{R}^n_+$ *with* $f(\widetilde{\mathbf{x}}_K) - f(\mathbf{x}^\star) \leq \epsilon |f(\mathbf{x}^\star)|$*, where* μ *is the constant in a local error bound for* (P).

Proving this bound requires bounding the expected number of iterations between restarts in conjunction with careful technical work in identifying an appropriate local error bound for NNLS+.

(3) Numerical experiments. We consolidate our theoretical contributions with a demonstration of the empirical advantage of our restarted method over state-of-the-art solvers via numerical experiments on datasets from LibSVM with sizes up to 19996×1355191. Figure 6.1 shows that, when combined with the restart strategy, our algorithm significantly outperforms the compared algorithms.

Related work. NNLS has seen a large body of work on the empirical front. The first method that was widely adopted in practice (including in the `lsqnonneg` implementation of MATLAB) is due to the seminal work of [LH95] (originally published in 1974). This method, based on active sets, solves NNLS via a sequence of (unconstrained) least squares problems and has been followed up by [BJ97, VBK04, MFLS17, DDM21] with improved empirical performance. While these variants are generally effective on small to mid-scale problem instances, they are not suitable for extreme-scale problems ubiquitous in machine learning. For example, in the experiments reported in [MFLS17], Fast NNLS [BJ97] took 6.63 days to solve a problem of size 45000×45000, while the TNT-NN algorithm [MFLS17] took 2.45 hours. However, the latter requires computing the Cholesky decomposition of $\mathbf{A}^\top\mathbf{A}$ at initialization, which can be prohibitively expensive both in computation and in memory. Another prominent work on the empirical front is that of [KSD13], which performs projected gradient descent with modified Barzilai-Borwein steps [BB88] and step sizes a carefully designed sequence of diminishing scalars.

Another, separate, line of work concerns optimization algorithms with multiplicative error guarantees. Of interest to us are standard first-order algorithms that run in time that is near-linear in the input size and are thus applicable to large-scale setting (for multiplicative-error algorithms applicable to broad classes of problems but that run in time that is superlinear in the input size, see [N+18, Chapter 7]). Most of the existing literature in this domain concerns positive (packing and covering) linear programs (e.g., [LN93, You01, AZO19, MRWZ16]). Results also exist for positive semidefinite programs [AZQRY16], (nonlinear) fair packing and covering problems [MSZ16, DFO20a], and fair packing problems under Schatten norms for matrices [JLT20]. With the exception of [DFO20a] (discussed below), none of these results are directly applicable to (P).

To the best of our knowledge, theoretical guarantees explicitly for (P) have been scarce. For instance, [KSD13] mentioned in the preceding paragraph provides only asymptotic guarantees. Orthogonally, the result on 1-fair covering by [DFO20a] solves the dual of NNLS+, which also gives a multiplicative guarantee for NNLS+, but with overall complexity $\tilde{O}(\frac{\mathrm{nnz}(\mathbf{A})}{\varepsilon})$.

Since our algorithm is based on the coordinate descent algorithm, we highlight some results of other coordinate descent algorithms when specialized to the closely related problem of *unconstrained linear regression*. The pioneering work of [Nes12] proposed a coordinate descent method called `RCDM`, which in our setting has an iteration cost $O\big(\frac{\sum_{j=1}^n \|\mathbf{A}_{:j}\|_2^2 \|\mathbf{x}_0 - \mathbf{x}^\star\|^2}{\epsilon}\big)$, where $\|\mathbf{A}_{:j}\|_2$ is the Euclidean norm of the j^{th} column of $\mathbf{A}$. This was improved by [LS13], in an algorithm termed `ACDM`, by combining Nesterov's estimation technique [Nes83a] and coordinate sampling, giving an iteration complexity of $O\big(\frac{\sqrt{n\sum_{j=1}^n \|\mathbf{A}_{:j}\|_2^2}\|\mathbf{x}_0 - \mathbf{x}^\star\|}{\sqrt{\epsilon}}\big)$ for solving (P). The latest results in this line of work by [AZQRY16, QR16, NS17] perform non-uniform sampling atop a framework of [Nes12] and

achieve iteration complexity of $O\Big(\frac{\sqrt{\sum_{j=1}^n \|\mathbf{A}_{:j}\|_2^2}\|\mathbf{x}_0-\mathbf{x}^\star\|}{\sqrt{\epsilon}}\Big)$, with [DO18] dropping the dependence on $\max_{1\le j\le n}\|\mathbf{A}_{:j}\|_2$. Additionally, the work of [LLX14] develops an accelerated randomized proximal coordinate gradient (APCG) method to minimize composite convex functions.

As remarked earlier, [DO18], coupled with insights on NNLS+ problems provided in this work, can recover our guarantee for the optimality gap from Theorem 6.3.5. However, our work is the first to bring to the fore the properties of NNLS+ *required* to get such a guarantee, and our choice of primal-dual perspective allows for a stronger guarantee in terms of an upper bound on the primal-dual gap. Further, our algorithm is a novel type of acceleration, with our primal-dual perspective transparently illustrating our use of the aforementioned properties. We believe that these technical contributions, along with our techniques to obtain vastly improved theoretical guarantees with the restart strategy applied to this problem, are valuable to the broader optimization and machine learning communities.

6.2 Notation and Preliminaries

Throughout this chapter, we use bold lowercase letters to denote vectors and bold uppercase letters for matrices. For vectors and matrices, the operator $'\ge'$ is applied element-wise, and $\mathbb{R}_+$ is the non-negative part of the real line. We use $\langle \mathbf{a}, \mathbf{b}\rangle$ to denote the inner product of vectors $\mathbf{a}$ and $\mathbf{b}$ and ∇ for gradient. Given a matrix $\mathbf{A}$, we use $\mathbf{A}_{:j}$ for its j^{th} column vector, and for a vector $\mathbf{x}$, x_j denotes its j^{th} coordinate. We use $\text{nnz}(\mathbf{A})$ for the number of non-zero entries of $\mathbf{A}$. We use $\mathbf{x}_k$ for the vector in the k^{th} iteration and, to disambiguate indexing, use $[\mathbf{x}_k]_j$ to mean the j^{th} coordinate of $\mathbf{x}_k$. The i^{th} standard basis vector is denoted by $\mathbf{e}_i$. For an n-dimensional vector $\mathbf{x}$ and $\mathbf{A}\in\mathbb{R}^{m\times n}$, we define $\mathbf{\Lambda} = \text{diag}([\|\mathbf{A}_{:1}\|_2^2,\dots,\|\mathbf{A}_{:n}\|_2^2])$ and $\|\mathbf{x}\|_{\mathbf{\Lambda}}^2 = \sum_{i=1}^n x_i^2\|\mathbf{A}_{:i}\|_2^2$. Finally, $[n] \stackrel{\text{def}}{=} \{1,2,\dots,n\}$.

A differentiable function $f:\mathbb{R}^n\to\mathbb{R}$ is convex if for any $\mathbf{x},\widehat{\mathbf{x}}\in\mathbb{R}^n$, we have $f(\widehat{\mathbf{x}}) \ge f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \widehat{\mathbf{x}}-\mathbf{x}\rangle$. A differentiable function $f:\mathbb{R}^n\to\mathbb{R}$ is said to be μ-strongly convex w.r.t. the ℓ_2-norm if for any $\mathbf{x},\widehat{\mathbf{x}}\in\mathbb{R}^n$, we have that $f(\widehat{\mathbf{x}}) \ge f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \widehat{\mathbf{x}}-\mathbf{x}\rangle + \frac{\mu}{2}\|\mathbf{x}-\widehat{\mathbf{x}}\|_2^2$. We have analogous definitions for concave and strongly concave functions, which flip the inequalities noted.

Given a convex program $\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$, where $f:\mathbb{R}^n\to\mathbb{R}$ is differentiable and convex and $\mathcal{X}\subseteq\mathbb{R}^n$ closed and convex, the first-order optimality condition of a solution $\mathbf{x}^\star\in\text{argmin}_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$ is

$$(\forall \mathbf{x}\in\mathcal{X}): \quad \langle \nabla f(\mathbf{x}^\star), \mathbf{x}-\mathbf{x}^\star\rangle \ge 0. \tag{6.2.1}$$

Problem setup. As discussed in the introduction, our goal is to solve (P), with $\mathbf{A}\in\mathbb{R}_+^{m\times n}$. For notational convenience, we work with the problem in the following scaled form:

$$\min_{\mathbf{x}\in\mathbb{R}_+^n}\Big\{f(\mathbf{x}) \stackrel{\text{def}}{=} \frac{1}{2}\|\mathbf{A}\mathbf{x}\|_2^2 - \mathbf{1}^\top\mathbf{x}\Big\}, \tag{6.2.2}$$

This assumption is w.l.o.g. since any (P) can be brought to this form by a simple change of variable $\hat{x}_j = c_j x_j$ (see also e.g., [AZO19, DFO20a] for similar scaling ideas). The scaling need not be explicit in the algorithm since the change of variable $\hat{x}_j = c_j x_j$ is easily reversible.

Properties of the objective. To kick off our analysis, we highlight some properties inherent to the objective defined in (6.2.2). These properies, which strongly need the non-negativity of $\mathbf{A}$ and $\mathbf{x}$, are central to obtain a scale-invariant algorithm for (P).

Proposition 6.2.1. *Given $f:\mathbb{R}_+^n\to\mathbb{R}$ as defined in (6.2.2) and $\mathbf{x}^\star\in\text{argmin}_{\mathbf{x}\in\mathbb{R}_+^n} f(\mathbf{x})$, the following statements all hold.*

a) $\nabla f(\mathbf{x}^\star) \geq \mathbf{0}$.
b) $f(\mathbf{x}^\star) = -\frac{1}{2}\|\mathbf{A}\mathbf{x}^\star\|_2^2 = -\frac{1}{2}\mathbf{1}^\top \mathbf{x}^\star$.
c) for all $j \in [n]$, *we have* $x_j^\star \in \left[0, \frac{1}{\|\mathbf{A}_{:j}\|_2^2}\right]$.
d) $-\frac{1}{2}\sum_{j\in[n]} \frac{1}{\|\mathbf{A}_{:j}\|_2^2} \leq f(\mathbf{x}^\star) \leq -\frac{1}{2\min_{j\in[n]} \|\mathbf{A}_{:j}\|_2^2}$.

The validity of division by $\|\mathbf{A}_{:j}\|_2$ in the preceding proposition is by the non-degeneracy of $\mathbf{A}$ discussed in the introduction. We prove this proposition in Section E.1.1.

An important consequence of Proposition 6.2.1 (c) is that (P) can be restricted to the hyperrectangle $\mathcal{X} = \{\mathbf{x} \in \mathbb{R}^n : 0 \leq x_j \leq \frac{1}{\|\mathbf{A}_{:j}\|_2^2}\}$ without affecting its optimal solution, but effectively reducing the search space. Thus, going forward, we replace the constraint $\mathbf{x} \geq \mathbf{0}$ in (P) by $\mathbf{x} \in \mathcal{X}$.

Primal-dual gap perspective. As alluded earlier, our algorithm is analyzed through a primal-dual perspective. For this reason, it is useful to consider the Lagrangian

$$\mathcal{L}(\mathbf{x}, \mathbf{y}) = \langle \mathbf{A}\mathbf{x}, \mathbf{y}\rangle - \frac{1}{2}\|\mathbf{y}\|_2^2 - \mathbf{1}^\top \mathbf{x} \tag{6.2.3}$$

from which we can derive our rescaled problem (6.2.2) as the primal problem $\min_{\mathbf{x}\in\mathcal{X}} \mathcal{P}(\mathbf{x})$, where

$$\mathcal{P}(\mathbf{x}) = \max_{\mathbf{y}\in\mathbb{R}^m} \mathcal{L}(\mathbf{x}, \mathbf{y}) = -\mathbf{1}^\top \mathbf{x} + \max_{\mathbf{y}\geq\mathbf{0}} \Big[-\frac{1}{2}\|\mathbf{y}\|_2^2 + \langle \mathbf{A}\mathbf{x}, \mathbf{y}\rangle \Big] = -\mathbf{1}^\top \mathbf{x} + \frac{1}{2}\|\mathbf{A}\mathbf{x}\|^2.$$

Thus, the Lagrangian is constructed in a way that one can derive the primal problem from it while also localizing the matrix in the bilinear term and ensuring coordinate-wise separability of the terms involving either only the dual or only the primal variable since this greatly simplifies our analysis. Similar to [SWD21], we use Equation (6.2.3) to define the following relaxation of the primal-dual gap, for arbitrary but fixed $\mathbf{u} \in \mathcal{X}$, and $\mathbf{v} \in \mathbb{R}^m$:

$$\mathrm{Gap}_{\mathcal{L}}^{(\mathbf{u},\mathbf{v})}(\mathbf{x}, \mathbf{y}) \stackrel{\text{def}}{=} \mathcal{L}(\mathbf{x}, \mathbf{v}) - \mathcal{L}(\mathbf{u}, \mathbf{y}). \tag{6.2.4}$$

The significance of this relaxed gap function is that for a candidate solution $\widetilde{\mathbf{x}}$ and an arbitrary $\widetilde{\mathbf{y}} \in \mathbb{R}^m$, a bound on $\mathrm{Gap}_{\mathcal{L}}^{(\mathbf{u},\mathbf{v})}(\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}})$ translates to one on the primal error, as follows. First select $\mathbf{u} = \mathbf{x}^\star$, $\mathbf{v} = \mathbf{A}\widetilde{\mathbf{x}}$. Then, by observing that $\mathcal{L}(\widetilde{\mathbf{x}}, \mathbf{A}\widetilde{\mathbf{x}}) = f(\widetilde{\mathbf{x}})$ and $\mathcal{L}(\mathbf{x}^\star, \mathbf{A}\mathbf{x}^\star) = f(\mathbf{x}^\star)$, we have

$$f(\widetilde{\mathbf{x}}) - f(\mathbf{x}^\star) = \mathcal{L}(\widetilde{\mathbf{x}}, \mathbf{A}\widetilde{\mathbf{x}}) - \mathcal{L}(\mathbf{x}^\star, \mathbf{A}\mathbf{x}^\star).$$

For a fixed $\mathbf{x}$, $\mathcal{L}(\mathbf{x}, \cdot)$ is 1-strongly concave and minimized at $\mathbf{A}\mathbf{x}$. Thus, $\mathcal{L}(\mathbf{x}^\star, \mathbf{A}\widetilde{\mathbf{x}}) \leq \mathcal{L}(\mathbf{x}^\star, \mathbf{A}\mathbf{x}^\star) - \frac{1}{2}\|\mathbf{A}(\widetilde{\mathbf{x}} - \mathbf{x}^\star)\|^2$. Hence, we have the following primal bound:

$$f(\widetilde{\mathbf{x}}) - f(\mathbf{x}^\star) + \frac{1}{2}\|\mathbf{A}(\widetilde{\mathbf{x}} - \mathbf{x}^\star)\|^2 \leq \mathcal{L}(\widetilde{\mathbf{x}}, \mathbf{A}\widetilde{\mathbf{x}}) - \mathcal{L}(\mathbf{x}^\star, \widetilde{\mathbf{y}}) = \mathrm{Gap}_{\mathcal{L}}^{(\mathbf{x}^\star, \mathbf{A}\widetilde{\mathbf{x}})}(\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}}). \tag{6.2.5}$$

In light of this connection, our algorithm for bounding the primal error is one that generates iterates that can be used to construct bounds on $\mathrm{Gap}_{\mathcal{L}}^{(\mathbf{u},\mathbf{v})}(\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}})$, as we detail next.

6.3 Our Algorithm and Convergence Analysis

Our algorithm, Scale Invariant NNLS+ (SI-NNLS+), is an iterative algorithm using the estimate sequences $\phi_k(\mathbf{x})$ and $\psi_k(\mathbf{y})$ for $k \geq 1$ (see Section 6.3.1) giving the primal and dual updates

$$\mathbf{x}_k = \operatorname{argmin}_{\mathbf{x}\in\mathcal{X}} \phi_k(\mathbf{x}) \quad \text{and} \quad \mathbf{y}_k = \operatorname*{argmax}_{\mathbf{y}\in\mathbb{R}^m} \psi_k(\mathbf{y}). \tag{6.3.1}$$

We use our algorithm's iterates from Equation (6.3.1) to construct G_k, an upper estimate of $\mathrm{Gap}_{\mathcal{L}}^{(\mathbf{u},\mathbf{v})}(\widetilde{\mathbf{x}}_k, \widetilde{\mathbf{y}}_k)$, where $\widetilde{\mathbf{x}}_k, \widetilde{\mathbf{y}}_k$ are convex combinations of the iterates and $\mathbf{u} \in \mathcal{X}, \mathbf{v} \in \mathbb{R}^m_+$, where $\mathbf{u}$ is fixed and $\mathbf{v}$ is arbitrary. Our motivation for constructing $G_k(\mathbf{u}, \mathbf{v}) \geq \mathrm{Gap}_{\mathcal{L}}^{(\mathbf{u},\mathbf{v})}(\widetilde{\mathbf{x}}_k, \widetilde{\mathbf{y}}_k)$ is to obtain a bound on the primal error, using Inequality 6.2.5. The main goal in the analysis is to show $G_k \leq \frac{Q}{A_k}$, where A_k is a sequence of positive numbers and Q is bounded. To obtain the claimed multiplicative approximation, we use Inequality 6.2.5 and argue that for $\mathbf{u} = \mathbf{x}^\star, \mathbf{v} = \mathbf{A}\widetilde{\mathbf{x}}_k$, we have $Q \leq O\Big(|f(\mathbf{x}^*)|\Big)$.

Our algorithm may be interpreted as a variant of the VRPDA^2 algorithm [SWD21], with our analysis inspired by the approximate duality gap technique [DO19]; however, unlike VRPDA^2, which uses estimate sequences, our analysis directly bounds the primal-dual gap. Another difference from VRPDA^2 is in our choice of primal regularizer (described shortly) and our lack of a dual regularizer. A variant of SI-NNLS+ suitable for analysis is shown in Algorithm 6.3.1, with proofs in Section E.1.2 and Section E.1.3. We give an equivalent implementation version ("*Lazy* SI-NNLS+", which updates as few coordinates as is possible to improve cost per iteration) in Algorithm E.2.1 and its analysis in Section E.2.

Algorithm 6.3.1 Scale Invariant Non-negative Least Squares with Non-negative Data (SI-NNLS+)

Input: Matrix $\mathbf{A} \in \mathbb{R}^{m\times n}_+$ with $n \geq 4$, accuracy ε, initial point $\mathbf{x}_0$
Initialize: $\widetilde{\mathbf{x}}_0 = \mathbf{x}_0$, $\overline{\mathbf{y}}_0 = \mathbf{y}_0 = \mathbf{A}\mathbf{x}_0$, $K = \frac{5}{2}n\log n + \frac{6n}{\sqrt{\epsilon}}$, $a_1 = \frac{1}{\sqrt{2}n^{1.5}}$, $a_2 = \frac{a_1}{n-1}$, $A_0 = 0$, $A_1 = a_1$, $\phi_0(\mathbf{x}) = \frac{1}{2}\|\mathbf{x} - \mathbf{x}_0\|^2_{\mathbf{\Lambda}}$.
for $k = 1, 2, \ldots, K$ **do**
  Sample j_k uniformly at random from $\{1, 2, \ldots, n\}$
  $\mathbf{x}_k \leftarrow \arg\min_{\mathbf{x}\in\mathcal{X}} \phi_k(\mathbf{x})$, for $\phi_k(\mathbf{x})$ defined by Equation (6.3.7) and Equation (6.3.10)
  $\mathbf{y}_k \leftarrow \arg\max_{\mathbf{y}\in\mathbb{R}^m} \psi_k(\mathbf{y})$, for $\psi_k(\mathbf{y})$ defined by Equation (6.3.4)
  $\widetilde{\mathbf{x}}_k = \frac{1}{A_k}\Big[A_{k-1}\widetilde{\mathbf{x}}_{k-1} + a_k\Big(n\mathbf{x}_k - (n-1)\mathbf{x}_{k-1}\Big)\Big]$.
  $\overline{\mathbf{y}}_k \leftarrow \mathbf{y}_k + \frac{a_k}{a_{k+1}}(\mathbf{y}_k - \mathbf{y}_{k-1})$
  $A_{k+1} \leftarrow A_k + a_{k+1}$, $a_{k+2} = \min\{\frac{na_{k+1}}{n-1}, \frac{\sqrt{A_{k+1}}}{2n}\}$
end for
Return $\widetilde{\mathbf{x}}_K$

6.3.1 Gap Estimate Construction

The gap estimate G_k is constructed as the difference $G_k(\mathbf{u}, \mathbf{v}) = U_k(\mathbf{v}) - L_k(\mathbf{u})$,where $U_k(\mathbf{v}) \geq \mathcal{L}(\widetilde{\mathbf{x}}_k, \mathbf{v})$ and $L_k(\mathbf{u}) \leq \mathcal{L}(\mathbf{u}, \widetilde{\mathbf{y}}_k)$ are, respectively, upper and lower bounds we construct on the Lagrangian. It then follows by Equation (6.2.4) that $G_k(\mathbf{u}, \mathbf{v})$ is a valid upper estimate of $\mathrm{Gap}_{\mathcal{L}}^{(\mathbf{u},\mathbf{v})}(\widetilde{\mathbf{x}}_k, \widetilde{\mathbf{y}}_k)$.

We first introduce a technical component our constructions L_k and U_k crucially hinge on: we define two positive sequences of numbers $\{a_i\}_{i\geq 1}$ and $\{a^k_i\}_{1\leq i\leq k}$, with one of their properties being that both sum up to $A_k > 0$ for $k \geq 1$. Specifically, we define $A_0 = 0$ and $\{a_i\}_{i\geq 1}$ as $a_i = A_i - A_{i-1}$. The sequence

$\{a_i^k\}$ changes with k and for $k = 1$ is defined by $a_1^1 = a_1$, while for $k \geq 2$:

$$a_i^k = \begin{cases} a_1 - (n-1)a_2, & \text{if } i = 1, \\ na_i - (n-1)a_{i+1}, & \text{if } 2 \leq i \leq k-1, \\ na_k, & \text{if } i = k. \end{cases} \tag{6.3.2}$$

Summing over $i \in [k]$ verifies that $A_k = \sum_{i=1}^k a_i^k$. For the sequence $\{a_i^k\}_{1 \leq i \leq k}$ to be non-negative, we further require that $a_1 - (n-1)a_2 \geq 0$ and $\forall i \geq 2$, $na_i - (n-1)a_{i+1} \geq 0$.

The significance of these two sequences lies in defining the algorithm's primal-dual output pair by

$$\widetilde{\mathbf{x}}_k = \frac{1}{A_k} \sum_{i \in [k]} a_i^k \mathbf{x}_i \quad \text{and} \quad \widetilde{\mathbf{y}}_k = \frac{1}{A_k} \sum_{i \in [k]} a_i \mathbf{y}_i. \tag{6.3.3}$$

The intricate interdependence of $\{a_i\}$ and $\{a_i^k\}$ enables expressing $\widetilde{\mathbf{x}}_k$ in terms of only $\{a_i\}$. This expression further simplifies to a cheaper recursive one, which is used in Algorithm 6.3.1.

With the sequences $\{a_i\}_{i \geq 1}$ and $\{a_i^k\}_{1 \leq i \leq k}$ in tow, we are now ready to show the construction of an upper bound $\mathrm{U}_k(\mathbf{v})$ on $\mathcal{L}(\widetilde{\mathbf{x}}_k, \mathbf{v})$ and a lower bound $\mathrm{L}_k(\mathbf{u})$ on $\mathcal{L}(\mathbf{u}, \widetilde{\mathbf{y}}_k)$.

Upper bound. To construct an upper bound, first observe that by Equation (6.2.3) and Equation (6.3.3),

$$\mathcal{L}(\widetilde{\mathbf{x}}_k, \mathbf{v}) = \langle \mathbf{A}\widetilde{\mathbf{x}}_k, \mathbf{v} \rangle - \frac{1}{2}\|\mathbf{v}\|_2^2 - \mathbf{1}^\top \widetilde{\mathbf{x}}_k = \frac{1}{A_k} \sum_{i \in [k]} a_i^k \Big[\langle \mathbf{A}\mathbf{x}_i, \mathbf{v} \rangle - \frac{1}{2}\|\mathbf{v}\|_2^2 - \mathbf{1}^\top \mathbf{x}_i \Big].$$

Consider the primal estimate sequence defined for $k = 0$ as $\psi_0 = 0$ and for $k \geq 1$ by

$$\psi_k(\mathbf{v}) \stackrel{\text{def}}{=} \sum_{i \in [k]} a_i^k \Big[\langle \mathbf{A}\mathbf{x}_i, \mathbf{v} \rangle - \frac{1}{2}\|\mathbf{v}\|_2^2 - \mathbf{1}^\top \mathbf{x}_i \Big], \tag{6.3.4}$$

which ensures that $\mathcal{L}(\widetilde{\mathbf{x}}_k, \mathbf{v}) = \frac{1}{A_k}\psi_k(\mathbf{v})$. A key upshot of constructing $\psi_k(\mathbf{v})$ as in Equation (6.3.4) is that the quadratic term implies A_k-strong concavity of ψ_k for $k \geq 1$, which in turn ensures that the vector $\mathbf{y}_k = \arg\max_{\mathbf{y} \in \mathbb{R}^m} \psi_k(\mathbf{y})$ from Equation (6.3.1) is unique. This property, coupled with the first-order optimality condition in Inequality 6.2.1, gives that for any $\mathbf{y} \in \mathbb{R}^m$, $\psi_k(\mathbf{y}) \leq \psi_k(\mathbf{y}_k) - \frac{A_k}{2}\|\mathbf{y} - \mathbf{y}_k\|_2^2$. We are now ready to define the following upper bound by:

$$\mathrm{U}_k(\mathbf{v}) \stackrel{\text{def}}{=} \frac{1}{A_k}\psi_k(\mathbf{y}_k) - \frac{1}{2}\|\mathbf{v} - \mathbf{y}_k\|_2^2. \tag{6.3.5}$$

The preceding discussion immediately implies that U_k is a valid upper bound for the Lagrangian.

Lemma 6.3.1. *For U_k as defined in Equation (6.3.5), Lagrangian defined in Equation (6.2.3) and $\widetilde{\mathbf{x}}_k \in \mathbb{R}_+^n$ in Equation (6.3.3), we have, for all $\mathbf{y} \in \mathbb{R}^m$, the upper bound $\mathrm{U}_k(\mathbf{y}) \geq \mathcal{L}(\widetilde{\mathbf{x}}_k, \mathbf{y})$.*

Lower bound. Analogous to the preceding section, we now obtain a *lower bound* on the Lagrangian, completing the bound on the gap estimate. However, the construction becomes more technical. We start with the same approach as for the upper bound. Since $\mathcal{L}(\mathbf{u}, \widetilde{\mathbf{y}}_k)$ is concave in $\widetilde{\mathbf{y}}_k$, by Jensen's inequality: $\mathcal{L}(\mathbf{u}, \widetilde{\mathbf{y}}_k) \geq \frac{1}{A_k} \sum_{i \in [k]} a_i \Big(\langle \mathbf{A}\mathbf{u}, \mathbf{y}_i \rangle - \mathbf{1}^\top \mathbf{u} - \frac{1}{2}\|\mathbf{y}_i\|_2^2 \Big)$.Were we to define the dual estimate sequence ϕ_k in the same way as we did for the primal estimate sequence ψ_k, we would now simply

define it as A_k times the right-hand side in the last inequality. However, doing so would make ϕ_k depend on $\mathbf{y}_k$, which is updated *after* $\mathbf{x}_k$, which in Equation (6.3.1) is defined as the minimizer of the ϕ_k.

To avoid such a circular dependency, we add and subtract a linear term $\sum_{i\in[k]}\langle \mathbf{A}^\top \overline{\mathbf{y}}_{i-1}, \mathbf{u}\rangle$, where $\overline{\mathbf{y}}_{i-1}$, defined later, are extrapolation points that depend only on $\mathbf{y}_1, \dots \mathbf{y}_{i-1}$. We thus have

$$\mathcal{L}(\mathbf{u}, \widetilde{\mathbf{y}}_k) \geq \frac{1}{A_k}\sum_{i\in[k]} a_i\Big[\langle \mathbf{A}\mathbf{u}, \overline{\mathbf{y}}_{i-1}\rangle - \mathbf{1}^\top\mathbf{u} - \frac{1}{2}\|\mathbf{y}_i\|_2^2\Big] + \frac{1}{A_k}\sum_{i\in[k]} a_i\langle \mathbf{A}\mathbf{u}, \mathbf{y}_i - \overline{\mathbf{y}}_{i-1}\rangle.$$

If we now defined ϕ_k based on the first term in the above inequality, we run into another obstacle: the linearity of the resulting estimate sequence is insufficient for cancelling all the error terms in the analysis. Hence, as is common, we introduce strong convexity by adding and subtracting an appropriate strongly convex function. Our chosen strongly convex function is motivated by the box-constrained property of the optimum from Proposition 6.2.1 (c) and crucial in bounding the initial gap estimate. It coincides with ϕ_0: for any $\mathbf{x} \in \mathbb{R}^n_+$, define the function

$$\phi_0(\mathbf{x}) = \frac{1}{2}\|\mathbf{x} - \mathbf{x}_0\|_{\mathbf{\Lambda}}^2. \tag{6.3.6}$$

This function is 1-strongly convex with respect to $\|\cdot\|_{\mathbf{\Lambda}}$ and used in defining ϕ_1 as:

$$\phi_1(\mathbf{u}) = a_1\langle \mathbf{A}^\top\bar{\mathbf{y}}_0 - \mathbf{1}, \mathbf{u}\rangle + \phi_0(\mathbf{u}). \tag{6.3.7}$$

The definition of ϕ_1 is driven by the purpose of cancelling initial error terms. Next, we choose ϕ_k so that for any fixed $\mathbf{u} \in \mathcal{X}$, we have

$$\mathbb{E}[\phi_k(\mathbf{u})] = \mathbb{E}\Big[\sum_{i\in[k]} a_i\langle \mathbf{A}^\top\overline{\mathbf{y}}_{i-1} - \mathbf{1}, \mathbf{u}\rangle + \phi_0(\mathbf{u})\Big], \tag{6.3.8}$$

where the expectation is with respect to all the randomness in the algorithm. This construction is used to reduce the per-iteration complexity, for which we employ a *randomized* coordinate update on $\mathbf{x}_k$ for $k \geq 2$. To support such updates, we relax the lower bound to hold only *in expectation*.

Concretely, let j_i be the coordinate sampled uniformly at random from $[n]$ in the i^{th} iteration of SI-NNLS+, independent of history. Fix $\mathbf{y}_i$ for $i = 1, \dots, k-1$ and for $k \geq 2$ and $\mathbf{x} \in \mathcal{X}$, define

$$\phi_k(\mathbf{x}) = \phi_1(\mathbf{x}) + \sum_{i=2}^{k} n a_i\langle \mathbf{A}^\top\overline{\mathbf{y}}_{i-1} - \mathbf{1}, x_{j_i}\mathbf{e}_{j_i}\rangle. \tag{6.3.9}$$

For $k \geq 2$, $\phi_k(\mathbf{u})$ can also be defined recursively via

$$\phi_k(\mathbf{x}) = \phi_{k-1}(\mathbf{x}) + n a_k\langle \mathbf{A}^\top\overline{\mathbf{y}}_{k-1} - \mathbf{1}, x_{j_k}\mathbf{e}_{j_k}\rangle. \tag{6.3.10}$$

The function ϕ_k inherits the strong convexity of ϕ_0. This property, together with Equation (6.3.1) and first-order optimality from Inequality 6.2.1, give

$$\phi_k(\mathbf{x}) \geq \phi_k(\mathbf{x}_k) + \frac{1}{2}\|\mathbf{x} - \mathbf{x}_k\|_{\mathbf{\Lambda}}^2. \tag{6.3.11}$$

Along with strong convexity, our choice of ϕ_k in Equation (6.3.10) leads to the following properties essential to our analysis: (1) ϕ_k is separable in its coordinates; (2) the primal variable $\mathbf{x}_k$ is updated

only at its j_k^{th} coordinate; (3) Equation (6.3.8) is true. These are formally stated in Proposition E.1.2. With the dual estimate sequence ϕ_k defined in Equation (6.3.10), we now define the sequence L_k by

$$\mathrm{L}_k(\mathbf{x}) \stackrel{\text{def}}{=} \frac{\phi_k(\mathbf{x}_k) + \frac{1}{2}\|\mathbf{x}-\mathbf{x}_k\|_{\mathbf{\Lambda}}^2 - \phi_0(\mathbf{x}) + \sum_{i\in[k]} a_i(\langle \mathbf{A}\mathbf{x}, \mathbf{y}_i - \overline{\mathbf{y}}_{i-1}\rangle - \frac{1}{2}\|\mathbf{y}_i\|_2^2)}{A_k}. \tag{6.3.12}$$

We conclude this section by justifying our choice of L_k as a valid expected lower bound on $\mathbb{E}\mathcal{L}(\mathbf{x}, \widetilde{\mathbf{y}}_k)$.

Lemma 6.3.2. *For L_k defined in Equation (6.3.12), for the Lagrangian in Equation (6.2.3) and $\widetilde{\mathbf{y}}_k$ in Equation (6.3.3), we have, for a fixed $\mathbf{u} \in \mathcal{X}$, the lower bound $\mathbb{E}\mathcal{L}(\mathbf{u}, \widetilde{\mathbf{y}}_k) \geq \mathbb{E}\mathrm{L}_k(\mathbf{u})$, where the expectation is with respect to all the random choices of coordinates in Algorithm 6.3.1.*

6.3.2 Bounding the Gap Estimate

With the gap estimate G_k constructed as in the preceding section and combining Equation (6.3.5) and Equation (6.3.12), we now achieve our goal of bounding $A_k\mathrm{G}_k$ (to obtain a convergence rate of the order $1/A_k$) by bounding the change in $A_k\mathrm{G}_k$ and the initial scaled gap $A_1\mathrm{G}_1$.

Lemma 6.3.3. *Consider the iterates $\{\mathbf{x}_k\}$ and $\{\mathbf{y}_k\}$ evolving according to Algorithm 6.3.1. Let $n \geq 2$ and assume that $a_1 = \frac{1}{\sqrt{2}n^{1.5}}$ and $a_1 \geq (n-1)a_2$, while for $k \geq 3$,*

$$a_k \leq \min\Big(\frac{na_{k-1}}{n-1}, \frac{\sqrt{A_{k-1}}}{2n}\Big). \tag{6.3.13}$$

Then, for fixed $\mathbf{u} \in \mathcal{X}$, any $\mathbf{v} \in \mathbb{R}^m$, and all $k \geq 2$, the gap estimate $\mathrm{G}_k = \mathrm{U}_k - \mathrm{L}_k$ satisfies

$$\begin{aligned}
&\mathbb{E}(A_kG_k(\mathbf{x},\mathbf{y}) - A_{k-1}G_{k-1}(\mathbf{x},\mathbf{y})) \\
&\leq -\mathbb{E}\Big(\frac{A_k}{2}\|\mathbf{y}-\mathbf{y}_k\|_2^2 - \frac{A_{k-1}}{2}\|\mathbf{y}-\mathbf{y}_{k-1}\|_2^2\Big) - \frac{1}{2}\mathbb{E}\|\mathbf{x}-\mathbf{x}_k\|_{\mathbf{\Lambda}}^2 + \frac{1}{2}\mathbb{E}\|\mathbf{x}-\mathbf{x}_{k-1}\|_{\mathbf{\Lambda}}^2 \\
&\quad - a_k\mathbb{E}\langle \mathbf{A}(\mathbf{x}-\mathbf{x}_k), \mathbf{y}_k - \mathbf{y}_{k-1}\rangle + a_{k-1}\mathbb{E}\langle \mathbf{A}(\mathbf{x}-\mathbf{x}_{k-1}), \mathbf{y}_{k-1} - \mathbf{y}_{k-2}\rangle \\
&\quad - \frac{1}{4}A_{k-1}\mathbb{E}\|\mathbf{y}_k - \mathbf{y}_{k-1}\|_2^2 + \frac{1}{4}A_{k-2}\mathbb{E}\|\mathbf{y}_{k-1} - \mathbf{y}_{k-2}\|_2^2.
\end{aligned}$$

Lemma 6.3.4. *Given a fixed $\mathbf{u} \in \mathcal{X}$, any $\mathbf{v} \in \mathbb{R}^m$, $\bar{\mathbf{y}}_0 = \mathbf{y}_0$, and $\mathbf{x}_1$ and $\mathbf{y}_1$ from Algorithm 6.3.1, we have*

$$A_1\mathrm{G}_1(\mathbf{u},\mathbf{v}) = a_1\langle \mathbf{A}^\top(\mathbf{y}_1 - \mathbf{y}_0), \mathbf{x}_1 - \mathbf{u}\rangle + \phi_0(\mathbf{u}) - \phi_0(\mathbf{x}_1) - \frac{1}{2}\|\mathbf{u}-\mathbf{x}_1\|_{\mathbf{\Lambda}}^2 - \frac{A_1}{2}\|\mathbf{v}-\mathbf{y}_1\|_2^2.$$

Combining the two lemmas, we now bound G_K and deduce our final result on the primal error.

Theorem 6.3.5. *[Main Result] Assume that $n \geq 4$. Given a matrix $\mathbf{A} \in \mathbb{R}_+^{m\times n}$, $\epsilon > 0$, an arbitrary $\mathbf{x}_0 \in \mathcal{X}$ and $\bar{\mathbf{y}}_0 = \mathbf{y}_0 = \mathbf{A}\mathbf{x}_0$, let $\mathbf{x}_k$ and A_k evolve according to SI-NNLS+ (Algorithm 6.3.1) for $k \geq 1$. For f defined in (6.2.2), define $\mathbf{x}^\star \in \operatorname{argmin}_{\mathbf{x}\geq\mathbf{0}} f(\mathbf{x})$. Then, for all $K \geq 2$, we have*

$$\mathbb{E}\Big[\langle \nabla f(\widetilde{\mathbf{x}}_K), \widetilde{\mathbf{x}}_K - \mathbf{x}^\star\rangle + \frac{1}{2}\|\mathbf{A}(\widetilde{\mathbf{x}}_K - \mathbf{x}^\star)\|^2\Big] \leq \frac{2\phi_0(\mathbf{x}^\star)}{A_K} = \frac{\|\mathbf{x}_0 - \mathbf{x}^\star\|_{\mathbf{\Lambda}}^2}{A_K}.$$

When $K \geq \frac{5}{2}n\log n$, we have $A_K \geq \frac{(K-\frac{5}{2}n\log n)^2}{36n^2}$. If $\phi_0(\mathbf{x}^\star) \leq |f(\mathbf{x}^\star)|$, then for $K \geq \frac{5}{2}n\log n + \frac{6n}{\sqrt{\epsilon}}$, we have $\mathbb{E}[f(\widetilde{\mathbf{x}}_K) - f(\mathbf{x}^\star)] \leq \varepsilon|f(\mathbf{x}^\star)|$. The total cost is $O\Big(nnz(\mathbf{A})\Big(\log n + \frac{1}{\sqrt{\epsilon}}\Big)\Big)$.

The assumption $\phi_0(\mathbf{x}^\star) \leq |f(\mathbf{x}^\star)|$ above is satisfied by $\mathbf{x}_0 = \mathbf{0}$ (c.f. Proposition 6.2.1 and Equation (6.3.6)). We reiterate that the reason $\|\mathbf{A}\|$ does not show up in the final bounds (thereby rendering our algorithm "scale-invariant") is because Proposition 6.2.1 allows bounding $\|\mathbf{x}_0 - \mathbf{x}^\star\|_{\mathbf{\Lambda}}^2$ by $|f(\mathbf{x}^\star)|$, where we crucially used the non-negativity of $\mathbf{A}$ and $\mathbf{x}$; this does not seem possible for general $\mathbf{A}$.

Remark 6.3.6. *SI-NNLS+ (Algorithm 6.3.1) and Theorem 6.3.5 also generalize to a mini-batch version. Increasing the batch size grows our bounds and number of data passes by a factor of at most square-root of the batch size s, by relating the spectral norm of the s columns of $\mathbf{A}$ corresponding to a batch to the Euclidean norms of individual columns of $\mathbf{A}$ from the same batch. However, due to efficient available implementations of vector operations, mini-batch variants of our algorithm with small batch sizes can have lower total runtimes on some datasets (see Section 6.5).*

6.4 Adaptive Restart

We now describe how SI-NNLS+ can be combined with adaptive restart to obtain linear convergence rate. To apply the restart strategy, we need suitable upper and lower bounds on the measure of convergence rate. Our measure of optimality is the *natural residual* $r(\mathbf{x}) = \|\mathbf{R}(\mathbf{x})\|_{\mathbf{\Lambda}}$ [MR94] for

$$\mathbf{R}(\mathbf{x}) = \mathbf{x} - \Pi_{\mathbb{R}^n_+}(\mathbf{x} - \mathbf{\Lambda}^{-1}\nabla f(\mathbf{x})) = \mathbf{x} - (\mathbf{x} - \mathbf{\Lambda}^{-1}\nabla f(\mathbf{x}))_+, \tag{6.4.1}$$

where $\Pi_{\mathbb{R}^n_+}$ is the projection operator onto $\mathbb{R}^n_+$ and $\mathbf{\Lambda}$ is as defined in Section 6.2. For $\mathbf{\Lambda} = \mathbf{I}$, $\mathbf{R}(\mathbf{x})$ is the natural map as defined in, e.g., [FP07]. Due to space constraints, we only state the main result of this section in the following theorem, while full technical details are deferred to Section E.1.4.

Theorem 6.4.1. *Given an error parameter $\varepsilon > 0$ and $\mathbf{x}_0 = \mathbf{0}$, consider the following algorithm $\mathcal{A}$:*

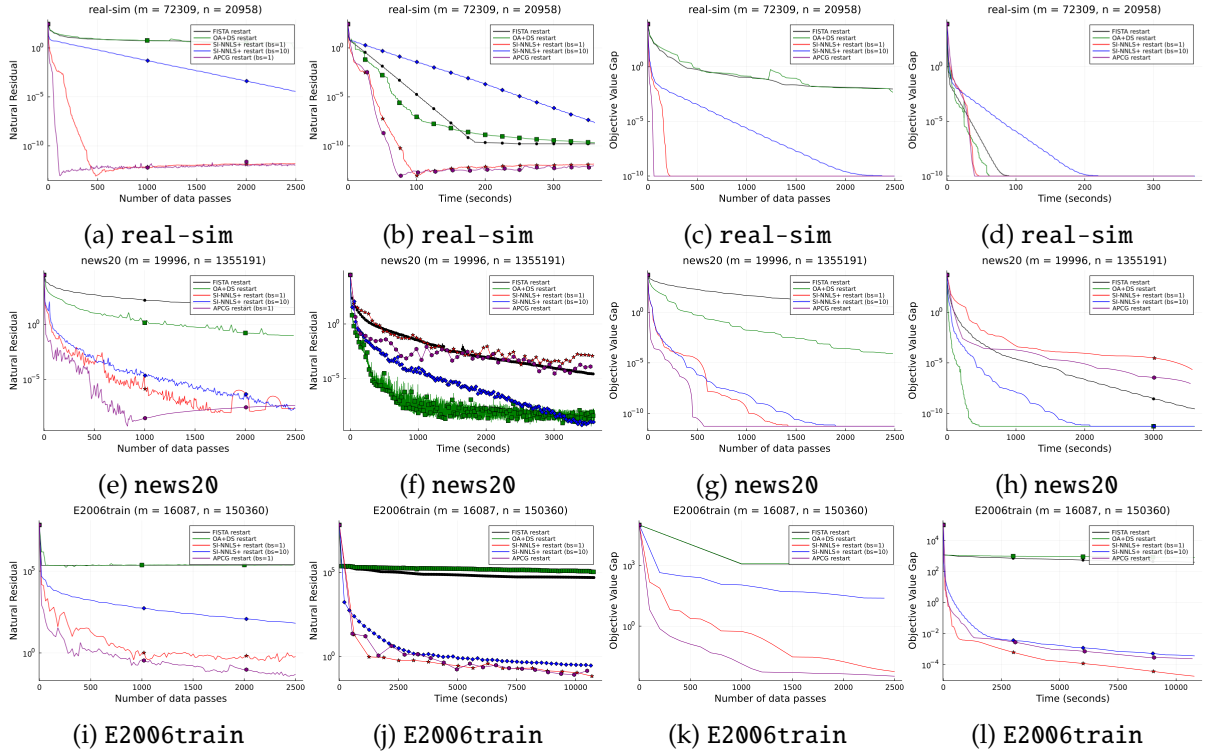
> *$\mathcal{A}$: **SI-NNLS+ with Restarts***
> *Initialize: $k = 1$.*
> *Initialize Lazy SI-NNLS+ at $\mathbf{x}_{k-1}$.*
> *Run Lazy SI-NNLS+ until the output $\widetilde{\mathbf{x}}_K^k$ satisfies $r(\widetilde{\mathbf{x}}_K^k) \leq \frac{1}{2} r(\mathbf{x}_{k-1})$.*
> *Restart Lazy SI-NNLS+ initializing at $\mathbf{x}_k = \widetilde{\mathbf{x}}_K^k$.*
> *Increment k.*
> *Repeat until $r(\widetilde{\mathbf{x}}_K^k) \leq \epsilon$.*

Then, the expected number of arithmetic operations of $\mathcal{A}$ is $O\Big(\mathrm{nnz}(\mathbf{A})\Big(\log n + \frac{\sqrt{n}}{\mu}\Big)\log\Big(\frac{r(\mathbf{x}_0)}{\epsilon}\Big)\Big)$. As a consequence, given $\bar{\epsilon} > 0$, the total expected number of arithmetic operations until a point with $f(\mathbf{x}) - f(\mathbf{x}^\star) \leq \bar{\epsilon}|f(\mathbf{x}^\star)|$ can be constructed by $\mathcal{A}$ is $O\Big(\mathrm{nnz}(\mathbf{A})\Big(\log n + \frac{\sqrt{n}}{\mu}\Big)\log\Big(\frac{n}{\mu\bar{\epsilon}}\Big)\Big)$.

6.5 Numerical Experiments and Discussion

We conclude this chapter by presenting the numerical performance of SI-NNLS+ and its restart versions (see the efficient implementation version in Algorithm E.2.1) against FISTA with restart [BT09, Nes13], a general-purpose large-scale optimization algorithm, OA+DS with restart designed by [KSD13] specifically for large-scale NNLS problems, and lazy implemented APCG [FR15] with restart. We use the same restart strategy for all the algorithms, proposed in Section 6.4.

As an accelerated algorithm, FISTA has the optimal $1/k^2$ convergence rate; OA+DS, while often efficient in practice, has only an asymptotic convergence guarantee. For FISTA, we compute the tightest Lipschitz constant (i.e., the spectral norm $\|\mathbf{A}\|$); for OA+DS, we follow the best practices



(a) `real-sim` (b) `real-sim` (c) `real-sim` (d) `real-sim`

(e) `news20` (f) `news20` (g) `news20` (h) `news20`

(i) `E2006train` (j) `E2006train` (k) `E2006train` (l) `E2006train`

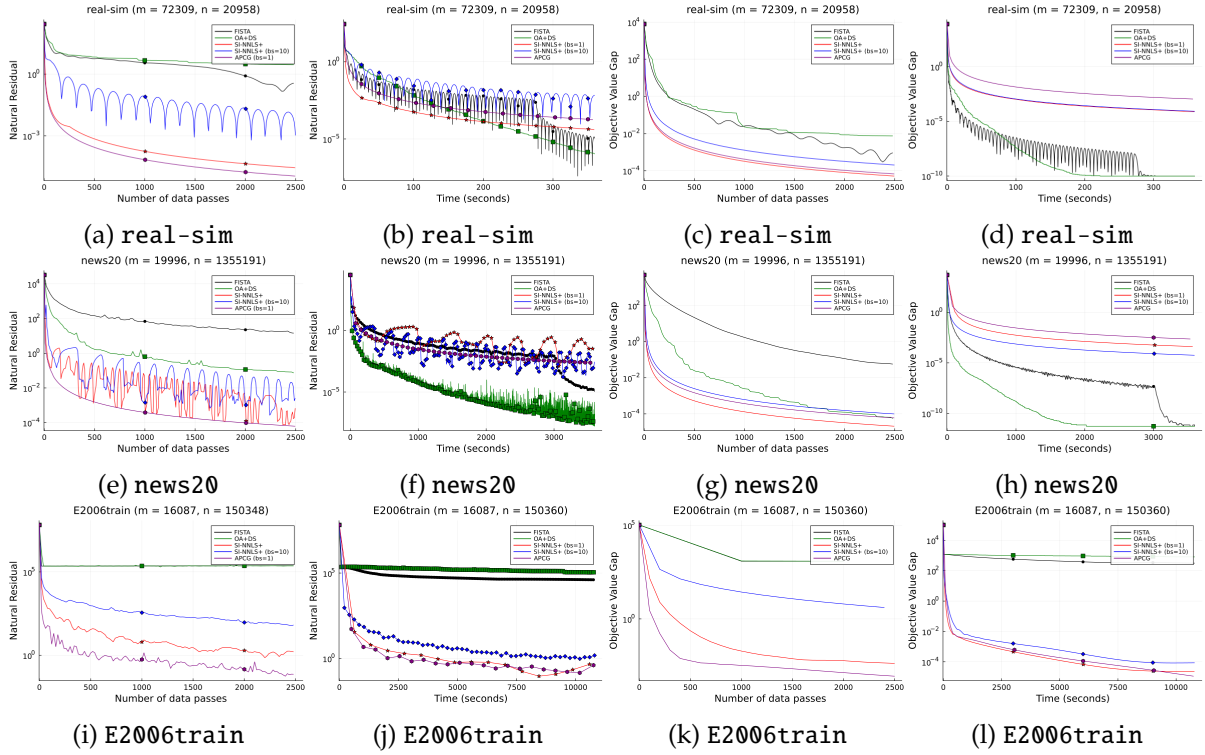
Figure 6.1: NNLS+ algorithms with restart on `real-sim`, `news20` and `E2006train` with spectral norm $3 \cdot 10^{-3}, 10^{-3}, 5 \cdot 10^{-6}$ and condition number $6 \cdot 10^{18}, 5 \cdot 10^{10}, 6 \cdot 10^{5}$, respectively.

laid out by [KSD13]. For our SI-NNLS+ algorithm and its restart version with batch size bs = 1, we follow Algorithm E.2.1 and the restart strategy in Section 6.4.[3] For the restart version with batch size larger than 1, we choose the best batch size in $\{10, 50, 300, 500\}$ and compute the block coordinate Lipschitz constants as the spectral norms of the corresponding block matrices. All algorithms were implemented in Julia and run on a server with 32 Intel(R) Xeon(R) Silver 4110 32-Core Processors.

We evaluated the performance of the algorithms on the large-scale sparse datasets `real-sim`, `news20`, and `E2006train` from the LibSVM library [CL11b]. Both `real-sim` and `news20` datasets have non-negative data matrices, but the labels may be negative. When there exist negative labels, it is possible for the elements of $\mathbf{A}^\top \mathbf{b}$ to be negative. In such cases, per the discussion from the introduction, we can simply remove the corresponding columns of $\mathbf{A}$ and solve an equivalent problem with smaller dimension. On the other hand, the data matrix in `E2006train` dataset is not non-negative, which means that this dataset does not satisfy the assumption required for the analysis of Algorithm 6.3.1. However, Algorithm 6.3.1 can still be run by keeping only the non-negativity constraints for primal updates. This example is provided solely for illustration of empirical performance.

Results. To compare all implemented algorithms, we plot the natural residual/objective value gap versus number of data passes/time in Figure 6.1 for all algorithms implemented with restart and in Figure 6.2 for all algorithms implemented without restart. As can be observed from the two figures, our proposed restart speeds up all the algorithms.

[3]The algorithm is implemented for the non-scaled version of the problem, (P); see Section 6.2. An implementation is in https://github.com/arcturus611/nnlr-2021.



(a) `real-sim` (b) `real-sim` (c) `real-sim` (d) `real-sim`

(e) `news20` (f) `news20` (g) `news20` (h) `news20`

(i) `E2006train` (j) `E2006train` (k) `E2006train` (l) `E2006train`

Figure 6.2: NNLS+ algorithms without restart on `real-sim`, `news20` and `E2006train` with spectral norm $3 \cdot 10^{-3}, 10^{-3}, 5 \cdot 10^{-6}$ and condition number $6 \cdot 10^{18}, 5 \cdot 10^{10}, 6 \cdot 10^{5}$, respectively.

Figure 6.1(a)-(d) shows that SI-NNLS+ is better than FISTA and OA+DS in terms of number of data passes on the `real-sim` dataset and better than APCG in terms of time. While the proposed restart strategy speeds up all the algorithms to linear convergence, variants of SI-NNLS+ remain competitive in all the settings. In terms of the performance of different variants of SI-NNLS+, with bs = 1, we have a much better coordinate Lipschitz constant than for bs = 10 and thus the case of bs = 1 dominates bs = 10 in terms of data passes. As FISTA and OA+DS take less time accessing the full dataset once, they have lower runtimes than SI-NNLS+ but are beaten by SI-NNLS+ with restart and bs = 1.

In Figure 6.1(e)-(h), on the `news20` dataset, in terms of number of data passes, restarted APCG and SI-NNLS+ with bs = 1 are dominant. However, as `news20` is a very sparse dataset, letting bs = 1 significantly increases the total time to access the full data once due to the overhead per iteration. As a result, single-coordinate methods have the worst runtimes, while restarted OA+DS is the fastest but SI-NNLS+ with bs = 10 remains competitive.

Figure 6.1(i)-(l) shows the performance comparison on the `E2006train` dataset. On this dataset, both restarted FISTA and restarted OA+DS ran for 4 hours without visibly reducing the function value. SI-NNLS+ outperforms FISTA and OA+DS in both number of data passes and time, and outperforms APCG in terms of time. Further experiments are left for future work.

Chapter 7

Computing Lewis Weights to High Precision

In Chapter 6, we saw an algorithm for structured ℓ_2 regression. In this chapter, we expand our focus from ℓ_2 to the ℓ_p setting. Specifically, we present an algorithm for computing approximate ℓ_p Lewis weights to high precision. Given a full-rank $\mathbf{A} \in \mathbb{R}^{m \times n}$ with $m \geq n$ and a scalar $p > 2$, our algorithm computes ε-approximate ℓ_p Lewis weights of $\mathbf{A}$ in $\widetilde{O}(p \log(1/\varepsilon))$ iterations; the cost of each iteration is linear in the input size plus the cost of computing the leverage scores of $\mathbf{DA}$ for diagonal $\mathbf{D} \in \mathbb{R}^{m \times m}$. Prior to our work, such a computational complexity was known only for $p \in (0, 4)$ in the work of Cohen and Peng. Combined with this result, our work yields the first polylogarithmic-depth polynomial-work algorithm for the problem of computing ℓ_p Lewis weights to high precision for all constant $p > 0$. An important consequence of this result is also the first polylogarithmic-depth polynomial-work algorithm for computing a nearly optimal self-concordant barrier for a polytope.

7.1 Introduction to Lewis Weights

In this chapter, we study the problem of computing the ℓ_p Lewis weights[1] of a matrix.

Definition 7.1.1. *[Lew78, CP15] Given a full-rank matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ with $m \geq n$ and a scalar $p \in (0, \infty)$, the Lewis weights of $\mathbf{A}$ are the entries of the unique[2] vector $\overline{w} \in \mathbb{R}^m$ satisfying the equation*

$$\overline{w}_i^{2/p} = a_i^\top (\mathbf{A}^\top \overline{\mathbf{W}}^{1-2/p} \mathbf{A})^{-1} a_i \text{ for all } i \in [m], \tag{7.1.1}$$

where a_i is the i'th row of matrix $\mathbf{A}$ and $\overline{\mathbf{W}}$ is the diagonal matrix with vector $\overline{w}$ on the diagonal.

Motivation. We contextualize our problem with a simpler, geometric notion. Given a set of m points $\{a_i\}_{i=1}^m \in \mathbb{R}^n$ (the rows of the preceding matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$), their *John ellipsoid* [Joh48] is the minimum[3] volume ellipsoid enclosing them. This ellipsoid finds use across experiment design and computational geometry [Tod16] and is central to certain cutting-plane methods [Vai89a, LS14], an algorithm fundamental to mathematical optimization (Section 7.1.3). It turns out that the John ellipsoid of a set of points $\{a_i\}_{i=1}^m \in \mathbb{R}^n$ is expressible [BV04b] as the solution to the following convex program, with the objective being a stand-in for the volume of the ellipsoid and the constraints encoding the requirement that each given point a_i lie *within* the ellipsoid:

$$\text{minimize}_{\mathbf{M} \geq 0} \det(\mathbf{M})^{-1}, \text{ subject to } a_i^\top \mathbf{M} a_i \leq 1, \text{ for all } i \in [m]. \tag{7.1.2}$$

[1] From hereon, we refer to these simply as "Lewis weights" for brevity.

[2] Existence and uniqueness was first proven by D.R.Lewis [Lew78], after whom the weights are named.

[3] The John ellipsoid may also refer to the maximal volume ellipsoid enclosed by the set $\{x : |x^\top a_i| \leq 1\}$, but in this chapter, we use the former definition.

The problem (7.1.2) may be generalized by the following convex program [Woj96, CP15], the generalization immediate from substituting $p = \infty$ in (7.1.3):

$$\text{minimize}_{\mathbf{M} \geq 0} \det(\mathbf{M})^{-1}, \text{ subject to } \sum_{i=1}^{m} (a_i^\top \mathbf{M} a_i)^{p/2} \leq 1. \tag{7.1.3}$$

Geometrically, (7.1.3) seeks the minimum volume ellipsoid with a bound on the $p/2$-norm of the distance of the points to the ellipsoid, and its solution $\mathbf{M}$ is the "Lewis ellipsoid" [CP15] of $\{a_i\}_{i=1}^m$.

The optimality condition of (7.1.3), written using $\overline{w} \in \mathbb{R}^m$ defined as $\overline{w}_i \stackrel{\text{def}}{=} (a_i^\top \mathbf{M} a_i)^{p/2}$, is equivalent to Equation (7.1.1), and this demonstrates that solving (7.1.3) is one approach to obtaining the Lewis weights of $\mathbf{A}$ (see [CP15]). This equivalence also underscores the fact that the problem of computing Lewis weights is a natural ℓ_p generalization of the problem of computing the John ellipsoid.

More broadly, Lewis weights are ubiquitous across statistics, machine learning, and mathematical optimization in diverse applications, of which we presently highlight two (see Section 7.1.3 for details). First, their interpretation as "importance scores" of rows of matrices makes them key to shrinking the row dimension of input data [DMM06]. Second, through their role in constructing self-concordant barriers of polytopes [LS14], variants of Lewis weights have found prominence in recent advances in the computational complexity of linear programming.

From a purely optimization perspective, Lewis weights may be viewed as the optimal solution to the following convex optimization problem (which is in fact essentially dual to (7.1.3)):

$$\overline{w} = \arg\min_{w \in \mathbb{R}^m_{>0}} \mathcal{F}(w) \stackrel{\text{def}}{=} -\log\det\left(\mathbf{A}^\top \mathbf{W} \mathbf{A}\right) + \frac{1}{1+\alpha} \mathbf{1}^\top w^{1+\alpha}, \text{ for } \alpha = \tfrac{2}{p-2}. \tag{7.1.4}$$

As elaborated in [CP15, LS14], the reason this problem yields the Lewis weights is that an appropriate scaling of its solution $\overline{w}$ transforms its optimality condition from $\overline{w}_i^\alpha = a_i^\top (\mathbf{A}^\top \overline{\mathbf{W}} \mathbf{A})^{-1} a_i$ to Equation (7.1.1). The problem (7.1.4) is a simple and natural one and, in the case of $\alpha = 1$ (corresponding to the John ellipsoid), has been the subject of study for designing new optimization methods [Tod16].

In summary, Lewis weights naturally arise as generalizations of extensively studied problems in convex geometry and optimization. This, coupled with their role in machine learning, makes understanding the complexity of computing Lewis weights, i.e., solving (7.1.4), a fundamental problem.

Our Goal. We aim to design *high-precision* algorithms for computing *ϵ-approximate* Lewis weights, i.e., a vector $w \in \mathbb{R}^m$ satisfying

$$w_i \approx_\epsilon \overline{w}_i, \text{ for all } i \in [m], \text{ where } \overline{w} \text{ is defined in } \textit{Equation (7.1.1)} \text{ and } (7.1.4). \tag{7.1.5}$$

where $a \approx_\epsilon b$ is used to denote $(1-\epsilon)a \leq b \leq (1+\epsilon)a$. To this end, we design algorithms to solve the convex program (7.1.4) to $\widetilde{\varepsilon}$-additive accuracy for an appropriate $\widetilde{\varepsilon} = \text{poly}(\epsilon, n)$, which we prove suffices in Lemma 7.2.1.

By a "high-precision" algorithm, we mean one with a runtime polylogarithmic in ϵ. We emphasize that for several applications such as randomized sampling [CP15], *approximate* Lewis weights suffice; however, we believe that high-precision methods such as ours enrich our understanding of the structure of the optimization problem (7.1.4). Further, as stated in Theorem 7.1.5, such methods yield new runtimes for directly computing a near-optimal self-concordant barrier for polytopes.

We use *number of leverage score computations* as the complexity measure of our algorithms. Our

choice is a result of the fact that leverage scores of appropriately scaled matrices appear in both $\nabla\mathcal{F}(w)$ (see Lemma 7.2.3) and in the verification of correctness of Lewis weights. This measure of complexity stresses the *number of iterations* rather than the details of iteration costs (which depend on exact techniques used for leverage core computation, e.g., fast matrix multiplication) and is consistent with many prior algorithms (see Table 7.1).

Prior Results. The first polynomial-time algorithm for computing Lewis weights was presented by [CP15] and performed only $\widetilde{O}_p(\log(1/\epsilon))$[4] leverage score computations. However, *their result holds only for* $p \in (0, 4)$. We explain the source of this limited range in Section 7.1.2.

In comparison, for $p \geq 4$, existing algorithms are slower: the algorithms by [CP15], [Lee16], and [LS14] perform $\widetilde{\Omega}(n)$, $\widetilde{O}(1/\epsilon)$, and $\widetilde{O}(\sqrt{n})$ leverage score computations, respectively. [CP15] also gave an algorithm with total runtime $O(\frac{1}{\epsilon}\text{nnz}\mathbf{A} + c_p n^{O(p)})$. Of note is the fact that the algorithms with runtimes polynomial in $1/\epsilon$ ([Lee16, CP15]) satisfy the weaker approximation condition $\overline{w}_i^{2/p} \approx_\epsilon a_i^\top(\mathbf{A}^\top\overline{\mathbf{W}}^{1-2/p}\mathbf{A})^{-1}a_i$, which is in fact implied by our condition in Equation (7.1.5).

We display these runtimes in Table 7.1, assuming that the cost of a leverage score computation is $O(mn^2)$ (which, we reiterate, may be reduced through the use of fast matrix multiplication). In terms of the number of leverage score computations, Table 7.1 highlights the contrast between the *polylogarithmic* dependence on input size and accuracy for $p \in (0, 4)$ and *polynomial* dependence on these factors for $p \geq 4$. *The motivation behind our chapter is to close this gap.*

7.1.1 Our Contribution

We design *an algorithm that computes Lewis weights to high precision for all* $p > 2$ *using only* $\widetilde{O}_p(\log(1/\epsilon))$ *leverage score computations*. Together with [CP15]'s result for $p \in (0, 4)$, our result therefore completes the picture on a near-optimal reduction from leverage scores to Lewis weights for all $p > 0$.

Theorem 7.1.2 (Main Theorem (Parallel))**.** *Given a full-rank matrix* $\mathbf{A} \in \mathbb{R}^{m\times n}$ *and* $p \geq 4$*, we can compute (Algorithm 7.2.1 and Algorithm 7.2.2) its* ϵ*-approximate Lewis weights Equation (7.1.5) in* $O(p^3 \log(mp/\epsilon))$ *iterations*[5]*. Each iteration computes the leverage scores of a matrix* $\mathbf{DA}$ *for a diagonal matrix* $\mathbf{D}$*. The total runtime is* $O(p^3mn^2 \log(mp/\epsilon))$*, with* $O(p^3 \log(mp/\epsilon)\log^2(m))$ *depth.*

Theorem 7.1.2 is attained by a parallel algorithm for computing Lewis weights that consists of polylogarithmic rounds of leverage score computations and therefore has polylogarithmic-depth, a result that was not known prior to this work.

Theorem 7.1.3 (Main Theorem (Sequential))**.** *Given a full-rank matrix* $\mathbf{A} \in \mathbb{R}^{m\times n}$ *and* $p \geq 4$*, we can compute (Algorithm 7.2.1 and Algorithm 7.2.3) its* ϵ*-approximate Lewis weights Equation (7.1.5) in* $O(pm \log(mp/\epsilon))$ *iterations. Each iteration computes the leverage score of one row of* $\mathbf{DA}$ *for a diagonal matrix* $\mathbf{D}$*. The total runtime is* $O(pmn^2 \log(mp/\epsilon))$*.*

Remark 7.1.4. *The solution to (7.1.3) characterizes a "Lewis ellipsoid," and the* ℓ_∞ *Lewis ellipsoid of* $\mathbf{A}$ *is precisely its John ellipsoid. After symmetrization [Tod16], computing the John ellipsoid is equivalent to solving a linear program (LP). Therefore, computing Lewis weights in* $O(\log(mp/\epsilon))$ *iterations would imply a polylogarithmic-depth algorithm for solving LPs, which, given the current* $O(\sqrt{n})$ *depth [LS14], would be a significant breakthrough in the field of optimization. We therefore believe that it would be difficult to remove the polynomial dependence on* p *in our runtime.*

[4] We use O_p to hide a polynomial in p and $\widetilde{O}$ and $\widetilde{\Omega}$ to hide factors polylogarithmic in p, n, and m.

[5] Our algorithms work for all $p > 2$, as can be seen in our proof in Section 7.3.1. However, for $p \in (2, 4)$, the algorithm of [CP15] is faster, and therefore, in our main theorems, we state runtimes only for $p \geq 4$.

Table 7.1: Runtime comparison for computing Lewis weights. Results with asterisks use a weaker notion of approximation than our chapter Equation (7.1.1). All dependencies on n in the running times of these methods can be improved using fast matrix multiplication.

Authors	Range of p	Number of Leverage Score Computations/Depth	Total Runtime
[CP15]	$p \in (0,4)$	$O\left(\frac{1}{1-\lvert 1-p/2\rvert} \cdot \log\left(\frac{\log(m)}{\epsilon}\right)\right)$	$O\left(\frac{1}{1-\lvert 1-p/2\rvert} \cdot mn^2 \cdot \log\left(\frac{\log(m)}{\epsilon}\right)\right)$
[CP15]	$p \geq 4$	$\Omega(n)$	$\Omega(mn^3 \cdot \log\left(\frac{m}{\epsilon}\right))$
[CP15]*	$p \geq 4$	not applicable	$O\left(\frac{\text{nnz}(\mathbf{A})}{\epsilon} + c_p n^{O(p)}\right)$
[Lee16]*	$p \geq 4$	$O\left(\frac{1}{\epsilon} \cdot \log(m/n)\right)$	$O\left(\left(\frac{\text{nnz}(\mathbf{A})}{\epsilon} + \frac{n^3}{\epsilon^3}\right) \cdot \log(m/n)\right)$
[LS14]	$p \geq 4$	$O(p^2 \cdot n^{1/2} \cdot \log\left(\frac{1}{\epsilon}\right))$	$O(p^2 \cdot mn^{2.5} \cdot \text{poly}\log\left(\frac{m}{\epsilon}\right))$
Theorem 7.1.2	$p \geq 4$	$O(p^3 \cdot \log\left(\frac{mp}{\epsilon}\right))$	$O(p^3 \cdot mn^2 \cdot \log\left(\frac{mp}{\epsilon}\right))$

7.1.2 Overview of Approach

Before presenting our algorithm, we describe obstacles to directly extending previous work on the problem for $p \in (0,4)$ to the case $p \geq 4$. For $p \in (0,4)$, [CP15, LS14] design algorithms that, with a single computation of leverage scores, make constant (dependent on p) multiplicative progress on error (such as function error or distance to optimal point), thus attaining runtimes polylogarithmic in ϵ. However, these methods crucially rely on *contractive properties* that do *not* necessarily hold for $p \geq 4$.

For example, one of the algorithms in [CP15] starts with a vector $v \approx_c \overline{w}$, where $\overline{w}$ is the vector of true Lewis weights and c some constant. Consequently, we have $(a_i^\top (\mathbf{A}^\top \mathbf{V}^{1-2/p} \mathbf{A})^{-1} a_i)^{p/2} \approx_{c^{|p/2-1|}} (a_i^\top (\mathbf{A}^\top \overline{\mathbf{W}}^{1-2/p} \mathbf{A})^{-1} a_i)^{p/2}$. Due to this map being a contraction for $|p/2-1| < 1$, or equivalently, for $p \in (0,4)$, $O(\log(\log n))$ recursive calls to it give Lewis weights for $p < 4$, but the contraction - and, by extension, this method - does not immediately extend to the setting $p \geq 4$.

Prior algorithms for $p \geq 4$, therefore, resort to alternate optimization techniques. [CP15] frames Lewis weights computation as determinant maximization (7.1.3) (see Section F.4) and applies cutting plane methods [GLS81b, LSW15]. [Lee16] uses mirror descent, and [LS14] uses homotopy methods. These approaches yield runtimes with $\text{poly}(n)$ or $\text{poly}(\frac{1}{\epsilon})$ leverage score computations, and therefore, attaining runtimes of $\text{polylog}(1/\epsilon)$ leverage score computations requires rethinking the algorithm.

Our Approach. As stated in Section 7.1, to obtain ϵ-approximate Lewis weights for $p \geq 4$, we compute a w that satisfies $\mathcal{F}(\overline{w}) \leq \mathcal{F}(w) \leq \mathcal{F}(\overline{w}) + \widetilde{\varepsilon}$, where $\mathcal{F}$ and $\overline{w}$ are as defined in (7.1.4) and $\widetilde{\varepsilon} = O(\text{poly}(n, \epsilon))$. In light of the preceding bottlenecks in prior work, we circumvent techniques that directly target constant multiplicative progress (on some potential) in each iteration.

Our main technical insight is that *when the leverage scores for the current weight $w \in \mathbb{R}^n_{>0}$ satisfy a certain technical condition (Inequality 7.1.6)*, it is indeed possible to update w to get multiplicative decrease in function error ($\mathcal{F}(w) - \mathcal{F}(\overline{w})$), thus resulting in our target runtime. To turn this insight into an algorithm, we design a corrective procedure that ensures that Inequality 7.1.6 is always satisfied: in other words, whenever Inequality 7.1.6 is *violated*, this procedure updates w so that the new w *does* satisfy Inequality 7.1.6, setting the stage for the aforementioned multiplicative progress. An important additional property of this procedure is that it does *not* increase the objective function and is therefore in keeping with our goal of minimizing (7.1.4).

Specifically, the technical condition that our geometric decrease in function error hinges on is

$$\max_{i\in[m]} \frac{a_i^\top (\mathbf{A}^\top \mathbf{W}\mathbf{A})^{-1} a_i}{w_i^\alpha} \leq 1+\alpha\,. \tag{7.1.6}$$

This ratio follows naturally from the gradient and Hessian of the function objective (see Lemma 7.2.3). Our algorithm's update rule to solve (7.1.4) is obtained from minimizing a second-order approximation to the objective at the current point, and the condition specified in Inequality 7.1.6 allows us to relate the progress of a type of quasi-Newton step to lower bounds on the progress there is to make, which is critical to turning a runtime of poly$(1/\epsilon)$ into polylog$(1/\epsilon)$ (Lemma 7.2.5).

The process of updating w so that Inequality 7.1.6 goes from being violated to being satisfied corresponds, geometrically, to sufficiently rounding the ellipsoid $\mathcal{E}(w) = \{x : x^\top \mathbf{A}^\top \mathbf{W}\mathbf{A}x \leq 1\}$; specifically, the updated ellipsoid satisfies $\mathcal{E}(w) \subseteq \{\|\mathbf{W}^{\frac{1}{2-p}}\mathbf{A}x\|_\infty \leq \sqrt{1+\alpha}\}$ (see Section F.3), and this is the reason we use the term "rounding" to describe our corrective procedure to get w to satisfy Inequality 7.1.6 and the term "rounding condition" to refer to Inequality 7.1.6.

We develop two versions of rounding: a parallel method and a sequential one that has an improved dependence on p. Each version is based on the principles that (1) one can increase those entries of w at which the rounding condition Inequality 7.1.6 does not hold *while decreasing* the objective value, and (2) the vector w obtained after this update is closer to satisfying Inequality 7.1.6.

We believe that such a principle of identifying a technical condition needed for fast convergence and the accompanying rounding procedures could be useful in other optimization problems. Additionally, we develop Algorithm 7.5.1, which, by varying the step sizes in the update rule, maintains Inequality 7.1.6 as invariant, thereby eliminating the need for a separate rounding and progress steps.

7.1.3 Applications and Related Work

We elaborate here on the applications of Lewis weights we briefly alluded to in Section 7.1. While for many applications (such as pre-processing in optimization [CP15]) approximate weights suffice, solving regularized D-optimal and computing $\tilde{O}(n)$ self-concordant barriers to high precision do use high precision Lewis weights.

Pre-processing in optimization. Lewis weights are used as scores to sample rows of an input tall data matrix so the ℓ_p norms of the product of the matrix with vectors are preserved. They have been used in row sampling algorithms for data pre-processing [DMM06, DMIMW12, LMP13, CLM+15b], for computing dimension-free strong coresets for k-median and subspace approximation [SW18], and for fast tensor factorization in the streaming model [CCDS20]. Lewis weights are also used for ℓ_1 regression, a popular model in machine learning used to capture robustness to outliers, in: [DLS18] for stochastic gradient descent pre-conditioning, [LWYZ20] for quantile regression, and [BDM+20] to provide algorithms for linear algebraic problems in the sliding window model.

John ellipsoid and D-optimal design. As noted in Remark 7.1.4, a fast algorithm for Lewis weights could yield faster algorithms for computing John ellipsoid, a problem with a long history of work [Kha96, SF04, KY05, DAST08, CCLY19, ZF20]. It is known [Tod16] that the John ellipsoid problem is dual to the (relaxed) D-optimal experiment design problem [Puk06]. D-optimal design seeks to select a set of linear experiments with the largest confidence ellipsoid for its least-square estimator [AZLSW17, MSTX19, SX20].

Our problem (7.1.4) is equivalent to $\frac{p}{p-2}$-regularized D-optimal design, which can be interpreted as enforcing a polynomial experiment cost: viewing w_i as the fraction of resources allocated to

experiment i, each w_i is penalized by $w_i^{\frac{p}{p-2}}$. This regularization also appears in fair packing and fair covering problems [MSZ16, DFO20b] from operations research.

Self-concordance. Self-concordant barriers are fundamental in convex optimization [NN94], combinatorial optimization [LS14], sampling [KN09, LLV20], and online learning [AHR08]. Although there are (nearly) optimal self-concordant barriers for any convex set [NN94, BE15, LY18], computing them involves sampling from log-concave distributions, itself an expensive process with a poly$(1/\epsilon)$ runtime. [LS14] shows how to construct nearly optimal barriers for polytopes using Lewis weights. Unfortunately, doing so still requires polynomial-many steps to compute these weights; [LS14] bypass this issue by showing it suffices to work with Lewis weights for $p \approx 1$. In this chapter, we show how to compute Lewis weights by computing leverage scores of polylogarithmic-many matrices. This gives the first nearly optimal self-concordant barrier for polytopes that can be evaluated to high accuracy with depth polylogarithmic in the dimension.

Theorem 7.1.5 (Applying Theorem 7.1.2 to [LS14, Section 5]). *Given a non-empty polytope $P = \{x \in \mathbb{R}^n \mid \mathbf{A}x > b\}$ for full rank $\mathbf{A} \in \mathbb{R}^{m \times n}$, there is a $O(n \log^5 m)$-self concordant barrier ψ for P such that for any $\epsilon > 0$ and $x \in P$, in $O(mn^{\omega-1} \log^3 m \log(m/\epsilon))$-work and $O(\log^3 m \log(m/\epsilon))$-depth, we can compute $g \in \mathbb{R}^n$ and $\mathbf{H} \in \mathbb{R}^{n \times n}$ with $\|g - \nabla\psi(x)\|_{\nabla^2\psi(x)^{-1}} \leq \epsilon$ and $\nabla^2\psi(x) \preceq \mathbf{H} \preceq O(\log m)\nabla^2\psi(x)$. With an additional $O(m^{\omega+o(1)})$ work, $\mathbf{H} \in \mathbb{R}^{n \times n}$ with $(1-\epsilon)\nabla^2\psi(x) \preceq \mathbf{H} \preceq O(1+\epsilon)\nabla^2\psi(x)$ can be computed as well.*

7.1.4 Notation and Preliminaries

We use $\mathbf{A}$ to denote our full-rank $m \times n$ ($m \geq n$) real-valued input matrix and $\overline{w} \in \mathbb{R}^m$ to denote the vector of Lewis weights of $\mathbf{A}$, as defined in Equation (7.1.1) and (7.1.4). All matrices appear in boldface uppercase and vectors in lowercase. For any vector (say, σ), we use its uppercase boldfaced form ($\mathbf{\Sigma}$) to denote the diagonal matrix $\mathbf{\Sigma}_{ii} = \sigma_i$. For a matrix $\mathbf{M}$, the matrix $\mathbf{M}^{(2)}$ is the Schur product (entry-wise product) of $\mathbf{M}$ with itself. For matrices $\mathbf{A}$ and $\mathbf{B}$, we use $\mathbf{A} \succeq \mathbf{B}$ to mean $\mathbf{A} - \mathbf{B}$ is positive-semidefinite. For vectors a and b, the inequality $a \leq b$ applies entry-wise. We use e_i to denote the i'th standard basis vector. We define $[n] \stackrel{\text{def}}{=} \{1, 2, \ldots, n\}$. As in (7.1.4), since we defined $\alpha \stackrel{\text{def}}{=} \frac{2}{p-2}$, the ranges of $p \in (2, 4)$ and $p \geq 4$ translate to $\alpha > 1$ and $\alpha \in (0, 1]$, respectively. From hereon, we work with α. We also define $\bar{\alpha} = \max\{1, \alpha\}$. For a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $w \in \mathbb{R}^m_{>0}$, we define the projection matrix $\mathbf{P}(w) \stackrel{\text{def}}{=} \mathbf{W}^{1/2}\mathbf{A}(\mathbf{A}^\top\mathbf{W}\mathbf{A})^{-1}\mathbf{A}^\top\mathbf{W}^{1/2} \in \mathbb{R}^{m \times m}$. The quantity $\mathbf{P}(w)_{ii}$ is precisely the leverage score of the i'th row of $\mathbf{W}^{1/2}\mathbf{A}$:

$$\sigma_i(w) \stackrel{\text{def}}{=} w_i \cdot a_i^\top(\mathbf{A}^\top\mathbf{W}\mathbf{A})^{-1}a_i. \tag{7.1.7}$$

Fact 7.1.6 ([LS14]). *For all $w \in \mathbb{R}^m_{>0}$ we have that $0 \leq \sigma_i(w) \leq 1$ for all $i \in [m]$, $\sum_{i \in [m]} \sigma_i(w) \leq n$, and $\mathbf{0} \preceq \mathbf{P}(w)^{(2)} \preceq \mathbf{\Sigma}(w)$.*

7.2 Our Algorithm

We present Algorithm 7.2.1 to compute an $\widetilde{\varepsilon}$-additive solution to (7.1.4). We first provide the following definitions that we frequently refer to in our algorithm and analysis. Given $\alpha > 0$ and $w \in \mathbb{R}^m_{>0}$, the i'th coordinate of $\rho(w) \in \mathbb{R}^m$ is

$$\rho_i(w) \stackrel{\text{def}}{=} \frac{\sigma_i(w)}{w_i^{1+\alpha}}. \tag{7.2.1}$$

Based on this quantity, we define the following procedure, derived from approximating a quasi-Newton update on the objective $\mathcal{F}$ from (7.1.4):

$$[\textbf{Descent}(w, \mathrm{C}, \eta)]_i \stackrel{\text{def}}{=} w_i \left[1 + \eta_i \cdot \frac{\rho_i(w) - 1}{\rho_i(w) + 1}\right] \text{ for all } i \in \mathrm{C} \subseteq \{1, 2, \ldots, m\}. \tag{7.2.2}$$

Using these definitions, we can now describe our algorithm. Depending on whether the following condition ("rounding condition") holds, we run either **Descent**(·) or **Round**(·) in each iteration:

$$\rho_{\max}(w) \stackrel{\text{def}}{=} \max_{i \in [m]} \rho_i(w) \leq 1 + \alpha. \tag{7.2.3}$$

Specifically, if Inequality 7.2.3 is *not* satisfied, we run **Round**(·), which returns a vector that *does* satisfy it without increasing the objective value. We design two versions of **Round**(·), one parallel (Algorithm 7.2.2) and one sequential (Algorithm 7.2.3), with the sequential algorithm having an improved dependence on α, to update the coordinates violating Inequality 7.2.3. We apply one extra step of rounding to the vector returned after $\mathcal{T}_{\text{total}}$ iterations of Algorithm 7.2.1 and transform it appropriately to obtain our final output. In the following lemma (proved in Section F.2), we justify that this output is indeed the solution to Equation (7.1.5).

Lemma 7.2.1 (Lewis Weights from Optimization Solution)**.** *Let $w \in \mathbb{R}^m_{>0}$ be a vector at which the objective (7.1.4) is $\widetilde{\varepsilon}$-suboptimal in the additive sense for $\widetilde{\varepsilon} = \frac{\alpha^8 \epsilon^4}{(25m(\sqrt{n}+\alpha)(\alpha+\alpha^{-1}))^4}$, i.e., $\mathcal{F}(\overline{w}) \leq \mathcal{F}(w) \leq \mathcal{F}(\overline{w}) + \widetilde{\varepsilon}$. Further assume that w satisfies the rounding condition: $\rho_{\max}(w) \leq 1 + \alpha$. Then, the vector $\widehat{w}$ defined as $\widehat{w}_i = (a_i^\top (\mathbf{A}^\top \mathbf{W} \mathbf{A})^{-1} a_i)^{1/\alpha}$ satisfies $\widehat{w}_i \approx_\epsilon \overline{w}_i$ for all $i \in [m]$, thus achieving the goal spelt out in Equation (7.1.5).*

Algorithm 7.2.1 Lewis Weight Computation Meta-Algorithm

Input: Matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, parameter $p > 2$, accuracy ϵ
Output: Vector $\widehat{w} \in \mathbb{R}^m_{>0}$ that satisfies Equation (7.1.5)

1: For all $i \in [m]$, initialize $w_i^{(0)} = \frac{n}{m}$.
2: Set $\alpha = \frac{2}{p-2}$, $\bar{\alpha} = \max(\alpha, 1)$, $\widetilde{\varepsilon} = \frac{\alpha^8 \epsilon^4}{(25m(\sqrt{n}+\alpha)(\alpha+\alpha^{-1}))^4}$, and $\mathcal{T}_{\text{total}} = O(\max(\alpha^{-1}, \alpha) \log(m/\widetilde{\varepsilon}))$.
3: **for** $k = 1, 2, 3, \ldots, \mathcal{T}_{\text{total}}$ **do**
4: $\quad \widetilde{w}^{(k)} \leftarrow \textbf{Round}(w^{(k-1)}, \mathbf{A}, \alpha)$ ▷ Invoke Algorithm 7.2.2 (parallel) or Algorithm 7.2.3 (sequential)
5: $\quad w^{(k)} \leftarrow \textbf{Descent}(\widetilde{w}^{(k)}, [m], \frac{1}{3\bar{\alpha}}\mathbf{1})$ ▷ See Equation (7.2.2) and Lemma 7.2.2
6: **end for**
7: Set $w_{\mathrm{R}} \leftarrow \textbf{Round}(w^{(\mathcal{T}_{\text{total}})}, \mathbf{A}, \alpha)$
8: Return $\widehat{w} \in \mathbb{R}^m_{>0}$, where $\widehat{w}_i = (a_i^\top (\mathbf{A}^\top \mathbf{W}_{\mathrm{R}} \mathbf{A})^{-1} a_i)^{1/\alpha}$. ▷ See Section F.2

7.2.1 Analysis of Descent(·)

We first analyze **Descent**(·) since it is common to both the parallel and sequential algorithms.

Lemma 7.2.2 (Iteration Complexity of **Descent**(·))**.** *Each iteration of **Descent**(·) (described in Equation (7.2.2)) decreases the value of $\mathcal{F}$. Assuming that **Round**(·) does not increase the value of the objective in (7.1.4), for any given accuracy parameter $0 < \widetilde{\varepsilon} < 1$, the number of **Descent**(·) steps that Algorithm 7.2.1 performs before achieving $\mathcal{F}(w) \leq \mathcal{F}(\overline{w}) + \widetilde{\varepsilon}$ is given by the following bound:*

$$\mathcal{T}_{total} = O(\max(\alpha^{-1}, \alpha) \log(m/\widetilde{\varepsilon})).$$

Algorithm 7.2.2 RoundParallel$(w, \mathbf{A}, \alpha)$

Input: Vector $w \in \mathbb{R}^m_{>0}$, matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, parameter $\alpha > 0$ **Output**: Vector $w \in \mathbb{R}^m_{>0}$ satisfying Inequality 7.2.3

1: Define $\rho(w)$ as in (7.2.1)
2: **while** $\mathrm{C} = \{i \mid \rho_i(w) > 1 + \alpha\} \neq \emptyset$ **do**
3: $\quad w \leftarrow$ **Descent**$(w, \mathrm{C}, \frac{1}{3\bar{a}}\mathbf{1})$ ▷ See Section 7.3
4: **end while**
5: Return w

Algorithm 7.2.3 RoundSequential$(w, \mathbf{A}, \alpha)$

Input: Vector $w \in \mathbb{R}^m_{>0}$, matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, parameter $\alpha > 0$
Output: Vector $w \in \mathbb{R}^m_{>0}$ satisfying Inequality 7.2.3

1: Define $\rho(w)$ as in (7.2.1) and $\sigma(w)$ as in (7.1.7)
2: Define $\mathrm{C} = \{i \mid \rho_i(w) \geq 1\}$
3: **for** $i \in \mathrm{C}$ **do**
4: $\quad w_i \leftarrow w_i(1 + \delta_i)$, where δ_i solves $\rho_i(w) = (1 + \delta_i \sigma_i(w))(1 + \delta_i)^\alpha$ ▷ see Section 7.4
5: **end for**
6: Return w

As is often the case to obtain such an iteration complexity, we prove Lemma 7.2.2 by incorporating the maximum sub-optimality in function value (Lemma 7.2.5) and the initial error bound (Lemma 7.2.4) into the inequality describing minimum function progress (Lemma 7.2.6). The assumption that **Round**$(\cdot)$ does not increase the value of the objective is justified in Lemma 7.3.1.

Since our algorithm leverages quasi-Newton steps, we begin our analysis by stating the gradient and Hessian of the objective in (7.1.4) as well as the error at the initial vector $w^{(0)}$, as measured against the optimal function value. The Hessian below is positive semidefinite when $\alpha \geq 0$ (equivalently, when $p \geq 2$) and not necessarily so otherwise. Consequently, the objective is convex for $\alpha \geq 0$, and we therefore consider only this setting throughout.

Lemma 7.2.3 (Gradient and Hessian)**.** *For any $w \in \mathbb{R}^m_{>0}$, the objective in (7.1.4), $\mathcal{F}(w) = -\log\det(\mathbf{A}^\top \mathbf{W}\mathbf{A}) + \frac{1}{1+\alpha}\mathbf{1}^\top w^{1+\alpha}$, has gradient and Hessian given by the following expressions.*

$$[\nabla \mathcal{F}(w)]_i = w_i^{-1} \cdot (w_i^{1+\alpha} - \sigma_i(w)) \text{ and } \nabla^2 \mathcal{F}(w) = \mathbf{W}^{-1}\mathbf{P}(w)^{(2)}\mathbf{W}^{-1} + \alpha \mathbf{W}^{\alpha - 1}.$$

Lemma 7.2.4 (Initial Sub-Optimality)**.** *At the start of Algorithm 7.2.1, the value of the objective of (7.1.4) differs from the optimum objective value as $\mathcal{F}(w^{(0)}) \leq \mathcal{F}(\overline{w}) + n\log(m/n)$.*

Minimum Progress and Maximum Sub-optimality in Descent$(\cdot)$

We first prove an upper bound on objective sub-optimality, necessary to obtain a runtime polylogarithmic in $1/\epsilon$. Often, to obtain such a rate, the bound involving objective sub-optimality has a quadratic term derived from the Hessian; our lemma is somewhat non-standard in that it uses only the convexity of $\mathcal{F}$. Note that this lemma crucially uses Inequality 7.2.3.

Lemma 7.2.5 (Objective Sub-optimality)**.** *Suppose $w \in \mathbb{R}^m_{>0}$ and $\rho_{\max}(w) \leq 1 + \alpha$. Then the value of the*

objective of (7.1.4) *at w differs from the optimum objective value as follows.*

$$\mathcal{F}(w) - \mathcal{F}(\overline{w}) \le \sum_{i\in[m]} \frac{w_i^{1+\alpha}}{1+\alpha}\left(1 + \frac{\rho_i(w)}{\alpha}\right)(\rho_i(w)-1)^2 \le 5\max\{1,\alpha^{-1}\}\sum_{i\in[m]} w_i^{1+\alpha}\frac{(\rho_i(w)-1)^2}{\rho_i(w)+1}.$$

Proof. Since $g(w) \stackrel{\text{def}}{=} -\log\det(\mathbf{A}^\top\mathbf{W}\mathbf{A})$ is convex and $[\nabla g(w)]_i = -w_i^{-1}\sigma_i(w)$, we have

$$g(\overline{w}) \ge g(w) + \nabla g(w)^\top(\overline{w}-w) = g(w) + \sum_{i\in[m]}\left(-\frac{\sigma_i(w)\overline{w}_i}{w_i} + \sigma_i(w)\right),$$

and therefore,

$$\begin{aligned}\mathcal{F}(\overline{w}) - \mathcal{F}(w) &= g(\overline{w}) - g(w) + \frac{1}{1+\alpha}\sum_{i\in[m]}\left([\overline{w}]_i^{1+\alpha} - w_i^{1+\alpha}\right)\\ &\ge \sum_{i\in[m]} c_i \text{ where } c_i \stackrel{\text{def}}{=} -\frac{\sigma_i(w)\overline{w}_i}{w_i} + \sigma_i(w) + \frac{1}{1+\alpha}\left([\overline{w}]_i^{1+\alpha} - w_i^{1+\alpha}\right).\end{aligned}$$

To prove the claim, it suffices to bound each c_i from below. First, note that

$$\begin{aligned}c_i &\ge \min_{v\ge 0} -\frac{v\cdot\sigma_i(w)}{w_i} + \sigma_i(w) + \frac{1}{1+\alpha}\left(v^{1+\alpha} - w_i^{1+\alpha}\right) = -\frac{\alpha}{1+\alpha}\left(\frac{\sigma_i(w)}{w_i}\right)^{1+\frac{1}{\alpha}} + \sigma_i(w) - \frac{w_i^{1+\alpha}}{1+\alpha}\\ &= \frac{w_i^{1+\alpha}}{1+\alpha}\left[-\alpha\rho_i(w)^{1+\frac{1}{\alpha}} + (1+\alpha)\rho_i(w) - 1\right] \qquad (7.2.4)\end{aligned}$$

where the first equality used that the minimization problem is convex and the solutions to $-\sigma_i(w)w_i^{-1} + v^\alpha = 0$ (i.e. where the gradient is 0) is a minimizer, and the second equality used $\rho_i(w) = \sigma_i(w)/w_i^{1+\alpha}$. Applying $\rho_i(w) \le 1+\alpha$, $1+x \le \exp x$, and $\exp x \le 1 + x + x^2$ for $x \le 1$ yields

$$\rho_i(w)^{\frac{1}{\alpha}} = (1-(1-\rho_i(w)))^{\frac{1}{\alpha}} \le \exp\left(\tfrac{1}{\alpha}(\rho_i(w)-1)\right) \le 1 + \frac{1}{\alpha}(\rho_i(w)-1) + \frac{1}{\alpha^2}(\rho_i(w)-1)^2. \qquad (7.2.5)$$

Combining (7.2.5) with (7.2.4) yields that

$$\begin{aligned}c_i &\ge \frac{w_i^{1+\alpha}}{1+\alpha}\left[-\alpha\rho_i(w)\left[1 + \left(\frac{\rho_i(w)-1}{\alpha}\right) + \left(\frac{\rho_i(w)-1}{\alpha}\right)^2\right] + (1+\alpha)\rho_i(w) - 1\right]\\ &= \frac{w_i^{1+\alpha}}{1+\alpha}\left[-1 + 2\rho_i(w) - \rho_i(w)^2 - \frac{\rho_i(w)}{\alpha}\cdot(\rho_i(w)-1)^2\right] = -\frac{w_i^{1+\alpha}}{1+\alpha}\left(1 + \frac{\rho_i(w)}{\alpha}\right)\cdot(\rho_i(w)-1)^2\end{aligned}$$

The claim then follows from the fact that for $\rho_i(w) \le 1+\alpha$, we have $\frac{(1+\rho_i(w)\alpha^{-1})(1+\rho_i(w))}{1+\alpha} \le \frac{1}{1+\alpha} + \frac{1}{\alpha} + 1 + 1 + \frac{1}{\alpha} \le 5\max\{1,\alpha^{-1}\}$. □

Lemma 7.2.6 (Function Decrease in **Descent**$(\cdot)$)**.** *Let $w, \eta \in \mathbb{R}^m_{>0}$ with $\eta_i \in [0, \frac{1}{3\overline{\alpha}}]$ for all $i \in [m]$. Further, let $w^+ =$ **Descent**$(w, [m], \eta)$, where **Descent** is defined in Equation (7.2.2). Then, $w^+ \in \mathbb{R}^m_{>0}$ with*

the following decrease in function objective.

$$\mathcal{F}(w^+) \le \mathcal{F}(w) - \sum_{i\in[m]} \frac{\eta_i}{2} \cdot w_i^{1+\alpha} \cdot \frac{(\rho_i(w)-1)^2}{\rho_i(w)+1}.$$

The proof of this lemma resembles that of quasi-Newton method: first, we write a second-order Taylor approximation of $\mathcal{F}(w^+)$ around w and apply Fact 7.1.6 to Lemma 7.2.3 to obtain the upper bound on Hessian: $\nabla^2\mathcal{F}(\widetilde{w}) = \widetilde{\mathbf{W}}^{-1}\mathbf{P}(\widetilde{w})^{(2)}\widetilde{\mathbf{W}}^{-1} + \alpha\widetilde{\mathbf{W}}^{\alpha-1} \preceq \widetilde{\mathbf{W}}^{-1}\Sigma(\widetilde{w})\widetilde{\mathbf{W}}^{-1} + \alpha\widetilde{\mathbf{W}}^{\alpha-1}$. We further use the expression for $\nabla\mathcal{F}(w)$ in this second-order approximation and simplify to obtain the claim, as detailed in Section F.1.

Iteration Complexity of Descent(·)

Proof of Lemma 7.2.2. Since Algorithm 7.2.1 calls **Descent**(·) after running **Round**(·), the requirement $\rho_{\max}(w) \le 1+\alpha$ in Lemma 7.2.5 is met. Therefore, we may combine Lemma 7.2.5 alongwith Lemma 7.2.6 and our choice of $\eta_i = \frac{1}{3\bar{\alpha}}$ in Algorithm 7.2.1 to get a geometric decrease in function error as follows.

$$\begin{aligned}\mathcal{F}(w^+) - \mathcal{F}(\overline{w}) &\le \mathcal{F}(w) - \mathcal{F}(\overline{w}) - \frac{1}{6\max(\alpha,1)}\sum_{i=1}^{m} w_i^{1+\alpha}\frac{(\rho_i(w)-1)^2}{\rho_i(w)+1} \\ &\le \left(1 - \frac{1}{30\max(1,\alpha)\cdot\max(1,\alpha^{-1})}\right)(\mathcal{F}(w) - \mathcal{F}(\overline{w})). \qquad (7.2.6)\end{aligned}$$

We apply this inequality recursively over all iterations of Algorithm 7.2.1, while also using the assumption that **Round**(·) does not increase the objective value. Setting the final value of (7.2.6) to $\widetilde{\varepsilon}$, bounding the initial error as $\mathcal{F}(w) - \mathcal{F}(\overline{w}) \le n\log(m/n) \le m^2$ by Lemma 7.2.4, observing $\max(1,\alpha)\cdot\max(1,\alpha^{-1}) = \max(\alpha,\alpha^{-1})$, and taking logarithms on both sides of the inequality gives the claimed iteration complexity of **Descent**(·). □

7.3 Analysis of Round(·): The Parallel Algorithm

The main export of this section is the proof of our main theorem about the parallel algorithm (Theorem 7.1.2). This proof combines the iteration count of **Descent**(·) from the preceding section with the analysis of Algorithm 7.2.2 (invoked by **Round**(·) in the parallel setting), shown next. In Lemma 7.3.1, we show that **RoundParallel**(·) decreases the function objective, thereby justifying the key assumption in Lemma 7.2.2. Lemma 7.3.1 also shows an upper bound on the new value of ρ after one `while` loop of **RoundParallel**(·), and by combining this with the maximum value of ρ for termination in Algorithm 7.2.2, we get the iteration complexity of **RoundParallel**(·) in Corollary 7.3.2.

Lemma 7.3.1 (Outcome of **RoundParallel**(·))**.** *Let $w^+ \in \mathbb{R}^m_{>0}$ be the state of $w \in \mathbb{R}^m_{>0}$ at the end of one* `while` *loop of* ***RoundParallel***(·) *(Algorithm 7.2.2). Then, $\mathcal{F}(w^+) \le \mathcal{F}(w)$ and $\rho_{\max}(w^+) \le (1+\frac{\alpha}{3\bar{\alpha}(2+\alpha)})^{-\alpha}\rho_{\max}(w)$.*

Proof. Each iteration of the `while` loop in **RoundParallel**(·) performs **Descent**$(w, \mathrm{C}, \frac{1}{3\bar{\alpha}}\mathbf{1})$ over the set of coordinates $\mathrm{C} = \{i : \rho_i(w) > 1+\alpha\}$. Lemma 7.2.6 then immediately proves $\mathcal{F}(w^+) \le \mathcal{F}(w)$, which is our first claim.

To prove the second claim, note that in Algorithm 7.2.2, for every $i \in C$

$$w_i^+ = w_i + \frac{w_i}{3\bar{\alpha}} \cdot \left[\frac{\rho_i(w) - 1}{\rho_i(w) + 1}\right] \geq w_i + \frac{w_i}{3\bar{\alpha}} \cdot \left[\frac{\alpha}{1 + 1 + \alpha}\right] = w_i \cdot \left(1 + \frac{\alpha}{3\bar{\alpha}(2+\alpha)}\right),$$

where the second step is by the monotonicity of $x \to \frac{x-1}{x+1}$ for $x \geq 1$. Combining it with $w_i^+ = w_i$ for all $i \notin C$ implies that $w^+ \geq w$. Therefore, for all $i \in C$, we have

$$\rho(w^+)_i = [w_i^+]^{-\alpha}[\mathbf{A}(\mathbf{A}^\top \mathbf{W}^+ \mathbf{A})^{-1}\mathbf{A}^\top]_{ii} \leq \left[1 + \frac{\alpha}{3\bar{\alpha}(2+\alpha)}\right]^{-\alpha} \cdot w_i^{-\alpha}[\mathbf{A}(\mathbf{A}^\top \mathbf{W}\mathbf{A})^{-1}\mathbf{A}^\top]_{ii}. \tag{7.3.1}$$

□

Corollary 7.3.2. *Let w be the input to **RoundParallel**(·). Then, the number of iterations of the* `while` *loop of **RoundParallel**(·) is at most $O\left((1 + \alpha^{-2}) \log\left(\frac{\rho_{\max}(w)}{1+\alpha}\right)\right)$.*

Proof. Let $w^{(i)}$ be the value of w at the start of the i'th execution of the `while` loop of **RoundParallel**(·). Repeated application of Lemma 7.3.1 over k executions of the `while` loop gives $\rho_{\max}(w^{(k)}) \leq \rho_{\max}(w)\left(1 + \frac{\alpha}{3\bar{\alpha}(2+\alpha)}\right)^{-\alpha k}$. We set $\rho_{\max}(w)\left(1 + \frac{\alpha}{3\bar{\alpha}(2+\alpha)}\right)^{-\alpha k} \leq 1 + \alpha$ in accordance with the termination condition of **RoundParallel**(·). Next, applying $1 + x \leq \exp(x)$, and taking logarithms on both sides yields the claimed limit on the number of iterations, k. □

Lemma 7.3.3. *Over the entire run of Algorithm 7.2.1, the* `while` *loop of **RoundParallel**(·) runs for at most $O\left(\mathcal{T}_{total} \cdot \alpha^{-2} \cdot \log\left(\frac{m}{n(1+\alpha)}\right)\right)$ iterations if $\alpha \in (0, 1]$ and $O\left(\mathcal{T}_{total} \cdot \alpha \cdot \log\left(\frac{m}{n(1+\alpha)}\right)\right)$ iterations if $\alpha \geq 1$.*

Proof. Note that $\rho_{\max}(\frac{n}{m}) \leq (\frac{m}{n})^{1+\alpha}$; consequently, in the first iteration of Algorithm 7.2.1, there are at most $O((\alpha + \alpha^{-2}) \log(m/(n(1+\alpha))))$ iterations of the `while` loop of **RoundParallel**(·) by Corollary 7.3.2. Note that between each call to **RoundParallel**(·), for all $i \in [m]$,

$$w_i^+ = w_i + \frac{w_i}{3\bar{\alpha}} \cdot \left[\frac{\rho_i(w) - 1}{\rho_i(w) + 1}\right] \geq w_i + \frac{w_i}{3\bar{\alpha}} \cdot \left[\frac{-1}{1 + 1 + \alpha}\right] = w_i \cdot \left(1 - \frac{1}{(3\bar{\alpha})(2+\alpha)}\right),$$

where the first inequality is by using the fact that the output w of **RoundParallel**(·) satisfies $\rho_{\max}(w) \leq 1 + \alpha$. Therefore, applying the same logic as in (7.3.1), we get that between two calls to **RoundParallel**(·), the value of $\rho_i(w)$ increases by at most $\left(1 - \frac{1}{(3\bar{\alpha})(2+\alpha)}\right)^{-(1+\alpha)} = O(1)$ for all $i \in [m]$. Combining this with Corollary 7.3.2 and the total initial iteration count and observing that $\mathcal{T}_{\text{total}}$ is the total number of calls to **RoundParallel**(·) finishes the proof. □

7.3.1 Proof of Main Theorem (Parallel)

Proof. (Proof of Theorem 7.1.2) First, we show correctness. Note that, as a corollary of Lemma 7.2.2, $\mathcal{F}(w^{(\mathcal{T}_{\text{total}})}) \leq \mathcal{F}(\overline{w}) + \widetilde{\varepsilon}$. By the properties of **Round** as shown in Lemma 7.3.1, we also have that $\mathcal{F}(w_R) \leq \mathcal{F}(\overline{w}) + \widetilde{\varepsilon}$ and $\rho_{\max}(w_R) \leq 1 + \alpha$ for $w_R = \textbf{Round}(w^{(\mathcal{T}_{\text{total}})}, \mathbf{A}, \alpha)$. Therefore, Lemma 7.2.1 is applicable, and by the choice of $\widetilde{\varepsilon} = \frac{\alpha^4\epsilon^4}{(2m(\sqrt{n}+\alpha)(\alpha+\alpha^{-1}))^4}$, we conclude that $\widehat{w} \in \mathbb{R}^m$ defined as $\widehat{w}_i = (a_i^\top(\mathbf{A}^\top \mathbf{W}_R \mathbf{A})^{-1} a_i)^{1/\alpha}$ satisfies $\widehat{w}_i \approx_\epsilon \overline{w}_i$ for all $i \in [m]$. Combining the iteration counts of **Descent**(·) from Lemma 7.2.2 and of **RoundParallel**(·) from Lemma 7.3.3 yields the total iteration count as $O(\alpha^{-3} \log(m/(\epsilon\alpha)))$ if $\alpha \leq 1$ and $O(\alpha^2 \log(m/\epsilon))$ if $\alpha > 1$. As stated in Section 7.1.4, $\alpha = \frac{2}{p-2}$,

and so translating these rates in terms of p gives $O(p^3 \log(mp/\epsilon))$ for $p \geq 4$ and $O(p^{-2} \log(mp/\epsilon))$ for $p \in (2, 4)$, thereby proving the stated claim. The cost per iteration is $O(mn^2)$[6] for multiplying two $m \times n$ matrices. □

7.4 Analysis of Round(·): Sequential Algorithm

We now analyze Algorithm 7.2.3. Note that these proofs work for all $\alpha > 0$.

Lemma 7.4.1 (Coordinate Step Progress). *Given $w \in \mathbb{R}^m_{>0}$, a coordinate $i \in [m]$, and $\delta_i \in \mathbb{R}$, we have*

$$\mathcal{F}(w + \delta_i w_i e_i) = \mathcal{F}(w) - \log(1 + \delta_i \sigma_i(w)) + \frac{w_i^{1+\alpha}}{1+\alpha}((1+\delta_i)^{1+\alpha} - 1).$$

Proof. By definition of $\mathcal{F}$, we have

$$\mathcal{F}(w + \delta_i w_i e_i) = -\log\det\left(\mathbf{A}^\top \mathbf{W}\mathbf{A} + \delta_i w_i a_i a_i^\top\right) + \frac{1}{1+\alpha}\sum_{j \neq i} w_j^{1+\alpha} + \frac{w_i^{1+\alpha}}{1+\alpha}(1+\delta_i)^{1+\alpha}.$$

Recall the matrix determinant lemma: $\det(\mathbf{A} + uv^\top) = (1 + v^\top \mathbf{A}^{-1} u)\det(\mathbf{A})$. Applying it to $\det(\mathbf{A}^\top \mathbf{diag}(()w + \delta_i w_i e_i)\mathbf{A})$ in the preceding expression for $\mathcal{F}(w + \delta_i w_i e_i)$ proves the lemma.

□

Lemma 7.4.2 (Coordinate Step Outcome). *Given $w \in \mathbb{R}^m_{>0}$ and $\mathrm{C} = \{i : \rho_i(w) \geq 1\}$, let $w^+ = w + \delta_i w_i e_i$ for any $i \in \mathrm{C}$, where $\delta_i = \arg\min_\delta \left[-\log(1 + \delta\sigma_i(w)) + \frac{1}{1+\alpha} w_i^{1+\alpha}((1+\delta)^{1+\alpha} - 1)\right]$. Then, we have $\mathcal{F}(w^+) \leq \mathcal{F}(w)$ and $\rho_i(w^+) \leq 1$.*

Proof. We note that $\min_\delta \left[-\log(1 + \delta\sigma_i(w)) + \frac{1}{1+\alpha} w_i^{1+\alpha}((1+\delta)^{1+\alpha} - 1)\right] \leq 0$. Then, Lemma 7.4.1 implies the first claim. Since the update rule optimizes over $\mathcal{F}$ coordinate-wise, at each step the optimality condition given by $\rho_i(w^+) = 1$ is met for each $i \in \mathrm{C}$. The second claim is then proved by noting that for $j \neq i$, $w_j^+ = w_j$ and by the Sherman-Morrison-Woodbury identity, $\rho_j(w^+) \leq \rho_j(w)$:

$$a_j^\top (\mathbf{A}^\top \mathbf{W}^+ \mathbf{A})^{-1} a_j = a_j^\top (\mathbf{A}^\top \mathbf{W}\mathbf{A})^{-1} a_j - \delta_i w_i \frac{(a_j^\top (\mathbf{A}^\top \mathbf{W}\mathbf{A})^{-1} a_j)^2}{1 + \delta_i w_i a_i^\top (\mathbf{A}^\top \mathbf{W}\mathbf{A})^{-1} a_i} \leq a_j^\top (\mathbf{A}^\top \mathbf{W}\mathbf{A})^{-1} a_j.$$

□

Lemma 7.4.3 (Number of Coordinate Steps). *For any $0 \leq \widetilde{\varepsilon} \leq 1$, over all $\mathcal{T}_{total}$ iterations of Algorithm 7.2.1, there are at most $O(m \max(\alpha, \alpha^{-1}) \log(m/\widetilde{\varepsilon}))$ coordinate steps (see Algorithm 7.2.3).*

Proof. There are at most m coordinate steps in each call to Algorithm 7.2.3. Combining this with the value of $\mathcal{T}_{\text{total}}$ in Algorithm 7.2.1 gives the count of $O(m\alpha^{-1} \log(m/\widetilde{\varepsilon}))$ coordinate steps. □

[6] This can be improved to $O(mn^{\omega-1})$ using fast matrix multiplication.

7.4.1 Proof of Main Theorem (Sequential)

We now combine the preceding results to prove the main theorem about the sequential algorithm (Algorithm 7.2.1 with Algorithm 7.2.3).

Proof of Theorem 7.1.3. The proof of correctness is the same as that for Theorem 7.1.2 since the parallel and sequential algorithms share the same meta-algorithm. Computing leverage scores in the sequential version (Algorithm 7.2.1 with Algorithm 7.2.3) takes $O(m \max(\alpha, \alpha^{-1}) \log(m/(\alpha\epsilon)))$ coordinate steps. The costliest component of a coordinate step is computing $a_i^\top (\mathbf{A}^\top (\mathbf{W} + \delta_i w_i e_i e_i^\top)\mathbf{A})^{-1} a_i$. By the Sherman-Morrison-Woodbury formula, computing this costs $O(n^2)$ for each coordinate. Since the initial cost to compute $(\mathbf{A}^\top \mathbf{W}\mathbf{A})^{-1}$ is $O(mn^2)$, the total run time is $O(\max(\alpha, \alpha^{-1}) mn^2 \log(m/\epsilon))$. In terms of p, this gives $O(pmn^2 \log(mp/\epsilon))$ for $p \geq 4$ and $O(p^{-1}mn^2 \log(mp/\epsilon))$ for $p \in (2, 4)$. □

7.5 A "One-Step" Parallel Algorithm

We conclude this chapter with an alternative algorithm (Algorithm 7.5.1) in which each iteration avoids any rounding and performs only **Descent**(·).

Algorithm 7.5.1 One-Step Algorithm

Input: Matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, parameter $p > 2$, accuracy ϵ
Output: Vector $\widehat{w} \in \mathbb{R}^m_{>0}$ that satisfies Equation (7.1.5)

1: For all $i \in [m]$, initialize $w_i^{(0)} = 1$. Set $\alpha = \frac{2}{p-2}$. Set $\widetilde{\varepsilon} = \frac{\alpha^4 \epsilon^4}{(2m(\sqrt{n}+\alpha)(\alpha+\alpha^{-1}))^4}$.
2: Set $\beta = \frac{1}{1000} \min(\alpha^2, 1)$ and $\mathcal{T}_{\text{total}} = \begin{cases} O(\alpha^{-3} \log(mp/\widetilde{\varepsilon})) & \text{if } \alpha \in (0, 1] \\ O(\alpha^2 \log(mp/\widetilde{\varepsilon})) & \alpha > 1 \end{cases}$
3: **for** $k = 0, 1, 2, 3, \ldots, \mathcal{T}_{\text{total}} - 1$ **do**
4: Let $\eta^{(k)} \in \mathbb{R}^m$ where for all $i \in [m]$ we let $\eta_i^{(k)} = \begin{cases} \frac{1}{3\bar{\alpha}} & \text{if } \rho_i(w^{(k)}) \geq 1 \\ \frac{1}{3\bar{\alpha}}\beta & \text{if } \rho_i(w^{(k)}) < 1 \end{cases}$
5: $w^{(k+1)} \leftarrow$ **Descent**$(w^{(k)}, [m], \eta^{(k)})$ ▷ See Equation (7.2.2) and Lemma 7.2.2
6: **end for**
7: Return $\widehat{w} \in \mathbb{R}^m_{>0}$, where $\widehat{w}_i = (a_i^\top (\mathbf{A}^\top \mathbf{W}^{(\mathcal{T}_{\text{total}})} \mathbf{A})^{-1} a_i)^{1/\alpha}$. ▷ See Section F.2

Theorem 7.5.1 (Main Theorem (One-Step Parallel Algorithm)). *Given a full rank matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $p \geq 4$, we can compute ϵ-approximate Lewis weights Equation (7.1.5) in $O(p^3 \log(mp/\epsilon)$ iterations. Each iteration computes the leverage score of one row of $\mathbf{DA}$ for some diagonal matrix $\mathbf{D}$. The total runtime is $O(p^3 mn^2 \log(mp/\epsilon))$.*

We first spell out the key idea of the proof of Theorem 7.5.1 in Lemma 7.5.2 next, which is that Inequality 7.2.3 is maintained in every iteration through the use of varying step sizes, *without explicitly invoking rounding procedures*. Since Inequality 7.2.3 always holds, we may use Lemma 7.2.5 in bounding the iteration complexity.

Lemma 7.5.2 (Rounding Condition Invariance). *For any iteration $k \in [\mathcal{T}_{\text{total}} - 2]$ in Algorithm 7.5.1, if $\rho_{\max}(w^{(k)}) \leq 1 + \alpha$, then $\rho_{\max}(w^{(k+1)}) \leq 1 + \alpha$.*

Proof. By the definition of **Descent**(·) in Equation (7.2.2) and choice of $\eta_i^{(k)}$ in Algorithm 7.5.1, we

have,

$$w_i^{(k+1)} = w_i^{(k)} \cdot \left[1 + \eta_i^{(k)} \left(\frac{\rho_i(w^{(k)}) - 1}{\rho_i(w^{(k)}) + 1}\right)\right] \tag{7.5.1}$$

$$\geq w_i^{(k)}(1 - \eta_i^{(k)}) \geq w_i^{(k)}\left(1 - \frac{\beta}{3\bar{\alpha}}\right). \tag{7.5.2}$$

Applying this inequality to the definition of $\rho(w)$ in (7.2.1), for all $i \in [m]$, we have

$$\rho_i(w^{(k+1)}) = \left[\frac{w_i^{(k+1)}}{w_i^{(k)}}\right]^{-\alpha} \frac{1}{[w_i^{(k)}]^{\alpha}} a_i^\top (\mathbf{A}^\top \mathbf{W}^{(k+1)} \mathbf{A})^{-1} a_i \leq \left(1 - \frac{\beta}{3\bar{\alpha}}\right)^{-1} \left[\frac{w_i^{(k+1)}}{w_i^{(k)}}\right]^{-\alpha} \rho_i(w^{(k)}). \tag{7.5.3}$$

Plugging Inequality 7.5.2 into Inequality 7.5.3 when $\rho_i(w^{(k)}) \leq 1$ and using the upper bound on β yields that

$$\rho_i(w^{(k+1)}) \leq \left(1 - \frac{\beta}{3\bar{\alpha}}\right)^{-(1+\alpha)} \leq 1 + \alpha\,.$$

If $\rho_i(w^{(k)}) \geq 1$, then Inequality 7.5.3, the equality in Inequality 7.5.2, the bound on β, and $\rho_i(w^{(k)}) \leq 1+\alpha$ imply that

$$\rho_i(w^{(k+1)}) \leq \left(1 - \frac{\beta}{3\bar{\alpha}}\right)^{-1} \left[1 + \frac{1}{3\bar{\alpha}} \left(\frac{\rho_i(w^{(k)}) - 1}{\rho_i(w^{(k)}) + 1}\right)\right]^{-\alpha} \rho_i(w^{(k)}) \leq 1 + \alpha.$$

□

Proof of Theorem 7.5.1. By our choice of $w_i^{(0)} = 1$ for all $i \in [m]$, we have that $\rho_i(w^{(0)}) = \sigma_i(w^{(0)}) \leq 1$ by Fact 7.1.6. Then applying Lemma 7.5.2 yields by induction that $\rho_{\max}(w^{(k)}) \leq 1 + \alpha$ at every iteration k. We may now therefore upper bound the objective sub-optimality from Lemma 7.2.5; as before, combining this with the lower bound on progress from Lemma 7.2.6 (noticing that $\eta_i \geq \frac{\beta}{3\bar{\alpha}}$) yields

$$\mathcal{F}(w^+) - \mathcal{F}(\overline{w}) \leq \mathcal{F}(w) - \mathcal{F}(\overline{w}) - \frac{\beta}{6\bar{\alpha}} \sum_{i=1}^m w_i^{1+\alpha} \frac{(\rho_i(w) - 1)^2}{\rho_i(w) + 1}$$

$$\leq \left(1 - \frac{\beta}{30 \max(1, \alpha) \max(1, \alpha^{-1})}\right)(\mathcal{F}(w) - \mathcal{F}(\overline{w})). \tag{7.5.4}$$

Thus, **Descent**($\cdot$) decreases $\mathcal{F}$. Using $\mathcal{F}(w) - \mathcal{F}(\overline{w}) \leq n \log(m/n) \leq m^2$ from Lemma 7.2.4 and setting Inequality 7.5.4 to $\widetilde{\varepsilon}$ gives an iteration complexity of $O(\beta^{-1}\alpha^{-1} \log(m/\widetilde{\varepsilon})) = O(\alpha^{-3} \log(m/\widetilde{\varepsilon}))$ if $\alpha \in (0, 1]$ and $O(\alpha\beta^{-1} \log(m/\widetilde{\varepsilon})) = O(\alpha \log(m/\widetilde{\varepsilon}))$ otherwise. As in the proofs of Theorem 7.1.2 and Theorem 7.1.3, we then invoke Lemma 7.2.1 to construct an ϵ-approximation to the Lewis weights. □

Chapter 8

Online Bidding Algorithms for Return-on-Spend Constrained Advertisers

Online advertising has recently grown into a highly competitive and complex multi-billion-dollar industry, with advertisers bidding for ad slots at large scales and high frequencies. This has resulted in a growing need for efficient "auto-bidding" algorithms that determine the bids for incoming queries to maximize advertisers' targets subject to their specified constraints. This work explores efficient online algorithms for a single value-maximizing advertiser under an increasingly popular constraint: Return-on-Spend (RoS). We quantify efficiency in terms of regret relative to the optimal algorithm, which knows all queries a priori. We contribute a simple online algorithm that achieves near-optimal regret in expectation while always respecting the specified RoS constraint when the input sequence of queries are i.i.d. samples from some distribution. We also integrate our results with the previous work of Balseiro, Lu, and Mirrokni [BLM20] to achieve near-optimal regret while respecting both RoS and fixed budget constraints. Our algorithm follows the primal-dual framework and uses online mirror descent (OMD) for the dual updates. However, we need to use a non-canonical setup of OMD, and therefore the classic low-regret guarantee of OMD, which is for the adversarial setting in online learning, no longer holds. Nonetheless, in our case and more generally where low-regret dynamics are applied in algorithm design, the gradients encountered by OMD can be far from adversarial but influenced by our algorithmic choices. We exploit this key insight to show our OMD setup achieves low regret in the realm of our algorithm.

8.1 Introduction

With the explosive growth of online advertising into a billion-dollar industry, auto-bidding — the practice of using optimization algorithms to generate bids for search queries on behalf of advertisers — has emerged as a predominant tool in online advertising [ABM19, BG19, BCH+21, GJLM21, DMMZ21, BDM+21a, BDM+21b, GYC+22]. Unlike manual cost-per-click bidding, which requires advertisers to manually submit bids for new search queries, auto-bidding needs only high-level goals and constraints from advertisers. The advertising platform then deploys its auto-bidding agent, which, using its underlying optimization algorithms, transforms these inputs into fine-grained bids. Hence, designing an efficient bidding algorithm for advertisers to meet their targets while respecting their specified constraints constitutes a central problem in online advertising.

We study return-on-spend (RoS) constrained auto-bidding for a single bidder. The RoS constraint requires the ratio of the bidder's total value to its total payment to be at least some specified target ratio. In practice, this captures popular constraints like target cost-per-acquisition (tCPA) and target return-on-ad-spend (tROAS).[1] Our algorithm, though tailored to the RoS constraint, easily adapts

[1] See Google ads support page and Meta business help center for examples.

to an additional budget constraint on the total payment.

Our setting is stochastic: In each round, a search query and auction are generated i.i.d. from an unknown distribution, after which the bidder observes the value of the query and submits a bid. The auction mechanism specifies if the bidder wins and the price. We aim to design an online bidding algorithm for the bidder to maximize its total value respecting the RoS and budget constraints.

8.1.1 Our Main Result

We evaluate our algorithm's performance via *regret* (Equation (8.2.8)), its expected loss over instances in reference to the offline optimal solution. Our main result follows.

Theorem 8.1.1 (Informal; see Theorem 8.5.2)**.** *We give an algorithm (Algorithm 8.5.2) for value maximization under RoS and budget constraints, which, for a T-length input i.i.d. sequence of search queries, provably attains $O(\sqrt{T}\log T)$ regret while respecting both the budget and RoS constraints.*

Our notion of regret differs from that in online learning, which compares the algorithm to the optimal *fixed* strategy over all instances. Moreover, our guarantee on the constraints is for *each instance* (i.e. worst-case) rather than in expectation. Our result holds for i.i.d. input sequences and under an additional mild technical assumption on the input distribution (Assumption 8.4.1). To the best of our knowledge, ours is the first algorithm attaining near-optimal regret satisfying *both* budget and RoS constraints *in any outcome*. In doing so, we improve upon the work of [BLM20], which obtains similar guarantees under only budget constraints.

8.1.2 Our Main Techniques

Underlying all our algorithms is a primal-dual framework similar to that used by [BLM20]. Such a framework lets our algorithms adapt to the changing values and prices of input queries while respecting the advertisers' specified constraints and goals over the entire time horizon. Specifically, the dual variable (which tracks the constraint violation) is updated via online mirror descent (OMD) using the generalized cross-entropy function, which imposes a large (exponential) penalty on total constraint violation.

An immediate technical challenge in attempting to use generalized cross-entropy as the mirror map is its lack of strong convexity on the non-negative orthant. While this mirror map appears in [BLM20] as well, here, the fixed budget constraint bounds the corresponding dual variable by a constant, which in turn implies the desired strong convexity on the space over which their algorithm operates. In our case (with the RoS constraint), *there exist example inputs on which no such bound exists*, thus necessitating a novel analysis that circumvents the lack of strong convexity.

Our novel insight is as follows. While the low-regret guarantee of OMD with a strongly convex mirror map holds even when the gradients seen by OMD are adversarial, OMD's dual updates in *our* algorithm produces gradients that are connected to our primal algorithmic decisions (i.e., bids); this connection suffices to give our algorithm the low-regret guarantee even *without a strongly convex mirror map*. In a white-box adaptation of the OMD analysis tailored to our algorithm, we use this connection along with properties of the generalized cross-entropy function from the (offline) positive linear programming literature [AZO14].

Our Algorithmic Outline. To build up to our result, we first obtain a $O(\sqrt{T})$-regret algorithm (Algorithm 8.3.1) for value maximization under an *approximate* RoS constraint, which allows up to $O(\sqrt{T}\log T)$ constraint violation; Assumption 8.4.1 is not needed for this result. When

Assumption 8.4.1 holds, a simple modification to Algorithm 8.3.1 yields Algorithm 8.4.1, with a $O(\sqrt{T}\log T)$ regret guarantee under a *strict* RoS constraint (Theorem 8.4.4). We then combine Algorithm 8.3.1 with ideas from [BLM20] to design Algorithm 8.5.1, which attains $O(\sqrt{T})$-regret under *two* constraints: strict budget and approximate RoS. Finally, unifying ideas from Algorithm 8.4.1 and Algorithm 8.5.1 yields our main result.

To turn an algorithm that only *approximately* satisfies a constraint into one that *strictly* satisfies it, we employ a simple strategy: We first submit a sequence of bids that lets us accumulate a slack on the RoS constraint (at the cost of bidding sub-optimally), followed by the existing algorithm, which suffers some bounded constraint violation (see Section 8.4). The first phase compensates for the constraint violation from the subsequent iterations. This allows trading off the violation on the RoS constraint for the objective value.

8.1.3 Related Work

Our problem falls under the broader umbrella of online optimization under stochastic time-varying constraints, and it has seen a long line of research by various research communities, e.g. [MTY09, MJY12, MYJ13, AD14, YNW17, BKS18, ISSS19, YN20, BLM20, CCM+22, GYC+22].

Most of these works study the *budget constraint*, e.g., [DJSW19] prove the optimal $O(\sqrt{T})$-regret under linear objective and constraints, [AD14] generalizes this beyond the linear objective, and [BLM20] generalizes it to nonlinear budget constraints. The RoS constraint we study is fundamentally different from these packing-type constraints studied in these works as well as in [BKS18, ISSS19]. There also exist papers that study a variant of our problem with a constraint class more general than ours (e.g. [AD14, CCM+22]); however, they do not provide guarantees as strong as ours, as we elaborate next.

For example, [CCM+22] gives a primal-dual framework using regret minimization, which, when adapted to our bidding problem under the RoS constraint, achieves $\tilde{O}(T^{3/4})$ regret with $\tilde{O}(T^{3/4})$ constraint violation (both with high probability). Both bounds are polynomially weaker than our guarantees (that further hold deterministically). Their bounds improve to $\tilde{O}(T^{1/2})$ under a 'strictly feasible' assumption, which is essentially our Assumption 8.4.1; in contrast, under that assumption, we guarantee *strict* constraint satisfaction (Theorem 8.4.4). Moreover, their algorithm uses techniques from the multi-armed bandits literature, thus requiring values and bids to be of finite size n_v, n_b respectively, and their regret bound scales with $n_v\sqrt{n_b}$. Our algorithm works directly with continuous values and bids.

Another example is [AD14], which considers general online optimization with convex constraints. This work uses black-box low-regret methods that rely on a globally strongly convex regularizer over the dual space, and a sub-linear regret bound is attainable only when the dual space is well-bounded (e.g. a scaled simplex) or the dual variable can be projected onto such a space without incurring too much additional regret. This canonical approach turns out to be difficult for the RoS constraint, which can incur poor problem-specific parameters in generic guarantees. Hence, this technique cannot give sub-linear regret for the RoS constraint. To circumvent this issue, we rely on problem-specific structure rather than globally strongly convex regularization.

Another closely related work is [GJLM21] which investigates the same problem we do but with the RoS and budget constraints holding *only in expectation over the distribution*; in contrast, our constraint guarantees hold for any realization of samples. Though [GJLM21] give an example where bidding based on the optimal offline (fixed) dual variable cannot achieve sub-linear regret, it does not contradict our results since our algorithm is *adaptive* rather than bidding based on fixed offline

optimal dual variables. Our algorithm updates our dual variables based on the previous outcomes, which takes advantage of stochastic information to balance the objective and constraint violation.

The problem of *learning to bid in repeated auctions* has been widely studied in both academia and industry, e.g. [BCI+07, WPR16, FPS18, HZF+20, BFG21, NS21, NCKP22]. These papers mainly abstract the problem of learning to bid as contextual bandits but do not incorporate constraints into them. Beyond this, there has been some work on bidding under budget constraints, e.g., [BG19, AWL+22], however, these papers focus on utility-maximizing agents with at most one constraint. [CCBKS22] also considers multiple different constraints in online bidding algorithms, however, they directly add a regularizer of one non-packing constraint in the objective and apply the standard dual mirror descent approach to design the algorithms. Their regret bound is measured against this relaxed objective, whereas, our regret guarantee is relative to the adaptive optimal benchmark.

Finally, loosely related work includes the AdWords problem [MSVV07, DH09], which manages budgets for multiple bidders to maximize the seller's revenue; in contrast, we focus on online bidding algorithms for a single bidder with RoS and budget constraints.

As a final remark on the novelty of our work, we note that algorithms in this line of research are never surprising: a gradient-based online optimization blackbox (e.g. OMD) to update dual variables, with primal variables choosing the best responses. The progress lies largely in analysis techniques showing if and why such simple methods achieve optimal guarantees in broader settings. For the budget constraint, [DJSW19] prove the optimal $O(\sqrt{T})$-regret under linear objective and constraints, [AD14] generalizes this beyond the linear objective, and [BLM20] generalizes it to nonlinear budget constraints. Our result is no exception to this trend in generalizing to RoS constraints, which are fundamentally different from packing-type budget constraints.

Our result is surprising given prior results for the RoS constraint which either use more complicated algorithms for suboptimal guarantees [CCM+22] or switch to an approach of learning the distribution, which yields only a bound on the expectation of the constraint error [GJLM21]. It is a bonus that our algorithm continues to remain simple and practical while achieving both optimal regret and ex-post constraint error bound.

8.2 Preliminaries

We consider an online bidding model for a single learner (auto-bidder): At each time step t, nature stochastically generates for the learner an ad query associated with a value $v_t \in [0,1]$ and an auction mechanism (x_t, p_t), where $x_t : \mathbb{R}_{\geq 0} \mapsto [0,1]$ is an allocation function and $p_t : \mathbb{R}_{\geq 0} \mapsto [0,1]$ the expected payment rule.[2] We assume the following stochastic model: (v_t, x_t, p_t) are drawn independently and identically (i.i.d.) from an unknown distribution. At each time step t, the value v_t is known to the learner before making a bid, and the learner decides its bid b_t given v_t and historical information. At the end of time step t, the learner observes the realized outcome of the auction mechanism, i.e., $x_t(b_t)$ and $p_t(b_t)$.

We focus only on truthful auctions. This requires that the allocation function $x_t(b)$ be non-decreasing with the input bid b and the payment function p_t be characterized by [Mye81] as

$$p_t(b) = b \cdot x_t(b) - \int_{z=0}^{b} x_t(z)dz. \tag{8.2.1}$$

[2]Our algorithm's regret guarantee depends linearly on the scales of the valuation and payment, and the assumed bounds on these quantities are for theoretical simplicity.

For instance, the well-known second-price auction for a single item is truthful, and its payment function satisfies Equation (8.2.1). Note the payment must be zero when the allocation is zero, and the payment is also at most the bid. This work also assumes $v_t \cdot x_t(b_t)$ to be the realized value of the learner in each round.

We design online bidding algorithms to maximize the learner's total realized value subject to an RoS constraint. Formally, the optimization problem under RoS constraint we study is

$$\begin{array}{ll} \underset{b_t : t=1,\cdots,T}{\text{maximize}} & \sum_{t=1}^{T} v_t \cdot x_t(b_t) \\ \text{subject to} & \mathsf{RoS} \cdot \sum_{t=1}^{T} p_t(b_t) \le \sum_{t=1}^{T} v_t \cdot x_t(b_t), \end{array} \tag{8.2.2}$$

where $\mathsf{RoS} > 0$ is the target ratio of the RoS bidder. Throughout this chapter we assume without loss of generality[3] that $\mathsf{RoS} = 1$. As noted in Section 6.1, our results can be extended to handle different learner objectives, e.g., a hybrid version between utility maximizing and value maximizing $\sum_{t=1}^{T} v_t \cdot x_t(b_t) - \tau p_t(b_t)$ for some $\tau \in [0, 1]$.

To simplify notation, we denote the difference between value and price in iteration t as

$$g_t(b) := v_t \cdot x_t(b) - p_t(b), \tag{8.2.3}$$

and now the RoS constraint in Problem 8.2.2 may be stated as

$$\sum_{t=1}^{T} g_t(b_t) \ge 0. \tag{8.2.4}$$

Our algorithm also extends to the bidding problem subject to an additional budget constraint:

$$\begin{array}{ll} \underset{b_t : t=1,\cdots,T}{\text{maximize}} & \sum_{t=1}^{T} v_t \cdot x_t(b_t) \\ \text{subject to} & \sum_{t=1}^{T} p_t(b_t) \le \sum_{t=1}^{T} v_t \cdot x_t(b_t), \\ & \sum_{t=1}^{T} p_t(b_t) \le \rho T. \end{array} \tag{8.2.5}$$

where ρT is the budget and $\rho > 0$ (assumed a fixed constant) is the limit of the average expenditure over T rounds (ad queries).

To collect notation, we denote the sample (ad query and auction) at time t as a tuple $\gamma_t = (v_t, p_t, x_t)$ and assume $\gamma_t \sim \mathcal{P}$ for all $t \in [T]$. We denote the sequence of T samples by $\overrightarrow{\gamma} := \{\gamma_1, \gamma_2, \ldots, \gamma_T\} \sim \mathcal{P}^T$ and sequences of length $\ell \neq T$ by $\overrightarrow{\gamma}_\ell$ where needed.

Analysis setup. We use the notions of regret and constraint violation to measure the performance of our algorithms. To define the regret, we first define the reward of Alg for a sequence of requests $\overrightarrow{\gamma}$ over a time horizon T as

$$\mathsf{Reward}(\mathsf{Alg}, \overrightarrow{\gamma}) := \sum_{t=1}^{T} v_t \cdot x_t(b_t). \tag{8.2.6}$$

Next, we define the optimal value in the same setup as for Alg as

$$\mathsf{Reward}(\mathsf{Opt}, \overrightarrow{\gamma}) := \text{maximum}_{b_t \in \mathcal{B}} \sum_{t=1}^{T} v_t \cdot x_t(b_t), \tag{8.2.7}$$

[3] For any $\mathsf{RoS} \neq 1$, we can scale the values to be $v_t := \mathsf{RoS} \cdot v_t$.

where $\mathcal{B}$ is the exact set of constraints. These definitions lead to the definition of regret of Alg in this setup as

$$\mathsf{Regret}(\mathsf{Alg}, \mathcal{P}^T) := \mathbb{E}_{\overrightarrow{\gamma} \sim \mathcal{P}^T} \left[\mathsf{Reward}(\mathsf{Opt}, \overrightarrow{\gamma}) - \mathsf{Reward}(\mathsf{Alg}, \overrightarrow{\gamma}) \right]. \tag{8.2.8}$$

We remark that we define Reward for some specific input sequence, whereas Regret is defined with respect to a distribution.

Finally, we use online mirror descent as a technical component in our analysis. In this regard, we use $V_h(y, x) = h(y) - h(x) - h'(x) \cdot (y - x)$ to denote the Bregman divergence of y in reference to x, measuring with the distance-generating function ("mirror map") h. We review online mirror descent in the supplemental material.

8.3 Approximate RoS Constraint

As noted in Section 8.1.2, we first design and analyze an algorithm for Problem 8.2.2 allowing for *sub-linear violation* of the RoS constraint. The main export of this section is this algorithm (Algorithm 8.3.1) and its guarantee (Theorem 8.3.1).

To explain Algorithm 8.3.1, we first rewrite Problem 8.2.2 as

$$\text{maximize}_{\{b_i\}} \left\{ \sum_{i=1}^{T} v_i \cdot x_i(b_i) + \min_{\lambda \geq 0} \lambda \cdot \sum_{i=1}^{T} g_i(b_i) \right\}, \tag{8.3.1}$$

in which $\sum_{i=1}^{T} g_i(b_i) \geq 0$ is enforced using the minimization via the dual variable λ that applies an unbounded penalty for the constraint violation. We update, in each iteration t, the bid b_t and the current best penalizing (dual) variable dual variable λ_t, described next.

Updating the bid. Based on the formulation in Problem 8.3.1, our algorithm chooses the bid b_t as the maximizer of the penalty-adjusted reward of the *current round*, with the penalty applied by the current dual variable λ_t:

$$b_t = \arg\max_{b \geq 0} \left[\frac{1 + \lambda_t}{\lambda_t} \cdot v_t \cdot x_t(b) - p_t(b) \right] = v_t + \frac{v_t}{\lambda_t}, \tag{8.3.2}$$

where the final step is because of the auction being truthful.[4] The final expression for b_t is consistent with the setting that we first observe only the value v_t before making the bid.

Updating the dual variable. To maintain a meaningful dual variable, we relax the penalty on the constraint violation in Problem 8.3.1 by adding a scaled regularization function $h(\lambda)$. This regularizer prevents λ from getting too large:

$$\max_{\{b_i\}} \left\{ \sum_{i=1}^{T} v_i \cdot x_i(b_i) + \min_{\lambda \geq 0} \left[\lambda \cdot \sum_{i=1}^{T} g_i(b_i) + \frac{h(\lambda)}{\alpha} \right] \right\}, \tag{8.3.3}$$

where $\alpha > 0$ is the scaling factor of the regularizer to be set later. At any iteration t, the value of λ_{t+1} is chosen to be the minimizer of the inner constrained minimization problem (until iteration $i = t$). We choose the generalized negative entropy $h(u) = u \log u - u$, which gives the following

[4] Because it is a truthful auction, we have $\nabla_b x_t(b) \geq 0$, which, when used in the definition of b_t, gives the claimed final step.

expression for λ_{t+1}.

$$\lambda_{t+1} = \arg\min_{\lambda \geq 0} \left[g_t(b_t) \cdot \lambda + \frac{V_h(\lambda, \lambda_t)}{\alpha} \right] = \lambda_1 \cdot \exp\left[-\sum_{i=1}^{t} \alpha \cdot g_i(b_i) \right]. \tag{8.3.4}$$

Through this rule, a net constraint violation (i.e., $\sum_{i=1}^{t} g_i(b_i) \leq 0$) makes the next dual variable (λ_{t+1}) exponential in the net violation, which in turn shrinks the next bid (in Equation (8.3.2)); on the other hand, an accumulated buffer in the net constraint violation (i.e., $\sum_{i=1}^{t} g_i(b_i) > 0$) encourages λ_{t+1} to be small, allowing the next bid to grow. We formalize this notion in Lemma 13 and now display Algorithm 8.3.1 and its guarantee, Theorem 8.3.1.

Algorithm 8.3.1 Bidding under an approximate RoS constraint in a truthful auction (i.i.d. inputs).

1: **Input:** Total time horizon T and requests $\overrightarrow{\gamma}$ from the distribution $\mathcal{P}^T$.
2: **Initialize:** Initial dual variable $\lambda_1 = 1$ and dual mirror descent step size $\alpha = \frac{1}{\sqrt{T}}$.
3: **for** $t = 1, 2, \cdots, T$ **do**
4: Observe the value v_t, and set the bid $b_t = v_t + \frac{v_t}{\lambda_t}$.
5: Observe the price p_t and allocation x_t at b_t, and compute $g_t(b_t) = v_t \cdot x_t(b_t) - p_t(b_t)$.
6: Update the dual variable as $\lambda_{t+1} = \lambda_t \exp\left[-\alpha \cdot g_t(b_t)\right]$.
7: **end for**
8: **return** The sequence $\{b_t\}_{t=1}^{T}$ of generated bids.

Theorem 8.3.1. *With i.i.d. inputs from a distribution $\mathcal{P}$ over a time horizon T, Algorithm 8.3.1 guarantees, for Problem 8.2.2, an RoS constraint violation of at most $2T\sqrt{T}\log T$ and a regret bound of*

$$\text{Regret}(\text{Algorithm 8.3.1}, \mathcal{P}^T) \leq O(\sqrt{T}). \tag{8.3.5}$$

The proof of the theorem is an application of our chief technical results Lemma 13 and Lemma 14, which we sketch next.

8.3.1 Constraint Violation of Algorithm 8.3.1

To conclude that the constraint described by Inequality 8.2.4 is violated by only a small amount in Algorithm 8.3.1, we observe that when the cumulative violation is (non-trivially) larger than $1/\alpha$, the exponential function quickly makes λ_t huge; in turn, our bid $b_t = v_t + \frac{v_t}{\lambda_t}$ prevents us from over-bidding. Formally, we show the following result, later used in Theorem 8.3.1 to obtain the stated constraint violation bound.

Lemma 13. *Consider the sequence $\{\lambda_t\}_{t=1}^{T}$ starting at $\lambda_1 = 1$ and evolving as $\lambda_{t+1} = \lambda_t \exp\left[-\alpha g_t(b_t)\right]$ where $g_t(b_t)$ satisfies $g_t(b_t) \geq -\frac{1}{\lambda_t}$ and $\alpha = \frac{1}{\sqrt{T}}$. Then,*

$$-\sum_{t=1}^{T} g_t(b_t) \leq 2\sqrt{T}\log T.$$

Proof. Equation (8.3.4) implies $\lambda_{t+1} = \exp\left[-\alpha \sum_{t'=1}^{t} g_{t'}(b_{t'})\right]$. If $-\sum_{t=1}^{T} g_t(b_t) \leq \sqrt{T}\log T$, we are done. If this is not the case, let T' be the last time that $-\sum_{t=1}^{T'} g_t(b_t) \leq \sqrt{T}\log T$, so we know for any $t > T'$,

the dual variable λ_t must be larger than T since

$$\lambda_t = \lambda_1 \cdot \exp\left[-\alpha \sum_{t'=1}^{t} g_{t'}(b_{t'})\right] > \exp\left[\alpha \sqrt{T} \log T\right] = T,$$

which suggests

$$-g_t(b_t) \leq \frac{1}{\lambda_t} \leq \frac{1}{T} \qquad \forall t > T'. \tag{8.3.6}$$

Finally, using Inequality 8.3.6 to bound the terms after iteration T' and the fact that there are at most T such iterations gives

$$-\sum_{t=1}^{T} g_t(b_t) = -\sum_{t=1}^{T'} g_t(b_t) - \sum_{t>T'} g_t(b_t) \leq \sqrt{T} \log T + 1 \leq 2\sqrt{T} \log T.$$

□

8.3.2 Regret of Algorithm 8.3.1

As noted in Section 6.1, our RoS constraint differs fundamentally from the well-studied budget constraint, which is why regret guarantees from the latter (e.g., [BLM20]) do not transfer to our setting. In the primal-dual framework, the dual variables corresponding to the budget constraint are naturally upper bounded by a constant, which is critical for sub-linear regret bounds with the existing analysis framework. Such bounds do not exist in general, which is why efforts to extend the approach beyond the budget constraint have used much more complicated algorithms (albeit getting weaker regret guarantees, e.g., [CCM+22]).

Our key insight is to recognize the RoS constraint's special structures that enable near-optimal results even with unbounded dual variables in the basic primal-dual method (which has no reason to perform well on arbitrary constraints). This is captured in Proposition 8.3.3 and Lemma 14.

We first state the following upper bound on the regret whose proof follows the primal-dual framework of Theorem 1 in [BLM20]; our effort in the remaining section is towards bounding the right-hand side of this result.

Proposition 8.3.2. *With i.i.d. inputs from a distribution $\mathcal{P}$ over a time horizon T, the regret of Algorithm 8.3.1 on Problem 8.2.2 is bounded by*

$$\textsf{Regret}(\textit{Algorithm 8.3.1}, \mathcal{P}^T) \leq \mathbb{E}_{\vec{\gamma} \sim \mathcal{P}^T}\left[\sum_{t \in [T]} \lambda_t \cdot g_t(b_t)\right],$$

where g_t and λ_t are as defined in Line 5 and Line 6 of Algorithm 8.3.1. We note that the bound on the right-hand side can be negative since Algorithm 8.3.1 does not guarantee $\sum_{t=1}^{T} g_t(b_t) \geq 0$, but we do show a bound on the worst-case constraint violation in Lemma 13.

Bounding the regret then requires bounding $\sum_{t=1}^{T} \lambda_t \cdot g_t(b_t)$. We start with the following structural lemma about the gradient.

Proposition 8.3.3. *Let g_t be as defined in Equation (8.2.3) and p_t be as defined in Equation (8.2.1). Let $b_t \leq \frac{1+\lambda_t}{\lambda_t} \cdot v_t$. Then we have*

$$\max(-1/\lambda_t, -1) \leq g_t(b_t) \leq v_t \cdot x_t(b_t).$$

Proof. The non-negativity of v_t and x_t along with the bound $p_t(b) \leq 1$ immediately give $g_t(b_t) \geq -1$. Further, the non-negativity of p_t from Equation (8.2.1) gives the upper bound $g_t(b_t) \leq v_t \cdot x_t(b_t)$. Using the expression for p_t from Equation (8.2.1), we may write $g_t(b_t) = (v_t - b_t) \cdot x_t(b_t) + \int_{z=0}^{b} x_t(z)dz$. The non-negativity of x_t implies $\int_{z=0}^{b} x_t(z)dz \geq 0$. Finally, applying the stated assumption $b_t \leq \frac{1+\lambda_t}{\lambda_t} \cdot v_t$ and $v_t, x_t \leq 1$ gives $g_t(b_t) \geq -\frac{1}{\lambda_t}$, as claimed. □

Proposition 8.3.3 gives us the following intuition for the main technical component (Lemma 14) of the regret bound. When $g_t(b_t) = -\frac{1}{\lambda_t}$, it means that the dual can grow at most linearly as seen by $\lambda_{t+1} = \lambda_t \exp[-\alpha g_t(b_t)] \approx \lambda_t + \alpha$. On the other hand, a large positive $g_t(b_t)$ decreases the dual multiplicatively. This suggests there cannot be too many iterations where both the dual and the (positive) gradient are large, and Proposition 8.3.2 suggests these are exactly the iterations that contribute a lot to the regret bound. We capture the above intuition in the following lemma.

Lemma 14. *For any input sequence $\overrightarrow{\gamma}$ of length T, running Algorithm 8.3.1 on Problem 8.2.2 generates sequences $\{b_t\}_{t=1}^T$, $\{g_t\}_{t=1}^T$ and $\{\lambda_t\}_{t=1}^T$ such that for any $\lambda \geq 0$, we have*

$$\sum_{t=1}^{T} g_t(b_t) \cdot (\lambda_t - \lambda) \leq \frac{V_h(\lambda, \lambda_1)}{\alpha} + (1 - \lambda_T) + O(\sqrt{T}),$$

where $V_h(\cdot, \cdot)$ is the Bregman divergence with $h(u) = u \log u - u$.

Proof. We first split $g_t(b_t) \cdot (\lambda_t - \lambda)$, for any $\lambda \geq 0$, as:

$$\alpha g_t(b_t) \cdot (\lambda_t - \lambda) = \alpha g_t(b_t) \cdot (\lambda_t - \lambda_{t+1}) + \alpha g_t(b_t) \cdot (\lambda_{t+1} - \lambda). \tag{8.3.7}$$

The dual variable update in Algorithm 8.3.1 implies

$$g_t(b_t) = \alpha^{-1} \log(\lambda_t / \lambda_{t+1}). \tag{8.3.8}$$

For the mirror map $h(u) = u \log u - u$, the Bregman divergence $V_h(y, x) = h(y) - h(x) - h'(x) \cdot (y - x)$ is

$$V_h(y, x) = y \log(y/x) - y + x. \tag{8.3.9}$$

We may then bound Equation (8.3.7) as follows:

$$\begin{aligned} \alpha g_t(b_t) \cdot (\lambda_t - \lambda) - [V_h(\lambda, \lambda_t) - V_h(\lambda, \lambda_{t+1})] &= \alpha g_t(b_t) \cdot (\lambda_t - \lambda_{t+1}) - V_h(\lambda_{t+1}, \lambda_t) \\ &\leq \alpha g_t(b_t) \cdot (\lambda_t - \lambda_{t+1}) - \frac{(\lambda_t - \lambda_{t+1})^2}{2 \max(\lambda_t, \lambda_{t+1})} \\ &\leq \frac{1}{2} \alpha^2 g_t(b_t)^2 \cdot \max(\lambda_t, \lambda_{t+1}), \end{aligned} \tag{8.3.10}$$

where the first step is by applying Equation (8.3.9) and Equation (8.3.8) to Equation (8.3.7), the second step is by the local strong convexity of V_h as shown in [AZO14] (see Lemma 15 for completeness), and the final step is by Cauchy-Schwarz inequality. We now show that

$$\frac{\alpha}{2} g_t(b_t)^2 \max(\lambda_t, \lambda_{t+1}) \leq (\lambda_t - \lambda_{t+1}) + 2\alpha + \frac{e}{2\alpha T}, \tag{8.3.11}$$

which when plugged into Inequality 8.3.10, summing over $t = 1, 2, \ldots, T$ (and telescoping), and

using the values of λ_1 and α from Algorithm 8.3.1 yields the claimed bound. We now prove Inequality 8.3.11 in a case-wise manner.

C.1 Assume $g_t(b_t) \geq 0$. Then, the inequality $g_t(b_t) \leq 1$ (from Proposition 8.3.3) and our choice of $\alpha = \frac{1}{\sqrt{T}}$ imply $\alpha g_t(b_t) \leq \frac{1}{\sqrt{T}}$, which in turn implies

$$\lambda_{t+1} = \lambda_t \exp\left[-\alpha g_t(b_t)\right] \leq \lambda_t - \frac{\alpha}{2}\lambda_t \cdot g_t(b_t), \tag{8.3.12}$$

where we used $\exp(-x) \leq 1 - x/2$ for $x \in [0, 1.5]$. Therefore, we have

$$\frac{\alpha}{2} g_t(b_t)^2 \max(\lambda_t, \lambda_{t+1}) = \frac{\alpha}{2} g_t(b_t)^2 \lambda_t \leq \lambda_t - \lambda_{t+1}, \tag{8.3.13}$$

where the first step uses $\lambda_t \geq \lambda_{t+1}$ (since this case assumes $g_t(b_t) \geq 0$), and the second step uses $g_t(b_t) \leq 1$ (from Proposition 8.3.3) and Inequality 8.3.12.

C.2 Assuming $g_t(b_t) < 0$ gives the following bound:

$$g_t(b_t)^2 \max(\lambda_t, \lambda_{t+1}) = g_t(b_t)^2 \lambda_{t+1} \leq \frac{1}{\lambda_t} \cdot \lambda_{t+1}, \tag{8.3.14}$$

where the first step uses $\lambda_{t+1} \geq \lambda_t$ (since this case assumes $g_t(b_t) < 0$), and the second uses $0 \geq g_t(b_t) \geq \max(-1, -1/\lambda_t)$ (in turn from Proposition 8.3.3).

Since $g_t(b_t) \geq -1$ and $\alpha = \frac{1}{\sqrt{T}}$, we have $-\alpha g_t(b_t) \leq 1$. This implies

$$\lambda_{t+1} = \lambda_t \exp\left[-\alpha g_t(b_t)\right] \leq e\lambda_t. \tag{8.3.15}$$

Plugging Inequality 8.3.15 back into Inequality 8.3.14 gives

$$\frac{1}{2}\alpha g_t(b_t)^2 \max(\lambda_t, \lambda_{t+1}) \leq \frac{e}{2\alpha T}. \tag{8.3.16}$$

Using $-\alpha g_t(b_t) \leq 1$ again allows us to claim

$$\lambda_{t+1} = \lambda_t \exp\left[-\alpha g_t(b_t)\right] \leq \lambda_t(1 - 2\alpha g_t(b_t)),$$

where we used $\exp(x) \leq 1 + 2x$ for $x \in [0, 1]$. Again applying $\lambda_t g_t(b_t) \geq -1$ gives

$$\lambda_{t+1} - \lambda_t \leq 2\alpha. \tag{8.3.17}$$

This proves Inequality 8.3.11 and finishes the proof of the lemma. □

Lemma 15 ([AZO14]). *The Bregman divergence of the generalized negative entropy satisfies "local strong convexity": for any $x, y > 0$,*

$$V_h(y, x) = y \log(y/x) + x - y \geq \frac{1}{2\max(x, y)} \cdot (y - x)^2.$$

Proof. The claimed inequality is equivalent to

$$t \log t \geq (t - 1) + \frac{1}{2\max(1, t)} \cdot (t - 1)^2 \tag{8.3.18}$$

for $t > 0$. Suppose $t \geq 1$. Then, choosing $u = 1 - 1/t$, Inequality 8.3.18 is equivalent to $-\log(1-u) \geq u + \frac{1}{2}u^2$, for $u \in [0, 1)$, which holds by Taylor series. Suppose $0 < t \leq 1$. Then Inequality 8.3.18 is equivalent to $\log t - \frac{1}{2}\left(t - \frac{1}{t}\right) \geq 0$, which may be checked by observing that the function is decreasing and equals zero at $t = 1$. This completes the proof of the claim. □

8.4 Strict RoS Constraint

Having started with an algorithm with non-zero (but bounded) violation of the RoS constraint, we now describe one *strictly* obeying it. Our core idea (displayed in Algorithm 8.4.1) is as follows.

Suppose we have an algorithm (say, Alg), which can guarantee an at most v_{RoS} violation of the RoS constraint on any sequence $\overrightarrow{\gamma}$ of input requests. We start by bidding the true value $b_t = v_t$ in the initial iterations $t = 1, \ldots, K(\overrightarrow{\gamma})$ for some $K(\overrightarrow{\gamma})$ — we call this sequence of iterations the *first phase*. In a truthful auction, this choice of bids guarantees $g_t(b_t) = v_t \cdot x_t(b_t) - p_t(b_t) \geq 0$ for all $t \leq K(\overrightarrow{\gamma})$. In other words, the bidder builds up a buffer on the RoS constraint. We continue until the cumulative buffer $\sum_{t=1}^{K(\overrightarrow{\gamma})} g_t(b_t)$ increases to at least v_{RoS}. Starting at iteration $K(\overrightarrow{\gamma}) + 1$, we run Alg afresh (i.e. without accounting for the first phase); recall, this violates the RoS constraint by at most v_{RoS} over the remaining iterations. We refer to this run of Alg as the *second phase*. Since the buffer from the first phase is enough to offset Alg's violation in the second phase, there is no violation of the RoS constraint at the end.

Algorithm 8.4.1 Bidding under a strict RoS constraint in a truthful auction (i.i.d. inputs)

Input: Total time horizon T and requests $\overrightarrow{\gamma}$ from the data distribution $\mathcal{P}^T$.
Initialize: Set $v_{\mathsf{RoS}} = 2\sqrt{T}\log T$ and $t = 1$.
while $\sum_{i=1}^{t-1} g_i(b_i) \leq v_{\mathsf{RoS}}$ **do** ▷ First phase
    Observe the value v_t, and set the bid $b_t = v_t$.
    Observe the price p_t and the allocation x_t at b_t, and compute $g_t(b_t) := v_t \cdot x_t(b_t) - p_t(b_t)$.
    Increment the iteration count $t = t + 1$.
end while
Run Algorithm 8.3.1 with time horizon $T - t$ and the remaining $T - t$ requests from $\overrightarrow{\gamma}$ as input. ▷ Second phase
return The sequence $\{b_t\}_{t=1}^{T}$ of generated bids from both phases.

8.4.1 Analysis of Algorithm 8.4.1

The high-level idea to guarantee a low regret for Algorithm 8.4.1 is to start with the observation that the reward collected by Algorithm 8.4.1 for any $\overrightarrow{\gamma}$ in T steps is at least that collected by Algorithm 8.3.1 in the second phase (i.e., the last $T - K(\overrightarrow{\gamma})$ steps). The second phase suffers (in expectation) a regret bounded by the guarantee of Theorem 8.3.1. We then use the i.i.d. assumption on the input sequence to bound the gap between the expected reward collected by Opt in a sequence of length $T - K(\overrightarrow{\gamma})$ to that in a sequence of length T, which naturally depends on the expected length of $K(\overrightarrow{\gamma})$; finally, we show this expected length is at most $O(\sqrt{T}\log T)$ under a mild technical assumption on the input distribution; this bounds the additional regret accrued over the first phase and thus completes the analysis.

To formally see this, we need two simple technical tools. First, we make the following assumption on the distribution $\mathcal{P}$. Our β is similar to the 'strictly feasible' margin used in the broader literature (e.g. d_g, ρ in [CCM+22]).

Assumption 8.4.1. *Define the parameter β of a distribution $\mathcal{P}$ as follows*

$$\beta = \mathbb{E}_{v,p,x\sim\mathcal{P}}\left[\max(0, v \cdot x(v) - p(v)\right].$$

We assume in our problem β is an absolute constant bounded away from 0 and independent of T.

The parameter β is the expected amount of buffer we accrue per iteration during the first phase. The assumption of β being a constant bounded away from 0 captures the more interesting scenarios of bidding under RoS. For example, when the allocation and price functions of each query arise from a single-item second-price auction, the essence of the problem becomes how best to spend the extra slack $v_t - p_t(v_t)$ gained from queries with $v_t > p(v_t)$. If β is tiny, or in the extreme case of $\beta = 0$, there is nothing to optimize for, and the optimal solution would simply be $b_t = v_t$ all the time.

Since we need to accrue a buffer of size only $O(\sqrt{T}\log T)$, and the expected increment of the buffer is a constant β per iteration, we can show the first phase has $O(\sqrt{T}\log T)$ iterations in expectation.

Proposition 8.4.2. *Under Assumption 8.4.1 for the distribution $\mathcal{P}$, let $K(\overrightarrow{\gamma})$ be the number of iterations in the first phase of Algorithm 8.4.1 for some input sequence $\overrightarrow{\gamma}$. Then, we have*

$$\mathbb{E}_{\overrightarrow{\gamma}\sim\mathcal{P}^T}[K(\overrightarrow{\gamma})] \le O(\sqrt{T}\log T).$$

We also need the following technical statement on the difference in reward collected by Algorithm 8.3.1 and Opt for various lengths of input sequences.

Proposition 8.4.3. *Let $\overrightarrow{\gamma}_\ell \sim \mathcal{P}^\ell$ and $\overrightarrow{\gamma}_r \sim \mathcal{P}^r$ be sequences of lengths ℓ and r, respectively, with $\ell \le r$, of i.i.d. requests each from a distribution $\mathcal{P}$. Then the following inequality holds.*

$$\mathbb{E}_{\overrightarrow{\gamma}_\ell\sim\mathcal{P}^\ell}\left[\mathsf{Reward}(\textit{Algorithm 8.3.1}, \overrightarrow{\gamma}_\ell)\right] \ge \frac{\ell}{r}\mathbb{E}_{\overrightarrow{\gamma}_r\sim\mathcal{P}^r}\left[\mathsf{Reward}(\mathsf{Opt}, \overrightarrow{\gamma}_r)\right] - O(\sqrt{r}).$$

The above two results help us bound the regret from the first phase, and we can use Theorem 8.3.1 to bound the regret due to the second phase. Altogether we get the main result below.

Theorem 8.4.4. *With i.i.d. inputs from a distribution $\mathcal{P}$ over a time horizon T, the regret of Algorithm 8.4.1 on Problem 8.2.2 is, under Assumption 8.4.1, bounded by*

$$\mathsf{Regret}(\textit{Algorithm 8.4.1}, \mathcal{P}^T) \le O(\sqrt{T}\log T).$$

Further, there is no violation of the RoS constraint in Problem 8.2.2.

Proof. The claim on constraint violation follows by design of the algorithm: we collect a constraint violation buffer of at least v_{RoS} before starting the second phase, in which we are guaranteed to violate the constraint by an additive factor of at most v_{RoS}.

We now prove the claimed regret bound by combining a lower bound on the expected reward and an upper bound on the expected optimum. To lower bound the algorithm's reward, we note that it is at least the reward from the second phase.

Let $K(\overrightarrow{\gamma})$ be the random variable that represents the last iteration of the first phase, after which we run Algorithm 8.3.1. In this proof, we use $\overrightarrow{\gamma}_{a:b}$ to denote the sequence $\overrightarrow{\gamma}$ from time steps a through b; when these end points do not matter, we simply denote a length T sequence as $\overrightarrow{\gamma}$. With this

notation, we have for any sequence $\overrightarrow{\gamma}$:

$$\textsf{Reward}(\textit{Algorithm 8.4.1}, \overrightarrow{\gamma}) \geq \sum_{t=K(\overrightarrow{\gamma})+1}^{T} v_t \cdot x_t(b_t).$$

Then taking expectations on both sides and using conditional expectations gives

$$\begin{aligned}
\mathbb{E}_{\overrightarrow{\gamma}\sim\mathcal{P}^T}\left[\textsf{Reward}(\textit{Algorithm 8.4.1}, \overrightarrow{\gamma})\right] &\geq \mathbb{E}_k\left[\mathbb{E}_{\overrightarrow{\gamma}\sim\mathcal{P}^T|K(\overrightarrow{\gamma})=k}\left[\sum_{t=k+1}^{T} v_t \cdot x_t(b_t) \mid K(\overrightarrow{\gamma}) = k\right]\right] \\
&= \mathbb{E}_k\left[\mathbb{E}_{\overrightarrow{\gamma}_{k+1:T}\sim\mathcal{P}^{T-k}}\left[\textsf{Reward}(\textit{Algorithm 8.3.1}, \overrightarrow{\gamma}_{k+1:T})\right]\right] \\
&\geq \mathbb{E}_k\left[\frac{T-k}{T} \cdot \mathbb{E}_{\overrightarrow{\gamma}\sim\mathcal{P}^T}\left[\textsf{Reward}(\textsf{Opt}, \overrightarrow{\gamma})\right]\right] - O(\sqrt{T}), \qquad (8.4.1)
\end{aligned}$$

where the third step is due to the requests all being i.i.d. and the fact that we run Algorithm 8.3.1 fresh in the second phase, and the fourth step is by Proposition 8.4.3. Finally, plugging Inequality 8.4.1 into the definition of Regret from Equation (8.2.8) and simplifying:

$$\begin{aligned}
\textsf{Regret}(\textit{Algorithm 8.4.1}, \mathcal{P}^T) &\leq \mathbb{E}_{\overrightarrow{\gamma}\sim\mathcal{P}^T}\left[K(\overrightarrow{\gamma})\right] \cdot \frac{\mathbb{E}_{\overrightarrow{\gamma}\sim\mathcal{P}^T}\left[\textsf{Reward}(\textsf{Opt}, \overrightarrow{\gamma})\right]}{T} + O(\sqrt{T}) \\
&\leq \mathbb{E}_{\overrightarrow{\gamma}\sim\mathcal{P}^T}\left[K(\overrightarrow{\gamma})\right] + O(\sqrt{T}) = O(\sqrt{T}\log T),
\end{aligned}$$

where the third step uses $\textsf{Reward}(\textsf{Opt}, \overrightarrow{\gamma}) \leq T$, and the last step uses Proposition 8.4.2 and that β is a constant (Assumption 8.4.1). □

Despite the two-phased structure of the algorithm, the used $v_{\textsf{RoS}}$ can be pessimistic and make us run the first phase unnecessarily longer. A more practical implementation can break the first phase into smaller chunks and intermingle them with the execution of Algorithm 8.3.1 on demand. That is, whenever we are about to violate the RoS constraint in Algorithm 8.3.1, we put it on hold and bid exactly v_t for some iterations until we build up a certain amount of buffer and resume Algorithm 8.3.1, where these special iterations are ignored (i.e., won't affect the dual variable updates). Intuitively this can perform better than Algorithm 8.4.1, although without a rigorous regret guarantee.

8.5 RoS and Budget Constraints

In this section, we combine our techniques from Section 8.3 and Section 8.4 with those of [BLM20] to obtain algorithms that satisfy *both* the RoS and budget constraints (i.e., Problem 8.2.5).

8.5.1 Approximate RoS and Strict Budget Constraints

We start with a bidding algorithm that satisfies the budget constraint exactly and the RoS constraint up to some small violation in the worst case. Similar to the bidding rule in Equation (8.3.2), the candidate bid for this algorithm is the one maximizing the price-adjusted reward, with one dual variable for each constraint:

$$b_t = \arg\max_{b\geq 0}\{v_t \cdot x_t(b) + \lambda_t \cdot (v_t \cdot x_t(b) - p_t(b)) - \mu_t \cdot p_t(b)\} = \frac{1+\lambda_t}{\mu_t + \lambda_t} \cdot v_t, \qquad (8.5.1)$$

where the final equation holds by the definition of a truthful auction. Since the budget constraint is strict, the candidate bid given by Equation (8.5.1) is used as the final bid in this iteration only if we are not close to exhausting the total budget, as formalized in Line 4 of Algorithm 8.5.1. Similar to Algorithm 8.3.1, the Lagrange multipliers λ_t and μ_t enforce the RoS and budget constraints respectively.

Algorithm 8.5.1 Bidding under an approximate RoS and a strict budget constraint in a truthful auction (i.i.d. inputs)

1: **Input:** Total time horizon T, requests $\overrightarrow{\gamma}$ i.i.d. from the distribution $\mathcal{P}^T$, total budget ρT.
2: **Initialize:** Initial dual variable $\lambda_1 = 1$, $\mu_1 = 0$, total initial budget $B_1 := \rho T$, dual mirror descent step size $\alpha = \frac{1}{\sqrt{T}}$ and $\eta = \frac{1}{(1+\rho^2)\sqrt{T}}$.
3: **for** $t = 1, 2, \cdots, T$ **do**
4: Observe the value v_t, and set the bid

$$b_t = \begin{cases} \frac{1+\lambda_t}{\mu_t+\lambda_t} \cdot v_t & \text{if } B_t \geq 1 \\ 0 & \text{otherwise} \end{cases}.$$

5: Compute $g_t(b_t) = v_t \cdot x_t(b_t) - p_t(b_t)$.
6: Update the dual variable of the RoS constraint $\lambda_{t+1} = \lambda_t \exp\left[-\alpha \cdot g_t(b_t)\right]$.
7: Compute $g'_t(b_t) = \rho - p_t(b_t)$.
8: Update the dual variable of the budget constraint as $\mu_{t+1} := \text{Proj}_{\mu \geq 0}(\mu_t - \eta \cdot g'_t(b_t))$.
9: Update the leftover budget $B_{t+1} = B_t - p_t(b_t)$.
10: **end for**
11: **return** The sequence $\{b_t\}_{t=1}^T$ of bids.

Algorithm 8.5.1 may be interpreted as combining our Algorithm 8.3.1 with Algorithm 1 of [BLM20]. The analysis of the regret bound also follows the outline of the main proof of [BLM20], integrating it with our regret bound for Algorithm 8.3.1 from Theorem 8.3.1. Intuitively the integration is straightforward since the analyses of both methods are linear in nature, allowing us to easily decompose the intermediate regret bound from the primal-dual framework into two components corresponding to the two constraints. We then bound them with the tools from earlier sections to handle the RoS constraint and those from [BLM20] for the budget constraint.

As for the constraint violation guarantees, the budget constraint is always satisfied by design, and the RoS violation bound follows from a simple corollary of Lemma 13, which applies here since the extra budget constraint makes our bid only *more* conservative than that of Algorithm 8.3.1. We formally collect and state these results in Theorem 8.5.1, deferring the proof to the supplemental material.

Theorem 8.5.1. *With i.i.d. inputs from a distribution $\mathcal{P}$ over a time horizon T, the regret of Algorithm 8.5.1 on Problem 8.2.5 is bounded by*

$$\text{Regret}(\textit{Algorithm 8.5.1}, \mathcal{P}^T) \leq O(\sqrt{T}).$$

Further, Algorithm 8.5.1 incurs a violation of at most $O(\sqrt{T}\log T)$ of the RoS constraint and no violation of the budget constraint.

8.5.2 Strict RoS and Strict Budget Constraints

In the case when we impose both strict RoS and strict budget constraints, we essentially combine the key ideas from Algorithm 8.4.1 and Algorithm 8.5.1: We keep bidding the value until we accumulate

a sufficient buffer on the RoS constraint; following this phase, we run Algorithm 8.5.1, which, as explained in the preceding section, imposes strict budget and approximate RoS constraints.

Algorithm 8.5.2 Bidding under a strict RoS and a strict budget constraint in a truthful auction (i.i.d. inputs)

1: **Input:** Total time horizon T, requests $\overrightarrow{\gamma}$ i.i.d. from the distribution $\mathcal{P}^T$, total budget ρT.
2: **Initialize:** Set the initial buffer $g_0(b_0) = 0$, $v_{\mathsf{RoS}} = 2\sqrt{T}\log T$, initial total budget $B_1 = \rho T$, and iteration $t = 1$.
3: **while** $\sum_{i=0}^{t-1} g_i(b_i) \le v_{\mathsf{RoS}}$ **do** ▷ First phase
4: Observe the value v_t, and set the bid $b_t = v_t$.
5: Observe the price p_t and the allocation x_t at b_t, and compute $g_t(b_t) := v_t \cdot x_t(b_t) - p_t(b_t)$.
6: Update the total budget to $B_{t+1} = B_t - p_t(b_t)$.
7: Increment the iteration count $t = t + 1$.
8: **if** $t \ge \rho T$ **then**
9: Exit algorithm.
10: **end if**
11: **end while**
12: Run Algorithm 8.5.1 with time horizon $T - t$ and the remaining $T - t$ requests from $\overrightarrow{\gamma}$ as input and initial total budget B_t. ▷ Second phase
13: **return** The sequence $\{b_t\}_{t=1}^T$ of generated bids.

Similar to Algorithm 8.4.1, the RoS constraint is not violated; the budget constraint is also respected via Line 8. The regret analysis follows a strategy similar to that of the proofs of Theorem 8.5.1 and Theorem 8.4.4. Our main result of this section follows, with its proof in Section G.2.

Theorem 8.5.2. *With i.i.d. inputs from a distribution $\mathcal{P}$ over a time horizon T, the regret of Algorithm 8.5.2 on Problem 8.2.5 is, under Assumption 8.4.1, bounded by*

$$\mathsf{Regret}(\textit{Algorithm 8.5.2}, \mathcal{P}^T) \le O(\sqrt{T}\log T).$$

Further, Algorithm 8.5.2 suffers no constraint violation of either the RoS or budget constraint.

8.6 Conclusion

We design algorithms for the online bidding problem under budget and Return-on-Spend constraints, a problem of tremendous practical importance in online advertising. In particular, we achieve $O(\sqrt{T})$ average regret compared to the offline optimal solution in the stochastic i.i.d. model and up to $O(\sqrt{T}\log T)$ violation on the RoS constraint *in any outcome*; under a mild assumption on the distribution, this result can be further improved to guarantee *no* constraint violation. Our algorithm is simple and easily implementable using the existing pacing controller [CKP+22, CKSSM22].

Our key novelty lies in the insight that when a low-regret learning dynamic is employed in the setting of designing primal-dual online algorithms, the gradients encountered during the dual updates may depend on the choices of the primal side of the algorithm (rather than being fully adversarial as in online learning). Our result is an example where one can exploit problem-specific structures to come up with tailored low-regret methods and achieve qualitatively stronger guarantees than those given by generic black-box methods. As low-regret methods have become workhorses in designing online constrained optimization algorithms, we are optimistic that this high-level idea may be broadly applicable.

Bibliography

[AB15] Alekh Agarwal and Leon Bottou. A lower bound for the optimization of finite sums. In *International conference on machine learning*, pages 78–86. PMLR, 2015. 4.1

[ABH15] Emmanuel Abbe, Afonso S Bandeira, and Georgina Hall. Exact recovery in the stochastic block model. *IEEE Transactions on Information Theory*, 62(1):471–487, 2015. 1.1, 2.1

[ABM19] Gagan Aggarwal, Ashwinkumar Badanidiyuru, and Aranyak Mehta. Autobidding with constraints. In *International Conference on Web and Internet Economics*, pages 17–30. Springer, 2019. 8.1

[AD14] Shipra Agrawal and Nikhil R Devanur. Fast algorithms for online stochastic convex programming. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 1405–1424. SIAM, 2014. 8.1.3

[AH53] Hoffman AJ and Wielandt HW. The variation of the spectrum of a normal matrix. *Duke Mathematical Journal*, 20(1):37, 1953. B.1.2

[AHK05] Sanjeev Arora, Elad Hazan, and Satyen Kale. Fast algorithms for approximate semidefinite programming using the multiplicative weights update method. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, pages 339–348. IEEE, 2005. 2.1.1

[AHR08] Jacob Abernethy, Elad E Hazan, and Alexander Rakhlin. Competing in the dark: An efficient algorithm for bandit linear optimization. In *21st Annual Conference on Learning Theory, COLT 2008*, 2008. 7.1.3

[AJRV20] Pranjal Awasthi, Himanshu Jain, Ankit Singh Rawat, and Aravindan Vijayaraghavan. Adversarial robustness via robust low rank representations. *CoRR*, abs/2007.06555, 2020. 1.1, 2.1

[AK07] Sanjeev Arora and Satyen Kale. A combinatorial, primal-dual approach to semidefinite programs. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC)*, 2007. 1.1, 2.1, 2.1, 2.1, 2.1.1, 2.1.1, 2.2, 2.2, 3.1.2, A.3.1, A.3.1

[AKM+21] Kyriakos Axiotis, Adam Karczmarz, Anish Mukherjee, Piotr Sankowski, and Adrian Vladu. Decomposable submodular function minimization via maximum flow. In *International Conference on Machine Learning*, pages 446–456. PMLR, 2021. 4.1.1

[ALO16a] Zeyuan Allen Zhu, Yin Tat Lee, and Lorenzo Orecchia. Using optimization to obtain a width-independent, parallel, simpler, and faster positive SDP solver. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1824–1831, 2016. 2.1.1

[ALO16b] Zeyuan Allen Zhu, Yin Tat Lee, and Lorenzo Orecchia. Using optimization to obtain a width-independent, parallel, simpler, and faster positive SDP solver. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms(SODA)*, pages 1824–1831. https://arxiv.org/pdf/1507.02259.pdf, 2016. 3.1.2, B.1.1

[Ans00] Kurt M Anstreicher. The volumetric barrier for semidefinite programming. *Mathematics of Operations Research*, 25(3):365–380, 2000. 1.1, 3.1, 3.1.1, 3.1, 3.2, 3.2

[AO15] Zeyuan Allen-Zhu and Lorenzo Orecchia. Using optimization to break the epsilon barrier: A faster and simpler width-independent algorithm for solving positive linear programs in parallel. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '15, 2015. 2.2.2

[AR04] Anders H Andersen and William S Rayens. Structure-seeking multilinear methods for the analysis of fmri data. *NeuroImage*, 22(2):728–739, 2004. 1.3, 6.1

[ARV09] Sanjeev Arora, Satish Rao, and Umesh Vazirani. Expander flows, geometric embeddings and graph partitioning. *Journal of the ACM (JACM)*, 56(2):1–37, 2009. 1.1, 3.1

[AV95] David S Atkinson and Pravin M Vaidya. A cutting plane algorithm for convex programming that uses analytic centers. *Mathematical Programming*, 69(1-3):1–43, 1995. 3.1.2

[AWL+22] Rui Ai, Chang Wang, Chenchen Li, Jinshan Zhang, Wenhan Huang, and Xiaotie Deng. No-regret learning in repeated first-price auctions with budget constraints, 2022. 8.1.3

[AZ17] Zeyuan Allen-Zhu. Katyusha: The first direct acceleration of stochastic gradient methods. *The Journal of Machine Learning Research*, 18(1):8194–8244, 2017. 4.1, 4.1, 4.1, 4.1.2

[AZ18] Zeyuan Allen-Zhu. How to make the gradients small stochastically: Even faster convex and nonconvex sgd. *Advances in Neural Information Processing Systems*, 31, 2018. 5.1

[AZL17a] Zeyuan Allen-Zhu and Yuanzhi Li. Follow the compressed leader: faster online learning of eigenvectors and faster mmwu. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 116–125, 2017. 2.1, 2.1.1, 2.1.1

[AZL17b] Zeyuan Allen-Zhu and Yuanzhi Li. Follow the compressed leader: faster online learning of eigenvectors and faster mmwu. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 116–125, 2017. 3.1.2

[AZLSW17] Zeyuan Allen-Zhu, Yuanzhi Li, Aarti Singh, and Yining Wang. Near-optimal design of experiments via regret minimization. In *Proceedings of the 34th International Conference on Machine Learning*, 2017. 7.1.3

[AZO14] Zeyuan Allen-Zhu and Lorenzo Orecchia. Using optimization to break the epsilon barrier: A faster and simpler width-independent algorithm for solving positive linear programs in parallel. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 1439–1456. SIAM, 2014. 8.1.2, 8.3.2, 15

[AZO19] Zeyuan Allen-Zhu and Lorenzo Orecchia. Nearly linear-time packing and covering lp solvers. *Mathematical Programming*, 175(1):307–353, 2019. 6.1, 6.1, 6.2

[AZQRY16] Zeyuan Allen-Zhu, Zheng Qu, Peter Richtárik, and Yang Yuan. Even faster accelerated coordinate descent using non-uniform sampling. In *International Conference on Machine Learning*, pages 1110–1119. PMLR, 2016. 6.1

[AZY16] Zeyuan Allen-Zhu and Yang Yuan. Improved svrg for non-strongly-convex or sum-of-non-convex objectives. In *International conference on machine learning*, pages 1080–1089. PMLR, 2016. 1.2, 4.1

[B+15] Sébastien Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015. A.2.2, A.4.6

[Bac22] Francis Bach. Sum-of-squares relaxations for information theory and variational inference. *arXiv preprint arXiv:2206.13285*, 2022. 1.1

[Ban19] Nikhil Bansal. On a generalization of iterated and randomized rounding. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1125–1135. https://arxiv.org/pdf/1811.01597.pdf, 2019. 1.1, 3.1

[BB88] Jonathan Barzilai and Jonathan M Borwein. Two-point step size gradient methods. *IMA journal of numerical analysis*, 8(1):141–148, 1988. 6.1

[BBL+07] Michael W Berry, Murray Browne, Amy N Langville, V Paul Pauca, and Robert J Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational statistics & data analysis*, 52(1):155–173, 2007. 1.3, 6.1

[BBN13] Michel Baes, Michael Bürgisser, and Arkadi Nemirovski. A randomized mirror-prox method for solving structured large-scale matrix saddle-point problems. *SIAM Journal on Optimization*, 23(2):934–962, 2013. 1.1, 2.1.1

[BBV04] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004. 4.2.1, 4.2.3

[BC03] Léon Bottou and Yann Cun. Large scale online learning. *Advances in neural information processing systems*, 16, 2003. 1.2, 4.1

[BCH+21] Moshe Babaioff, Richard Cole, Jason Hartline, Nicole Immorlica, and Brendan Lucier. Non-quasi-linear agents in quasi-linear mechanisms. In *12th Innovations in Theoretical Computer Science Conference (ITCS 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021. 8.1

[BCI+07] Christian Borgs, Jennifer Chayes, Nicole Immorlica, Kamal Jain, Omid Etesami, and Mohammad Mahdian. Dynamics of bid optimization in online advertisement auctions. In *Proceedings of the 16th international conference on World Wide Web*, pages 531–540, 2007. 8.1.3

[BCL+20] James V Burke, Frank E Curtis, Adrian S Lewis, Michael L Overton, and Lucas EA Simões. Gradient sampling methods for nonsmooth optimization. In *Numerical Nonsmooth Optimization*, pages 201–225. Springer, 2020. 5.1.1, 5.2

[BCS97] Peter Bürgisser, Michael Clausen, and Mohammad A Shokrollahi. *Algebraic complexity theory*, volume 315. Springer Science & Business Media, 1997. B.2.3

[BCSZ14] Afonso S Bandeira, Moses Charikar, Amit Singer, and Andy Zhu. Multireference alignment using semidefinite programming. In *Proceedings of the 5th conference on Innovations in theoretical computer science*, pages 459–470, 2014. 1.1, 2.1

[BDG16] Nikhil Bansal, Daniel Dadush, and Shashwat Garg. An algorithm for komlós conjecture matching banaszczyk. In *57th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 788–799. https://arxiv.org/pdf/1605.02882.pdf, 2016. 1.1, 3.1

[BDM+20] Vladimir Braverman, Petros Drineas, Cameron Musco, Christopher Musco, Jalaj Upadhyay, David P Woodruff, and Samson Zhou. Near optimal linear algebra in the online and sliding window models. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, 2020. 1.3, 7.1.3

[BDM+21a] Santiago Balseiro, Yuan Deng, Jieming Mao, Vahab Mirrokni, and Song Zuo. Robust auction design in the auto-bidding world. *Advances in Neural Information Processing Systems*, 34:17777–17788, 2021. 8.1

[BDM+21b] Santiago R Balseiro, Yuan Deng, Jieming Mao, Vahab S Mirrokni, and Song Zuo. The landscape of auto-bidding auctions: Value versus utility maximization. In *Proceedings of the 22nd ACM Conference on Economics and Computation*, pages 132–133, 2021. 8.1

[BE15] Sébastien Bubeck and Ronen Eldan. The entropic barrier: a simple and optimal universal self-concordant barrier. In *Conference on Learning Theory*, 2015. 5, 9, 4.4.1, 4.4.2, 7.1.3

[BEGFB94] Stephen Boyd, Laurent El Ghaoui, Eric Feron, and Venkataramanan Balakrishnan. *Linear matrix inequalities in system and control theory*. SIAM, 1994. 1.1

[Ber73] Dimitri P Bertsekas. Stochastic optimization problems with nondifferentiable cost functionals. *Journal of Optimization Theory and Applications*, 12(2):218–231, 1973. 5.1.1

[BEZ08] Alfred M Bruckstein, Michael Elad, and Michael Zibulevsky. On the uniqueness of nonnegative sparse solutions to underdetermined systems of equations. *IEEE Transactions on Information Theory*, 54(11):4813–4820, 2008. 6.1

[BFG21] Ashwinkumar Badanidiyuru, Zhe Feng, and Guru Guruganesh. Learning to bid in contextual first price auctions. *CoRR*, abs/2109.03173, 2021. 8.1.3

[BG17] Nikhil Bansal and Shashwat Garg. Algorithmic discrepancy beyond partial coloring. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 914–926. https://arxiv.org/pdf/1611.01805.pdf, 2017. 1.1, 3.1

[BG19] Santiago R Balseiro and Yonatan Gur. Learning in repeated auctions with budgets: Regret minimization and equilibrium. *Management Science*, 65(9):3952–3968, 2019. 8.1, 8.1.3

[BGJR88] Francisco Barahona, Martin Grötschel, Michael Jünger, and Gerhard Reinelt. An application of combinatorial optimization to statistical physics and circuit layout design. *Operations Research*, 36(3):493–513, 1988. 1.1, 2.1

[BGVV14] Silouanos Brazitikos, Apostolos Giannopoulos, Petros Valettas, and Beatrice-Helen Vritsiou. *Geometry of isotropic convex bodies*, volume 196. American Mathematical Soc., 2014. 4.5.4

[BHS05] Michel Benaïm, Josef Hofbauer, and Sylvain Sorin. Stochastic approximations and differential inclusions. *SIAM Journal on Control and Optimization*, 44(1):328–348, 2005. 1.2, 5.1

[BJ97] Rasmus Bro and Sijmen Jong. A fast non-negativity-constrained least squares algorithm. *Journal of Chemometrics*, 11:393–401, 09 1997. 1.3, 6.1, 6.1

[BKL+20] Sébastien Bubeck, Bo'az Klartag, Yin Tat Lee, Yuanzhi Li, and Mark Sellke. Chasing nested convex bodies nearly optimally. In *Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1496–1508, 2020. 4.2.9

[BKS18] Ashwinkumar Badanidiyuru, Robert Kleinberg, and Aleksandrs Slivkins. Bandits with knapsacks. *Journal of the ACM (JACM)*, 65(3):1–55, 2018. 8.1.3

[Blä13] Markus Bläser. Fast matrix multiplication. *Theory of Computing*, pages 1–60, 2013. B.2.3

[BLM20] Santiago Balseiro, Haihao Lu, and Vahab Mirrokni. Dual mirror descent for online allocation problems. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 613–628. PMLR, 13–18 Jul 2020. 8.1.1, 8.1.2, 8.1.2, 8.1.3, 8.3.2, 8.5, 8.5.1

[BLO02] J.V. Burke, A.S. Lewis, and M.L. Overton. Approximating subdifferentials by random sampling of gradients. *Mathematics of Operations Research*, 27(3):567–584, 2002. 5.1

[BLO05] James V Burke, Adrian S Lewis, and Michael L Overton. A robust gradient sampling algorithm for nonsmooth, nonconvex optimization. *SIAM Journal on Optimization*, 15(3):751–779, 2005. 5.2

[BLSS20] Jan van den Brand, Yin Tat Lee, Aaron Sidford, and Zhao Song. Solving tall dense linear programs in nearly linear time. In *52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*. https://arxiv.org/pdf/2002.02304.pdf, 2020. 3.1.2, 3.3.1, 3.3.1, 3.3.1

[BM20] Sébastien Bubeck and Dan Mikulincer. How to trap a gradient flow. In *Conference on Learning Theory*, pages 940–960. PMLR, 2020. 5.1.1

[Bot12] Léon Bottou. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pages 421–436. Springer, 2012. 1.2, 4.1

[BP20] Jerome Bolte and Edouard Pauwels. A mathematical model for automatic differentiation in machine learning. *arXiv preprint arXiv:2006.02080*, 2020. 5.1.1

[BP21] Jérôme Bolte and Edouard Pauwels. Conservative set valued fields, automatic differentiation, stochastic gradient methods and deep learning. *Mathematical Programming*, 188(1):19–51, 2021. 1.2, 5.1, 5.1.1

[Bra20] Jan van den Brand. A deterministic linear program solver in current matrix multiplication time. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*. https://arxiv.org/pdf/1910.11957.pdf, 2020. 3.1.2

[BT09] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009. 6.5

[BV02] Dimitris Bertsimas and Santosh Vempala. Solving convex programs by random walks. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing (STOC)*, pages 109–115. ACM, 2002. 3.1.2, 4.1

[BV04a] Dimitris Bertsimas and Santosh S. Vempala. Solving convex programs by random walks. *J. ACM*, 51(4):540–556, 2004. 5.3.6

[BV04b] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004. 7.1, A.4.1

[C+47] Augustin Cauchy et al. Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536–538, 1847. 4.1

[CCBKS22] Andrea Celli, Riccardo Colini-Baldeschi, Christian Kroer, and Eric Sodomka. The parity ray regularizer for pacing in auction markets. In *Proceedings of the ACM Web Conference 2022*, WWW '22, page 162–172, New York, NY, USA, 2022. Association for Computing Machinery. 8.1.3

[CCDS20] Rachit Chhaya, Jayesh Choudhari, Anirban Dasgupta, and Supratim Shit. Streaming coresets for symmetric tensor factorization. In *International Conference on Machine Learning*, 2020. 1.3, 7.1.3

[CCLY19] Michael B. Cohen, Ben Cousins, Yin Tat Lee, and Xin Yang. A near-optimal algorithm for approximating the john ellipsoid. In *Proceedings of the Thirty-Second Conference on Learning Theory*, 2019. 7.1.3

[CCM+22] Matteo Castiglioni, Andrea Celli, Alberto Marchesi, Giulia Romano, and Nicola Gatti. A unifying framework for online optimization with long-term constraints. *arXiv preprint arXiv:2209.07454*, 2022. 8.1.3, 8.3.2, 8.4.1

[CDG19] Yu Cheng, Ilias Diakonikolas, and Rong Ge. High-dimensional robust mean estimation in nearly-linear time. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2755–2771. SIAM, https://arxiv.org/pdf/1811.09380.pdf, 2019. 1.1, 3.1

[CDGW19] Yu Cheng, Ilias Diakonikolas, Rong Ge, and David Woodruff. Faster algorithms for high-dimensional robust covariance estimation. In *Conference on Learning Theory (COLT)*. https://arxiv.org/pdf/1906.04661.pdf, 2019. 1.1, 3.1

[CDHS18] Yair Carmon, John C Duchi, Oliver Hinder, and Aaron Sidford. Accelerated methods for nonconvex optimization. *SIAM Journal on Optimization*, 28(2):1751–1772, 2018. 5.1

[CDHS20] Yair Carmon, John C Duchi, Oliver Hinder, and Aaron Sidford. Lower bounds for finding stationary points i. *Mathematical Programming*, 184(1):71–120, 2020. 5.1.1

[CDST19a] Yair Carmon, John C. Duchi, Aaron Sidford, and Kevin Tian. A rank-1 sketch for matrix multiplicative weights. In *Conference on Learning Theory, COLT 2019, 25-28 June 2019, Phoenix, AZ, USA*, pages 589–623, 2019. 1.1, 2.1.1, A.1.2

[CDST19b] Yair Carmon, John C. Duchi, Aaron Sidford, and Kevin Tian. A rank-1 sketch for matrix multiplicative weights. In *Conference on Learning Theory, COLT 2019, 25-28 June 2019, Phoenix, AZ, USA*, pages 589–623, 2019. 3.1.2

[CG18] Yu Cheng and Rong Ge. Non-convex matrix completion against a semi-random adversary. In *Conference On Learning Theory (COLT)*, pages 1362–1394. https://arxiv.org/pdf/1803.10846.pdf, 2018. 1.1, 3.1

[Che21] Sinho Chewi. The entropic barrier is n-self-concordant. *arXiv preprint arXiv:2112.10947*, 2021. 5, 4.4.1, 4.4.3

[CKC83] Ruen-Wu Chen, Yoji Kajitani, and Shu-Park Chan. A graph-theoretic via minimization algorithm for two-layer printed circuit boards. *IEEE Transactions on Circuits and Systems*, 30(5):284–299, 1983. 1.1, 2.1

[CKP+22] Vincent Conitzer, Christian Kroer, Debmalya Panigrahi, Okke Schrijvers, Nicolas E Stier-Moses, Eric Sodomka, and Christopher A Wilkens. Pacing equilibrium in first price auction markets. *Management Science*, 2022. 8.6

[CKSSM22] Vincent Conitzer, Christian Kroer, Eric Sodomka, and Nicolas E Stier-Moses. Multiplicative pacing equilibria in auction markets. *Operations Research*, 70(2):963–989, 2022. 8.6

[CL11a] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm. 1.2, 4.1

[CL11b] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):1–27, 2011. 6.5

[Cla90] Frank H Clarke. *Optimization and nonsmooth analysis*. SIAM, 1990. 5.1

[CLM+15a] Michael B Cohen, Yin Tat Lee, Cameron Musco, Christopher Musco, Richard Peng, and Aaron Sidford. Uniform sampling for matrix approximation. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, pages 181–190, 2015. 1.3

[CLM+15b] Michael B. Cohen, Yin Tat Lee, Cameron Musco, Christopher Musco, Richard Peng, and Aaron Sidford. Uniform sampling for matrix approximation. In Tim Roughgarden, editor, *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS 2015, Rehovot, Israel, January 11-13, 2015*. ACM, 2015. 7.1.3

[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)*. https://arxiv.org/pdf/1810.07896.pdf, 2019. 3.1.2, 3.3.1, 3.3.1

[CMO23] Ashok Cutkosky, Harsh Mehta, and Francesco Orabona. Optimal stochastic non-smooth non-convex optimization through online-to-non-convex conversion. *arXiv preprint arXiv:2302.03775*, 2023. 5.2.1

[CP15] Michael B. Cohen and Richard Peng. Lp row sampling by lewis weights. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '15. Association for Computing Machinery, 2015. 1.3, 7.1.1, 7.1, 7.1, 7.1, 7.1, 7.1, 7.1.1, 5, 7.1, 7.1.2, 7.1.3, F.4

[CPRT22] Flavio Chierichetti, Alessandro Panconesi, Giuseppe Re, and Luca Trevisan. Spectral robustness for correlation clustering reconstruction in semi-adversarial models. In *AISTATS*, 2022. 1.1

[CRT06] E.J. Candes, J. Romberg, and T. Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52(2):489–509, 2006. 1.2

[CZPA09] Andrzej Cichocki, Rafal Zdunek, Anh Huy Phan, and Shun-ichi Amari. *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*. John Wiley & Sons, 2009. 1.3, 6.1

[Dan47] George B Dantzig. Maximization of a linear function of variables subject to linear inequalities. *Activity analysis of production and allocation*, 13:339–347, 1947. 3.1.2

[DAST08] S Damla Ahipasaoglu, Peng Sun, and Michael J Todd. Linear convergence of a modified frank–wolfe algorithm for computing minimum-volume enclosing ellipsoids. *Optimisation Methods and Software*, 23(1), 2008. 7.1.3

[DBLJ14a] A. Defazio, F. Bach, and S. Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, pages 1646–1654, 2014. 2.1

[DBLJ14b] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. *Advances in neural information processing systems*, 27, 2014. 1.2, 4.1

[DBW12] John C Duchi, Peter L Bartlett, and Martin J Wainwright. Randomized smoothing for stochastic optimization. *SIAM Journal on Optimization*, 22(2):674–701, 2012. 5.1.1

[DD19] Damek Davis and Dmitriy Drusvyatskiy. Stochastic model-based minimization of weakly convex functions. *SIAM Journal on Optimization*, 29(1):207–239, 2019. 5.1, 5.2.1

[DDKL20] Damek Davis, Dmitriy Drusvyatskiy, Sham Kakade, and Jason D Lee. Stochastic subgradient method converges on tame functions. *Foundations of computational mathematics*, 20(1):119–154, 2020. 1.2, 5.1

[DDL+22] Damek Davis, Dmitriy Drusvyatskiy, Yin Tat Lee, Swati Padmanabhan, and Guanghao Ye. A gradient sampling method with complexity guarantees for lipschitz functions in high and low dimensions. In *Advances in Neural Information Processing Systems*, 2022. 1.2

[DDM21] Monica Dessole, Marco Dell'Orto, and Fabio Marcuzzi. The lawson-hanson algorithm with deviation maximization: Finite convergence and sparse recovery. *arXiv preprint arXiv:2108.05345*, 2021. 6.1

[DDMP18] Damek Davis, Dmitriy Drusvyatskiy, Kellie J MacPhee, and Courtney Paquette. Subgradient methods for sharp weakly convex functions. *Journal of Optimization Theory and Applications*, 179(3):962–982, 2018. 5.1

[DFO20a] Jelena Diakonikolas, Maryam Fazel, and Lorenzo Orecchia. Fair packing and covering on a relative scale. *SIAM Journal on Optimization*, 30(4):3284–3314, 2020. 6.1, 6.1, 6.2

[DFO20b] Jelena Diakonikolas, Maryam Fazel, and Lorenzo Orecchia. Fair packing and covering on a relative scale. *SIAM J. Optim.*, 30(4), 2020. 7.1.3

[DH09] Nikhil R Devanur and Thomas P Hayes. The adwords problem: online keyword matching with budgeted bidders under random permutations. In *Proceedings of the 10th ACM conference on Electronic commerce*, pages 71–78, 2009. 8.1.3

[DJL+22] Sally Dong, Haotian Jiang, Yin Tat Lee, Swati Padmanabhan, and Guanghao Ye. Decomposable non-smooth convex optimization with nearly-linear gradient oracle complexity. In *Advances in Neural Information Processing Systems*, 2022. 1.2

[DJSW19] Nikhil R. Devanur, Kamal Jain, Balasubramanian Sivan, and Christopher A. Wilkens. Near optimal online algorithms and fast approximation algorithms for resource allocation problems. *J. ACM*, 66(1), 2019. 8.1.3

[DLPS22] Jelena Diakonikolas, Chenghui Li, Swati Padmanabhan, and Chaobing Song. A fast scale-invariant algorithm for non-negative least squares with non-negative data. In *Advances in Neural Information Processing Systems*, 2022. 1.3

[DLS18] David Durfee, Kevin A Lai, and Saurabh Sawlani. *ell*_1 regression using lewis weights preconditioning and stochastic gradient descent. In *Conference On Learning Theory*, 2018. 1.3, 7.1.3

[DLY21] Sally Dong, Yin Tat Lee, and Guanghao Ye. A nearly-linear time algorithm for linear programs with small treewidth: a multiscale representation of robust central path. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1784–1797, 2021. 4.1

[DMIMW12] Petros Drineas, Malik Magdon-Ismail, Michael W Mahoney, and David P Woodruff. Fast approximation of matrix coherence and statistical leverage. *The Journal of Machine Learning Research*, 13(1), 2012. 1.3, 7.1.3

[DMM06] Petros Drineas, Michael W Mahoney, and Shan Muthukrishnan. Sampling algorithms for l 2 regression and applications. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, 2006. 1.3, 7.1, 7.1.3

[DMMZ21] Yuan Deng, Jieming Mao, Vahab Mirrokni, and Song Zuo. Towards efficient auctions in an auto-bidding world. In *Proceedings of the Web Conference 2021*, pages 3965–3973, 2021. 8.1

[DO18] Jelena Diakonikolas and Lorenzo Orecchia. Alternating randomized block coordinate descent. In *International Conference on Machine Learning*, pages 1224–1232. PMLR, 2018. 6.1

[DO19] Jelena Diakonikolas and Lorenzo Orecchia. The approximate duality gap technique: A unified theory of first-order methods. *SIAM Journal on Optimization*, 29(1):660–689, 2019. 6.1, 6.3

[Eld13] Ronen Eldan. Thin shell implies spectral gap up to polylog via a stochastic localization scheme. *Geometric and Functional Analysis*, 23(2):532–569, 2013. B.1.1

[FGKS15] Roy Frostig, Rong Ge, Sham Kakade, and Aaron Sidford. Un-regularizing: approximate proximal point and faster stochastic algorithms for empirical risk minimization. In *International Conference on Machine Learning*, pages 2540–2548. PMLR, 2015. 4.1

[FHB04] M. Fazel, H. Hindi, and S. Boyd. Rank minimization and applications in system theory. In *Proceedings of the 2004 American Control Conference*, 2004. 1.1, 2.1

[FJPZ13] Alexander Fix, Thorsten Joachims, Sung Min Park, and Ramin Zabih. Structured learning of sum-of-submodular higher order energy functions. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3104–3111, 2013. 1.2, 4.1

[FK14] Simon Foucart and David Koslicki. Sparse recovery by means of nonnegative least squares. *IEEE Signal Processing Letters*, 21(4):498–502, 2014. 6.1

[FKM04] Abraham D Flaxman, Adam Tauman Kalai, and H Brendan McMahan. Online convex optimization in the bandit setting: gradient descent without a gradient. *arXiv preprint cs/0408007*, 2004. 5.1.1

[FLLZ18] Cong Fang, Chris Junchi Li, Zhouchen Lin, and Tong Zhang. Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. *Advances in Neural Information Processing Systems*, 31, 2018. 5.1

[FLPS22] Maryam Fazel, Yin Tat Lee, Swati Padmanabhan, and Aaron Sidford. Computing lewis weights to high precision. In *Symposium on Discrete Algorithms (SODA)*, 2022. 1.3

[FP07] Francisco Facchinei and Jong-Shi Pang. *Finite-dimensional variational inequalities and complementarity problems*. Springer Science & Business Media, 2007. 6.4

[FPS18] Zhe Feng, Chara Podimata, and Vasilis Syrgkanis. Learning to bid without knowing your value. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, page 505–522, 2018. 8.1.3

[FPW23] Zhe Feng, Swati Padmanabhan, and Di Wang. Online bidding algorithms for return-on-spend constrained advertisers. *TheWebConf*, 2023. 1.4

[FR15] Olivier Fercoq and Peter Richtárik. Accelerated, parallel, and proximal coordinate descent. *SIAM Journal on Optimization*, 25(4):1997–2023, 2015. 6.5

[FW56] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956. 2.1.1

[GH16a] Dan Garber and Elad Hazan. Sublinear time algorithms for approximate semidefinite programming. *Mathematical Programming*, 158(1-2):329–361, 2016. 1.1, 2.1.1, 2.1.1, A.1.2

[GH16b] Dan Garber and Elad Hazan. Sublinear time algorithms for approximate semidefinite programming. *Mathematical Programming*, 158(1-2):329–361, 2016. 3.1.2

[Gil14] Nicolas Gillis. The why and how of nonnegative matrix factorization. *Connections*, 12:2–2, 2014. 1.3, 6.1

[GJLM21] Negin Golrezaei, Patrick Jaillet, Jason Cheuk Nam Liang, and Vahab Mirrokni. Bidding and pricing in budget and roi constrained markets. *arXiv preprint arXiv:2107.07725*, 2021. 8.1, 8.1.3

[GL13] Saeed Ghadimi and Guanghui Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013. 5.1

[GLS81a] Martin Grötschel, László Lovász, and Alexander Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981. 1.1, 3.1

[GLS81b] Martin Grötschel, László Lovász, and Alexander Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981. 7.1.2

[GLS88] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization*. Springer, 1988. C.1.1, C.1.2, C.1.3

[GM12] Bernd Gärtner and Jiri Matousek. Approximation algorithms and semidefinite programming. Springer Science & Business Media, 2012. 1.1

[Gol77] AA Goldstein. Optimization of lipschitz continuous functions. *Mathematical Programming*, 13(1):14–22, 1977. 1.2, 5.1, 5.2, 5.2, 5.2.2

[Grü60] Branko Grünbaum. Partitions of mass-distributions and of convex bodies by hyperplanes. *Pacific Journal of Mathematics*, 10(4):1257–1261, 1960. 4.2.9, 8, 5.3.6

[GU18] François Le Gall and Florent Urrutia. Improved rectangular matrix multiplication using powers of the coppersmith-winograd tensor. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '18, 2018. 3.2.1, B.2.1, B.2.4

[Gül97] Osman Güler. On the self-concordance of the universal barrier function. *SIAM Journal on Optimization*, 7(2):295–303, 1997. 4.4.2, 4.4.5

[GV02] Jean-Louis Goffin and Jean-Philippe Vial. Convex nondifferentiable optimization: A survey focused on the analytic center cutting plane method. *Optimization methods and software*, 17(5):805–867, 2002. 1.1, 3.1

[GV16] Olivier Guédon and Roman Vershynin. Community detection in sparse networks via grothendieck's inequality. *Probability Theory and Related Fields*, 165(3-4):1025–1049, 2016. 1.1, 2.1

[GW95] Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995. 1.1, 1.1, 2.1, 3.1

[GYC+22] Yuan Gao, Kaiyu Yang, Yuanlong Chen, Min Liu, and Noureddine El Karoui. Bidding agent design in the linkedin ad marketplace. *arXiv preprint arXiv:2202.12472*, 2022. 8.1, 8.1.3

[Hal18] Georgina Hall. *Optimization over nonnegative and convex polynomials with and without semidefinite programming*. PhD thesis, Princeton University, 2018. 1.1

[Haz08] Elad Hazan. Sparse approximate solutions to semidefinite programs. In *Latin American symposium on theoretical informatics*, pages 306–316, 2008. 2.1.1

[HJ12] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, New York, NY, USA, 2nd edition, 2012. B.1.2, B.1.3

[HJST21] Baihe Huang, Shunhua Jiang, Zhao Song, and Runzhou Tao. Solving tall dense sdps in the current matrix multiplication time. *arXiv preprint arXiv:2101.08208*, 6, 2021. 1.1

[HL16a] Elad Hazan and Haipeng Luo. Variance-reduced and projection-free stochastic optimization. In *International Conference on Machine Learning*, pages 1263–1271. PMLR, 2016. 1.2, 4.1

[HL16b] Elad Hazan and Haipeng Luo. Variance-reduced and projection-free stochastic optimization. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, 2016. 2.1

[HRVW96] Christoph Helmberg, Franz Rendl, Robert J. Vanderbei, and Henry Wolkowicz. An interior-point method for semidefinite programming. *SIAM Journal on Optimization*, 6(2):342–361, 1996. 2.1.1, A.1.1

[HZF+20] Yanjun Han, Zhengyuan Zhou, Aaron Flores, Erik Ordentlich, and Tsachy Weissman. Learning to bid optimally and efficiently in adversarial first-price auctions. *CoRR*, abs/2007.04568, 2020. 8.1.3

[IFAB90] Takashi Isobe, Eric D Feigelson, Michael G Akritas, and Gutti Jogesh Babu. Linear regression in astronomy. *The astrophysical journal*, 364:104–113, 1990. 1.3, 6.1

[ISSS19] Nicole Immorlica, Karthik Abinav Sankararaman, Robert Schapire, and Aleksandrs Slivkins. Adversarial bandits with knapsacks. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 202–219. IEEE, 2019. 8.1.3

[Jag11] Martin Jaggi. *Sparse Convex Optimization Methods for Machine Learning*. PhD thesis, ETH Zurich, 2011. 1.1, 2.1

[JGN+17] Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M Kakade, and Michael I Jordan. How to escape saddle points efficiently. In *International Conference on Machine Learning*, pages 1724–1732. PMLR, 2017. 5.1

[JHD18] Ethan C Jackson, James Alexander Hughes, and Mark Daley. On the generalizability of linear and non-linear region of interest-based multivariate regression models for fmri data. In *2018 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, pages 1–8. IEEE, 2018. 1.3, 6.1

[JJUW11] Rahul Jain, Zhengfeng Ji, Sarvagya Upadhyay, and John Watrous. QIP = PSPACE. *Journal of the ACM (JACM)*, 58(6):1–27, 2011. 1.1, 3.1

[JKL+20] Haotian Jiang, Tarun Kathuria, Yin Tat Lee, Swati Padmanabhan, and Zhao Song. A faster interior point method for semidefinite programs. In *Proceedings of the 61st Annual IEEE Foundations of Computer Science, FOCS 2020, Virtual Conference, November 16-19, 2020*, 2020. 1.1

[JL84] William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984. 2.1, A.3.10

[JLL+20a] Arun Jambulapati, Yin Tat Lee, Jerry Li, Swati Padmanabhan, and Kevin Tian. Positive semidefinite programming: Mixed, parallel, and width-independent. *arXiv:2002.04830*, 2020. 1.1

[JLL+20b] Arun Jambulapati, Yin Tat Lee, Jerry Li, Swati Padmanabhan, and Kevin Tian. Positive semidefinite programming: Mixed, parallel, and width-independent. In *STOC*. https://arxiv.org/pdf/2002.04830.pdf, 2020. 3.1.2, B.1.1

[JLLV21] He Jia, Aditi Laddha, Yin Tat Lee, and Santosh Vempala. Reducing isotropy and volume to kls: an o*(n 3 ψ 2) volume algorithm. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 961–974, 2021. 4.1.2

[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 *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, 2020. 3.1, 3.2

[JLT20] Arun Jambulapati, Jerry Li, and Kevin Tian. Robust sub-gaussian principal component analysis and width-independent schatten packing. *Advances in Neural Information Processing Systems*, 33:15689–15701, 2020. 6.1

[JN08] Anatoli Juditsky and Arkadii S Nemirovski. Large deviations of vector-valued martingales in 2-smooth normed spaces. *arXiv preprint arXiv:0809.0813*, 2008. 2.2.2, 2.2.2, A.4.5

[Joh48] Fritz John. Extremum problems with inequalities as subsidiary conditions, studies and essays presented to r. courant on his 60th birthday, january 8, 1948, 1948. 7.1

[JP87] Michael S Jennis and Joyce L Peabody. Pulse oximetry: an alternative method for the assessment of oxygenation in newborn infants. *Pediatrics*, 79(4):524–528, 1987. 1.3, 6.1

[JST21] Fernando Granha Jeronimo, Shashank Srivastava, and Madhur Tulsiani. Near-linear time decoding of ta-shma's codes via splittable regularity. *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, 2021. 1.1

[JY11] Rahul Jain and Penghui Yao. A parallel approximation algorithm for positive semidefinite programming. In *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science (FOCS)*, 2011. 3.1.2

[JZ13a] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26, 2013. 1.2, 4.1, 4.1

[JZ13b] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pages 315–323, 2013. 2.1, 2.1.1

[Kar72] R. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972. 2.1

[Kar84] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing (STOC)*, pages 302–311, 1984. 1.1, 3.1, 3.1.2

[KB13] Kamil A Khan and Paul I Barton. Evaluating an element of the clarke generalized jacobian of a composite piecewise differentiable function. *ACM Transactions on Mathematical Software (TOMS)*, 39(4):1–28, 2013. 5.1

[Kha80] Leonid G Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72, 1980. 1.1, 3.1, 3.1, 3.2, 3.1.2, 3.1.2

[Kha96] Leonid G Khachiyan. Rounding of polytopes in the real number model of computation. *Mathematics of Operations Research*, 21(2), 1996. 7.1.3

[Kiw07] Krzysztof C Kiwiel. Convergence of the gradient sampling algorithm for nonsmooth nonconvex optimization. *SIAM Journal on Optimization*, 18(2):379–388, 2007. 1.2, 5.1

[KJ18] Richard Kueng and Peter Jung. Robust nonnegative sparse recovery and the nullspace property of 0/1 measurements. *IEEE Transactions on Information Theory*, 64(2):689–703, 2018. 6.1

[KKMO07] Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for max-cut and other 2-variable csps? *SIAM Journal on Computing*, 37(1):319–357, 2007. 4

[KL96] Philip Klein and Hsueh-I Lu. Efficient approximation algorithms for semidefinite programs arising from max cut and coloring. In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, 1996. 2.1.1, A.1.3

[Kla06] Boas Klartag. On convex perturbations with a bounded isotropic constant. *Geometric & Functional Analysis GAFA*, 16(6):1274–1290, 2006. 9

[KLS95a] R. Kannan, L. Lovász, and M. Simonovits. Isoperimetric problems for convex bodies and a localization lemma. *Discrete Comput. Geom.*, 13(3–4):541–559, Dec 1995. 5.3.7

[KLS95b] Ravi Kannan, László Lovász, and Miklós Simonovits. Isoperimetric problems for convex bodies and a localization lemma. *Discrete & Computational Geometry*, 13(3):541–559, 1995. 4.5.3

[KLS22] Tarun Kathuria, Yang P Liu, and Aaron Sidford. Unit capacity maxflow in almost $m^{4/3}$ time. *SIAM Journal on Computing*, pages FOCS20–175, 2022. 1.1

[KLT09] Pushmeet Kohli, Lubor Ladicky, and Philip H. S. Torr. Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision*, 82(3):302–324, 2009. 1.2, 4.1

[KM03] Kartik Krishnan and John E Mitchell. Properties of a cutting plane method for semidefinite programming. *submitted for publication*, 2003. 3.1, 3.2

[KMS94] David Karger, Rajeev Motwani, and Madhu Sudan. Approximate graph coloring by semidefinite programming. In *Proceedings 35th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 2–13. IEEE, 1994. 1.1, 3.1

[KN09] Ravi Kannan and Hariharan Narayanan. Random walks on polytopes and an affine interior point method for linear programming. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, New York, NY, USA, 2009. Association for Computing Machinery. 7.1.3

[KS21] Guy Kornowski and Ohad Shamir. Oracle complexity in nonsmooth nonconvex optimization. *Advances in Neural Information Processing Systems*, 34, 2021. 1.2, 5.1

[KS22] Guy Kornowski and Ohad Shamir. On the complexity of finding small subgradients in nonsmooth optimization. *arXiv preprint arXiv:2209.10346*, 2022. 5.2.1

[KSD13] Dongmin Kim, Suvrit Sra, and Inderjit S Dhillon. A non-monotonic method for large-scale non-negative least squares. *Optimization Methods and Software*, 28(5):1012–1039, 2013. 1.3, 6.1, 6.1, 6.5

[KSK13] Abhishek Kumar, Vikas Sindhwani, and Prabhanjan Kambadur. Fast conical hull algorithms for near-separable non-negative matrix factorization. In *International Conference on Machine Learning*, pages 231–239. PMLR, 2013. 1.3, 6.1

[KST09] Sham M. Kakade, Shai Shalev-Shwartz, and Ambuj Tewari. Applications of strong convexity–strong smoothness duality to learning with matrices. *CoRR*, abs/0910.0610, 2009. 2.2.2, 2.2.2, A.3.4, A.4.4

[KT10] Alex Kulesza and Ben Taskar. Structured determinantal point processes. *Advances in neural information processing systems*, 23, 2010. 1.2, 4.1

[KTE88] Leonid G Khachiyan, Sergei Pavlovich Tarasov, and I. I. Erlikh. The method of inscribed ellipsoids. In *Soviet Math. Dokl*, volume 37, pages 226–230, 1988. 3.1, 3.2, 3.1.2, 4.1

[KY05] Piyush Kumar and E Alper Yildirim. Minimum-volume enclosing ellipsoids and core sets. *Journal of Optimization Theory and applications*, 126(1), 2005. 7.1.3

[LCB+04] Gert RG Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine learning research*, 5(Jan):27–72, 2004. 1.1

[Lee16] Yin Tat Lee. *Faster Algorithms for Convex and Combinatorial Optimization*. PhD thesis, Massachusetts Institute of Technology, 2016. 1.1, 3.1.2, 7.1, 7.1, 7.1.2, F.4

[Lew78] D Lewis. Finite dimensional subspaces of *l_{p}*. *Studia Mathematica*, 63(2), 1978. 7.1.1, 2

[Lew95] Adrian S Lewis. The convex analysis of unitarily invariant matrix functions. *Journal of Convex Analysis*, 2(1):173–183, 1995. 2.2.2, 2.2.2, A.4.6

[LH95] Charles L. Lawson and Richard J. Hanson. *Solving least squares problems*, volume 15 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1995. Revised reprint of the 1974 original. 1.3, 6.1, 6.1

[LLV20] Aditi Laddha, Yin Tat Lee, and Santosh Vempala. Strong self-concordance and sampling. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, New York, NY, USA, 2020. Association for Computing Machinery. 7.1.3

[LLX14] Qihang Lin, Zhaosong Lu, and Lin Xiao. An accelerated proximal coordinate gradient method. *Advances in Neural Information Processing Systems*, 27:3059–3067, 2014. 6.1

[LMH15] Hongzhou Lin, Julien Mairal, and Zaid Harchaoui. A universal catalyst for first-order optimization. *Advances in neural information processing systems*, 28, 2015. 4.1

[LMP13] Mu Li, Gary L Miller, and Richard Peng. Iterative row sampling. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 127–136. IEEE, 2013. 1.3, 7.1.3

[LN93] Michael Luby and Noam Nisan. A parallel approximation algorithm for positive linear programming. In *Proc. ACM STOC'93*, 1993. 6.1

[LP20] Yin Tat Lee and Swati Padmanabhan. An $\widetilde\mathcalo(m/\varepsilonˆ3.5)$-cost algorithm for semidefinite programs with diagonal constraints. In Jacob D. Abernethy and Shivani Agarwal, editors, *Conference on Learning Theory, COLT 2020, 9-12 July 2020, Virtual Event [Graz, Austria]*, Proceedings of Machine Learning Research. PMLR, 2020. 1.1, 2.1, 3.1.2

[LS13] Yin Tat Lee and Aaron Sidford. Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems. In *2013 ieee 54th annual symposium on foundations of computer science*, pages 147–156. IEEE, 2013. 1.3, 6.1

[LS14] Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in õ(sqrt(rank)) iterations and faster algorithms for maximum flow. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, 2014. 1.1, 3.1.2, 7.1, 7.1, 7.1, 7.1, 7.1.4, 7.1, 7.1.2, 7.1.3, 7.1.5, 7.1.6, F.1, F.2.2

[LS15] Yin Tat Lee and Aaron Sidford. Efficient inverse maintenance and faster algorithms for linear programming. In *56th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 230–249. https://arxiv.org/pdf/1503.01752.pdf, 2015. 3.1.2

[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 *56th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1049–1065. https://arxiv.org/pdf/1508.04874.pdf, 2015. 1.1, 2.1.1, 3.1, 3.1.1, 3.1, 3.2, 3.1.1, 3.1.3, 3.1.2, 4.1, 4.1.1, 4.1.2, 7.1.2, A.1.1, 3.1.3, B.7, C.1.4, F.4

[LV07] László Lovász and Santosh Vempala. The geometry of logconcave functions and sampling algorithms. *Random Structures & Algorithms*, 30(3):307–358, 2007. A.4.20

[LV21] Yin Tat Lee and Santosh S Vempala. Tutorial on the robust interior point method. *arXiv preprint arXiv:2108.04734*, 2021. 4.1, 4.1.2

[LWYZ20] Yi Li, Ruosong Wang, Lin Yang, and Hanrui Zhang. Nearly linear row sampling algorithm for quantile regression. In *Proceedings of the 37th International Conference on Machine Learning*, 2020. 1.3, 7.1.3

[LY18] Yin Tat Lee and Man-Chung Yue. Universal barrier is n-self-concordant. *arXiv preprint arXiv:1809.03011*, 2018. 7.1.3

[LY21] Yin Tat Lee and Man-Chung Yue. Universal barrier is n-self-concordant. *Mathematics of Operations Research*, 46(3):1129–1148, 2021. 6, 4.4.2, 4.4.5

[Mai15] Julien Mairal. Incremental majorization-minimization optimization with application to large-scale machine learning. *SIAM Journal on Optimization*, 25(2):829–855, 2015. 1.2, 4.1

[MFLS17] Joe M Myre, Erich Frahm, David J Lilja, and Martin O Saar. TNT-NN: A fast active set method for solving large non-negative least squares problems. *Procedia Computer Science*, 108:755–764, 2017. 6.1

[MJY12] Mehrdad Mahdavi, Rong Jin, and Tianbao Yang. Trading regret for efficiency: online convex optimization with long term constraints. *The Journal of Machine Learning Research*, 13(1):2503–2528, 2012. 8.1.3

[MMM18] Szymon Majewski, Błażej Miasojedow, and Eric Moulines. Analysis of nonsmooth stochastic approximation: the differential inclusion approach. *arXiv preprint arXiv:1805.01916*, 2018. 1.2, 5.1

[MMMO17] Song Mei, Theodor Misiakiewicz, Andrea Montanari, and Roberto Imbuzeiro Oliveira. Solving sdps for synchronization and maxcut problems via the grothendieck inequality. In *Conference on Learning Theory, COLT 2017*, pages 1476–1515, 2017. 2.1.1

[MMSBVS08] Eduardo Martínez-Montes, José M Sánchez-Bornot, and Pedro A Valdés-Sosa. Penalized parafac analysis of spontaneous eeg recordings. *Statistica Sinica*, pages 1449–1464, 2008. 1.3, 6.1

[MR94] Olvi L. Mangasarian and J Ren. New improved error bounds for the linear complementarity problem. *Mathematical Programming*, 66(1):241–255, 1994. 6.4, E.1.4, E.1.10

[MRWZ16] Michael W Mahoney, Satish Rao, Di Wang, and Peng Zhang. Approximating the solution to mixed packing and covering LPs in parallel $\tilde{O}(\epsilon^{-3})$ time. In *Proc. ICALP'16*, 2016. 6.1

[MS16a] Andrea Montanari and Subhabrata Sen. Semidefinite programs on sparse random graphs and their application to community detection. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 814–827, 2016. 1.1, 2.1

[MS16b] Andrea Montanari and Subhabrata Sen. Semidefinite programs on sparse random graphs and their application to community detection. In *Proceedings of the Forty-eighth Annual ACM Symposium on Theory of Computing*, STOC '16, 2016. 2.1.1

[MSTX19] Vivek Madan, Mohit Singh, Uthaipon Tantipongpipat, and Weijun Xie. Combinatorial algorithms for optimal design. In *Proceedings of the Thirty-Second Conference on Learning Theory*, 2019. 7.1.3

[MSVV07] Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. Adwords and generalized online matching. *Journal of the ACM (JACM)*, 54(5):22–es, 2007. 8.1.3

[MSZ16] Jelena Marasevic, Clifford Stein, and Gil Zussman. A fast distributed stateless algorithm for *alpha*-fair packing problems. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, volume 55, 2016. 6.1, 7.1.3

[MTY09] Shie Mannor, John N Tsitsiklis, and Jia Yuan Yu. Online learning with sample path constraints. *Journal of Machine Learning Research*, 10(3), 2009. 8.1.3

[Mye81] Roger B Myerson. Optimal auction design. *Mathematics of operations research*, 6(1):58–73, 1981. 8.2

[MYJ13] Mehrdad Mahdavi, Tianbao Yang, and Rong Jin. Stochastic convex optimization with multiple objectives. *Advances in neural information processing systems*, 26, 2013. 8.1.3

[MZJ13] Mehrdad Mahdavi, Lijun Zhang, and Rong Jin. Mixed optimization for smooth functions. *Advances in neural information processing systems*, 26, 2013. 1.2, 4.1

[N+18] Yurii Nesterov et al. *Lectures on convex optimization*, volume 137. Springer, 2018. 6.1

[NCKP22] Thomas Nedelec, Clément Calauzènes, Noureddine El Karoui, and Vianney Perchet. Learning in repeated auctions. *Foundations and Trends® in Machine Learning*, 15(3):176–334, 2022. 8.1.3

[Nem92] Arkadi S Nemirovsky. Information-based complexity of linear operator equations. *Journal of Complexity*, 8(2):153–175, 1992. 6.1

[Nes83a] Yurii Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. In *Doklady AN SSSR (translated as Soviet Mathematics Doklady)*, volume 269, pages 543–547, 1983. 6.1

[Nes83b] Yurii E Nesterov. A method for solving the convex programming problem with convergence rate $O(1/k^2)$. In *Dokl. akad. nauk Sssr*, volume 269, pages 543–547, 1983. 4.1, 4.1, 4.1, 4.1.2

[Nes04] Yurii E. Nesterov. *Introductory Lectures on Convex Optimization - A Basic Course*, volume 87 of *Applied Optimization*. Springer, 2004. 4.1, 4.4.4, 4.5.2, 4.5.6

[Nes09] Yurii Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical programming*, 120(1):221–259, 2009. 2.1.2

[Nes12] Yu Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012. 6.1

[Nes13] Yu Nesterov. Gradient methods for minimizing composite functions. *Mathematical programming*, 140(1):125–161, 2013. 6.5

[NN89] Yurii Nesterov and Arkadi Nemirovski. Self-concordant functions and polynomial time methods in convex programming. preprint, central economic & mathematical institute, ussr acad. *Sci. Moscow, USSR*, 1989. 3.1, 3.2, 3.1.2, 4.1

[NN92] Yurii Nesterov and Arkadi Nemirovski. Conic formulation of a convex programming problem and duality. *Optimization Methods and Software*, 1(2):95–115, 1992. 1.1, 3.1, 3.1.1, 3.1, 3.2, 3.2, 3.2, 3.2, 3.2.1, 3.2.1, 3.2.1

[NN94] Yurii Nesterov and Arkadi Nemirovski. *Interior-point polynomial algorithms in convex programming*, volume 13. Siam, 1994. 1.1, 3.1.1, 3.1, 3.2, 3.2, 4.1.2, 4.2.2, 4, 6, 4.4.2, 4.4.5, 4, 7.1.3

[NS17] Yurii Nesterov and Sebastian U Stich. Efficiency of the accelerated coordinate descent method on structured optimization problems. *SIAM Journal on Optimization*, 27(1):110–123, 2017. 6.1

[NS21] Gali Noti and Vasilis Syrgkanis. Bid prediction in repeated auctions with learning. In *Proceedings of the Web Conference 2021*, WWW '21, page 3953–3964, New York, NY, USA, 2021. Association for Computing Machinery. 8.1.3

[Nur73] E. A. Nurminskii. The quasigradient method for the solving of the nonlinear programming problems. *Cybernetics*, 9(1):145–150, Jan 1973. 5.1

[NY83a] Arkadii Semenovich Nemirovsky and David Borisovich Yudin. *Problem complexity and method efficiency in optimization*. 1983. 6.1

[NY83b] A.S. Nemirovsky and D.B. Yudin. *Problem complexity and method efficiency in optimization*. A Wiley-Interscience Publication. John Wiley & Sons, Inc., New York, 1983. Translated from the Russian and with a preface by E. R. Dawson, Wiley-Interscience Series in Discrete Mathematics. 1.2

[Par00] Pablo A Parrilo. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. PhD thesis, California Institute of Technology, 2000. 1.1, 3.1

[PC99] Victor Y. Pan and Zhao Q. Chen. The complexity of the matrix eigenproblem. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, May 1-4, 1999, Atlanta, Georgia, USA*, pages 507–516, 1999. 2.1

[PST91] S. A. Plotkin, D. B. Shmoys, and E. Tardos. Fast approximation algorithms for fractional packing and covering problems. In *[1991] Proceedings 32nd Annual Symposium of Foundations of Computer Science*, 1991. 2.1.1

[PT12] Richard Peng and Kanat Tangwongsan. Faster and simpler width-independent parallel algorithms for positive semidefinite programming. In *Proceedings of the twenty-fourth annual ACM symposium on Parallelism in algorithms and architectures (SPAA)*, pages 101–108, 2012. 3.1.2

[Puk06] Friedrich Pukelsheim. *Optimal Design of Experiments (Classics in Applied Mathematics) (Classics in Applied Mathematics, 50)*. Society for Industrial and Applied Mathematics, USA, 2006. 7.1.3

[QR16] Zheng Qu and Peter Richtárik. Coordinate descent with arbitrary sampling i: Algorithms and complexity. *Optimization Methods and Software*, 31(5):829–857, 2016. 6.1

[Ren01a] James Renegar. *A mathematical view of interior-point methods in convex optimization*. SIAM, 2001. 4.2.2, 4.2.5, 4.2.6, 4.2.7, 4.2.8, 4.3.1

[Ren01b] James Renegar. *A Mathematical View of Interior-point Methods in Convex Optimization*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001. B.4.3, B.4.5, B.4.6

[RFP10] Benjamin Recht, Maryam Fazel, and Pablo A Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM review*, 52(3):471–501, 2010. 1.2

[RHS+16] Sashank J Reddi, Ahmed Hefny, Suvrit Sra, Barnabas Poczos, and Alex Smola. Stochastic variance reduction for nonconvex optimization. In *International conference on machine learning*, pages 314–323. PMLR, 2016. 5.1

[RM51] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951. 1.2, 4.1

[Roc70] R Tyrrell Rockafellar. *Convex Analysis*, volume 36. Princeton University Press, 1970. 4.2.1, 4.2.1, 4.2.2, 4.2.3, 4.2.4

[RRWN11] Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. *Advances in neural information processing systems*, 24, 2011. 4.1

[RS09] Prasad Raghavendra and David Steurer. How to round any csp. In *Proceedings of the 2009 50th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '09, 2009. 2.1.1

[RSB12] Nicolas Roux, Mark Schmidt, and Francis Bach. A stochastic gradient method with an exponential convergence rate for finite training sets. *Advances in neural information processing systems*, 25, 2012. 1.2, 4.1, 4.1

[RSL18] Aditi Raghunathan, Jacob Steinhardt, and Percy S Liang. Semidefinite relaxations for certifying robustness to adversarial examples. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 10877–10887. https://arxiv.org/pdf/1811.01057.pdf, 2018. 1.1

[RW09] R Tyrrell Rockafellar and Roger J-B Wets. *Variational analysis*, volume 317. Springer Science & Business Media, 2009. 3, 5.1

[SF04] Peng Sun and Robert M Freund. Computation of minimum-volume covering ellipsoids. *Operations Research*, 52(5), 2004. 7.1.3

[SH14] Martin Slawski and Matthias Hein. Non-negative least squares for high-dimensional linear models: consistency and sparse recovery without regularization, 2014. 6.1

[Sha20] Ohad Shamir. Can we find near-approximately-stationary points of nonsmooth nonconvex functions? *arXiv preprint arXiv:2002.11962*, 2020. 5.2.1

[Sho77] Naum Z Shor. Cut-off method with space extension in convex programming problems. *Cybernetics and systems analysis*, 13(1):94–96, 1977. 3.1, 3.2, 3.1.2

[Sid15] Aaron Daniel Sidford. *Iterative methods, combinatorial optimization, and linear programming beyond the universal barrier*. PhD thesis, Massachusetts Institute of Technology, 2015. 3.1.2

[SJC19] Yonatan Shadmi, Peter Jung, and Giuseppe Caire. Sparse non-negative recovery from biased subgaussian measurements using nnls. *arXiv preprint arXiv:1901.05727*, 2019. 6.1

[SK10] Peter Stobbe and Andreas Krause. Efficient minimization of decomposable submodular functions. *Advances in Neural Information Processing Systems*, 23, 2010. 4.1

[SKR85] Naum Z. Shor, Krzysztof C Kiwiel, and Andrzej Ruszcayński. Minimization methods for non-differentiable functions, 1985. 5.1

[SLRB17a] Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1):83–112, 2017. 1.2, 4.1

[SLRB17b] Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1-2):83–112, 2017. 2.1

[SS05] Nathan Srebro and Adi Shraibman. Rank, trace-norm and max-norm. In *International Conference on Computational Learning Theory*, pages 545–560, 2005. 1.1, 2.1

[SS11] Amit Singer and Yoel Shkolnisky. Three-dimensional structure determination from common lines in cryo-em by eigenvectors and semidefinite programming. *SIAM journal on imaging sciences*, 4(2):543–572, 2011. 1.1, 2.1

[SSZ13a] Shai Shalev-Shwartz and Tong Zhang. Accelerated mini-batch stochastic dual coordinate ascent. *Advances in Neural Information Processing Systems*, 26, 2013. 4.1

[SSZ13b] Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14(2), 2013. 1.2, 4.1, 4.1

[SSZ13c] Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14(Feb):567–599, 2013. 2.1

[SV+14] Sushant Sachdeva, Nisheeth K Vishnoi, et al. Faster algorithms via approximation theory. *Foundations and Trends® in Theoretical Computer Science*, 9(2):125–210, 2014. 2.1, A.4.2, A.4.14, A.4.15, A.4.16

[SW18] Christian Sohler and David P Woodruff. Strong coresets for k-median and subspace approximation: Goodbye dimension. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, 2018. 1.3, 7.1.3

[SWD21] Chaobing Song, Stephen J Wright, and Jelena Diakonikolas. Variance reduction via primal-dual accelerated dual averaging for nonsmooth convex finite-sums. In *International Conference on Machine Learning*, 2021. 6.1, 6.2, 6.3

[SX20] Mohit Singh and Weijun Xie. Approximation algorithms for d-optimal design. *Mathematics of Operations Research*, 45(4), 2020. 7.1.3

[Tib96] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996. 6.1

[Tod16] Michael J. Todd. *Minimum volume ellipsoids - theory and algorithms*, volume 23 of *MOS-SIAM Series on Optimization*. SIAM, 2016. 7.1, 7.1, 7.1.4, 7.1.3

[TZS22] Lai Tian, Kaiwen Zhou, and Anthony Man-Cho So. On the finite-time complexity and practical computation of approximate stationarity concepts of lipschitz functions. In *International Conference on Machine Learning*, pages 21360–21379. PMLR, 2022. 5.1.1

[UT19] Madeleine Udell and Alex Townsend. Why are big data matrices approximately low rank? *SIAM Journal on Mathematics of Data Science*, 1(1):144–160, 2019. 1.2

[Vai87] Pravin M Vaidya. An algorithm for linear programming which requires $o(((m+n)n^2 + (m+n)^{1.5}n)l)$ arithmetic operations. In *28th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 1987. 3.1.2

[Vai89a] Pravin M Vaidya. A new algorithm for minimizing convex functions over convex sets. In *30th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 338–343, 1989. 3.1, 3.2, 3.1.2, 3.2, 4.1, 7.1, F.2.1

[Vai89b] Pravin M Vaidya. Speeding-up linear programming using fast matrix multiplication. In *30th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 332–337. IEEE, 1989. 3.1.2

[VB96] Lieven Vandenberghe and Stephen P. Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, 1996. 3.1

[VBK04] Mark H Van Benthem and Michael R Keenan. Fast algorithm for the solution of large-scale non-negativity-constrained least squares problems. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 18(10):441–450, 2004. 6.1

[VBK20] Nate Veldt, Austin R Benson, and Jon Kleinberg. Minimizing localized ratio cut objectives in hypergraphs. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1708–1718, 2020. 4.1

[vdB21] Jan van den Brand. *Dynamic Matrix Algorithms and Applications in Convex and Combinatorial Optimization*. PhD thesis, KTH Royal Institute of Technology, 2021. 1.1

[VDBLL+21] Jan Van Den Brand, Yin Tat Lee, Yang P Liu, Thatchaphol Saranurak, Aaron Sidford, Zhao Song, and Di Wang. Minimum cost flows, mdps, and ℓ_1-regression in nearly linear time for dense instances. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 859–869, 2021. 1.1

[VKR09] Sara Vicente, Vladimir Kolmogorov, and Carsten Rother. Joint optimization of segmentation and appearance models. In *2009 IEEE 12th international conference on computer vision*, pages 755–762. IEEE, 2009. 1.2, 4.1

[Wan17] Di Wang. *Fast Approximation Algorithms for Positive Linear Programs*. University of California, Berkeley, 2017. 1.3, 1.4, 6.1

[WdM15] Irène Waldspurger, Alexandre d'Aspremont, and Stéphane Mallat. Phase recovery, maxcut and complex semidefinite programming. *Mathematical Programming*, 2015. 1.1, 2.1

[Wil67] R. M. Wilcox. Exponential Operators and Parameter Differentiation in Quantum Physics. *Journal of Mathematical Physics*, 1967. 2.1.2

[WJC13] Jun Wang, Tony Jebara, and Shih-Fu Chang. Semi-supervised learning using greedy max-cut. *Journal of Machine Learning Research*, 14(Mar):771–800, 2013. 1.1, 2.1

[Woj96] Przemyslaw Wojtaszczyk. *Banach spaces for analysts*. Cambridge University Press, 1996. 7.1

[Woo49] Max A Woodbury. The stability of out-input matrices. *Chicago, IL*, 9, 1949. B.1.4

[Woo50] Max A Woodbury. Inverting modified matrices. 1950. B.1.4

[WPR16] Jonathan Weed, Vianney Perchet, and Philippe Rigollet. Online learning in repeated auctions. In *Conference on Learning Theory*, pages 1562–1583. PMLR, 2016. 8.1.3

[WPTP88] Michael W Wukitsch, Michael T Petterson, David R Tobler, and Jonas A Pologe. Pulse oximetry: analysis of theory, technology, and practice. *Journal of clinical monitoring*, 4(4):290–301, 1988. 1.3, 6.1

[WS16] Blake E Woodworth and Nati Srebro. Tight complexity bounds for optimizing composite objectives. *Advances in neural information processing systems*, 29, 2016. 4.1, 4.1.2

[WXT11] Meng Wang, Weiyu Xu, and Ao Tang. A unique "nonnegative" solution to an underdetermined system: From vectors to matrices. *IEEE Transactions on Signal Processing*, 59(3):1007–1016, 2011. 6.1

[YN76] David B Yudin and Arkadi S Nemirovski. Evaluation of the information complexity of mathematical programming problems. *Ekonomika i Matematicheskie Metody*, 12:128–142, 1976. 3.1, 3.2, 3.1.2

[YN20] Hao Yu and Michael J Neely. A low complexity algorithm with $o(\sqrt{T})$ regret and $o(1)$ constraint violations for online convex optimization with long term constraints. *Journal of Machine Learning Research*, 21(1):1–24, 2020. 8.1.3

[YNW17] Hao Yu, Michael Neely, and Xiaohan Wei. Online convex optimization with stochastic constraints. *Advances in Neural Information Processing Systems*, 30, 2017. 8.1.3

[You01] Neal Young. Sequential and parallel algorithms for mixed packing and covering. In *Proc. IEEE FOCS'01*, 2001. 6.1

[YTF+19a] Alp Yurtsever, Joel A. Tropp, Olivier Fercoq, Madeleine Udell, and Volkan Cevher. Scalable semidefinite programming, 2019. 1.1, 2.1.1, A.1.3

[YTF+19b] Alp Yurtsever, Joel A. Tropp, Olivier Fercoq, Madeleine Udell, and Volkan Cevher. Scalable semidefinite programming, 2019. 3.1.2

[ZF20] Renbo Zhao and Robert M Freund. Analysis of the frank-wolfe method for logarithmically-homogeneous barriers, with an extension. *arXiv preprint arXiv:2010.08999*, 2020. 7.1.3

[Zha04] Tong Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the twenty-first international conference on Machine learning*, page 116, 2004. 1.2, 4.1

[ZL15] Yuchen Zhang and Xiao Lin. Stochastic primal-dual coordinate method for regularized empirical risk minimization. In *International Conference on Machine Learning*, pages 353–361. PMLR, 2015. 4.1

[ZLSJ20] Jingzhao Zhang, Hongzhou Lin, Suvrit Sra, and Ali Jadbabaie. On complexity of finding stationary points of nonsmooth nonconvex functions. *International Conference on Machine Learning*, 2020. 1.2, 5.1, 5.1, 5.1.1, 5.1.1, 5.2, 5.2, 5.2.2, 1, 5.2, 5.2, 5.2.3, 5.2, 5.2.1, 5.2.1, 5.2.1, 5.2.1

[ZLY22] Manru Zong, Yin Tat Lee, and Man-Chung Yue. Short-step methods are not strongly polynomial-time. *arXiv preprint arXiv:2201.02768*, 2022. 4.5.7

[ZXG18] Dongruo Zhou, Pan Xu, and Quanquan Gu. Stochastic nested variance reduction for nonconvex optimization. *Advances in Neural Information Processing Systems*, 31, 2018. 5.1

Appendices

Appendix A

Appendix for Chapter 2

This chapter contains details and proofs from Chapter 2.

A.1 Previous Results

In this section, we show how to derive the runtimes we mentioned in Section 2.1.1, of some previous works in solving (2.1.1).

A.1.1 Cutting plane and interior point methods

In this subsection, we derive the complexities we claimed are attained by some cutting plane and interior point methods in solving (2.1.1).

Result of Helmberg, Rendl, Vanderbei, and Wolkowikz. The interior point method of [HRVW96] first writes a dual of the problem and then runs an interior point method using log-barrier. The purpose of running IPM on the dualized problem is to reduce the problem dimension from $O(n^2)$ to n. The iteration complexity, by self concordance of log-barrier, is $O(\sqrt{n})$. Each iteration performs operations that have cost equal to that of a matrix multiplication of two $n \times n$ matrices; this gives a per iteration cost of n^ω, thus giving us a total cost of $O(n^{\omega+1/2})$, as we claimed in Section 2.1.1.

Result of Lee, Sidford, and Wong. The cutting plane method of [LSW15] solves an SDP of size $m \times m$ with n constraints in time $O(n(n^2 + m^\omega + S))$, where S is the number of nonzero entries in the cost matrix. Translated to our setting (and being careful with the differences in definitions of n and m), this implies a run time of $O(n(n^\omega + m))$, since we denote m to be the number of nonzero entries of the cost matrix, and the number of constraints equals the number of rows of our matrix variable.

A.1.2 Saddle point methods

Result of Garber and Hazan. Per Theorem 1 of [GH16a], their paper solves, up to additive ε accuracy, an instance of (2.1.5) in time at least $O(S\varepsilon^{-2.5})$, where S is the number of nonzero entries among all input matrices of the problem. Note that by scaling C by $\lambda = \sum_{i,j} |C_{ij}|$, we ensure, due to the Gershgorin Circle Theorem, that the spectral norm of the scaled cost matrix is bounded by one, as required by their theorem.

Result of Carmon, Duchi, Sidford, and Tian. Per Section 4 of [CDST19a], the cost of obtaining an ε-additive accurate solution to (2.1.5) is $O(\text{mv(A)}\omega^{2.5}\varepsilon^{-2.5})$, where they use mv(A) to denote the cost of a matrix vector product using any of the input matrices and ω to denote the maximum spectral norm of the input matrices. Translating to our notation, this becomes $O(m\varepsilon^{-2.5})$.

A.1.3 Low rank methods

Result of Klein and Lu. The result of [KL96] for solving (2.1.1) is stated in their Lemma 4. They have $O(n\varepsilon^{-2})$ iterations from the framework of Plotkin, Shmoys, and Tardos, and each iteration performs a power method that costs $O(m\varepsilon^{-1})$.

Result of Yurtsever et al. Per Theorem 6.3 of [YTF+19a], the cost is at least $O(\varepsilon^{-2}(d + Rn) + \varepsilon^{-3}n)$, where we omit polylogarithmic factors. The bound on distance of iterates of the algorithm from the optimal set is given in Theorem 6.2.

A.2 Analysis Common to Both Algorithms

In this section we provide proofs for two results: the first is that a solution to the reformulated problem (2.1.2) is indeed ε close to that of the original; the second is the convergence guarantee of approximate lazy mirror descent, the framework for both the Arora-Kale algorithm as well as ours.

Algorithm A.2.1 Approximate lazy mirror descent

Input: Objective function $f : \mathcal{X} \to \mathbb{R}$, accuracy parameter ε.
Parameters: Mirror map $\Phi : \mathcal{D} \to \mathbb{R}$, norm $\|\cdot\|$, step size η, iteration T, error bound δ.

1: Initialize: $x^{(1)} \in \operatorname{argmin}_{x \in \mathcal{X} \cap \mathcal{D}} \Phi(x)$, $\widetilde{x}^{(1)} = x^{(1)}$, and $z^{(1)}$ satisfying $\nabla\Phi(z^{(1)}) = 0$.
2: **for** $t = 1 \to T$ **do**
3: $\quad \nabla\Phi(z^{(t+1)}) \leftarrow \nabla\Phi(z^{(t)}) - \eta \nabla f(\widetilde{x}^{(t)})$ ▷ Lazy gradient update
4: $\quad$ Find $\widetilde{x}^{(t+1)}$ such that $\mathbb{E}\|\widetilde{x}^{(t+1)} - x^{(t+1)}\| \le \delta$, where $x^{(t+1)} \in \operatorname{argmin}_{x \in \mathcal{X} \cap \mathcal{D}} \mathcal{D}_\Phi(x, z^{(t+1)})$ ▷ Approximate projection
5: **end for**
6: For $t^* \overset{\text{unif.}}{\sim} \{1, 2, \ldots, T\}$, return $\widetilde{x}^{(t^*)}$.

A.2.1 From the Reformulated to the Original SDP

Our claim of reformulating (2.1.1) as (2.1.2) works because once we have a solution X for the latter, we can apply the following result to obtain a matrix $\widehat{X}$ which satisfies all the required constraints of (2.1.1), and at which the objective value in (2.1.1) is better than that at X in (2.1.2).

Lemma 2.1.4. *Given $C \in \mathbb{R}^{n \times n}$ and $0 \preceq X$, let $\rho \in \mathbb{R}^n$ with $\rho_i = \sum_{j=1}^n |C_{ij}|$; diagonal matrix S with $S_{ii} = \min(1/\sqrt{\rho_i}, 1/\sqrt{X_{ii}})$ for $i \in [n]$; $\widehat{X} = SXS$; $\widehat{C} = \mathbf{diag}(1/\sqrt{\rho}) C \mathbf{diag}(1/\sqrt{\rho})$. Then, $\widehat{X} \succeq 0$, $\widehat{X}_{ii} \le 1$ for all $i \in [n]$, and $\widehat{C} \bullet X - \sum_{i=1}^n (X_{ii} - \rho_i)^+ \le C \bullet \widehat{X}$.*

Proof. We first prove the positive semidefiniteness. Observe that since $\widehat{X}$ and X are similar matrices, $X \succeq 0$ implies $\widehat{X} \succeq 0$ as well. Next, we define a matrix Y as $Y_{ij} = \frac{X_{ij}}{\sqrt{\rho_i}\sqrt{\rho_j}}$. Without loss of generality, assume $Y_{11} \ge Y_{22} \ge \ldots \ge Y_{nn}$. We also define a diagonal matrix, $\widehat{D}$ as $\widehat{D}_{ii} = \min(1, 1/\sqrt{Y_{ii}})$. If $Y_{ii} \ge 1$, then $\widehat{X}_{ii} = \frac{\rho_i Y_{ii}}{\sqrt{\rho_i Y_{ii}}\sqrt{\rho_i Y_{ii}}} = 1$; otherwise, $\widehat{X}_{ii} = Y_{ii}$. This proves that $\widehat{X}_{ii} \le 1$ for all $1 \le i \le n$, which is precisely the claim bounding every diagonal entry. We now prove the claim about the

objective value. By definition of $\widehat{D}$, $\widehat{X}$ and Y, we have $\widehat{X} = \widehat{D} \cdot Y \cdot \widehat{D}$. Therefore we get

$$\begin{aligned} C \bullet (\widehat{X} - Y) - \sum_{i=1}^{n} C_{ii} Y_{ii} (\widehat{D}_{ii}^2 - 1) &= \sum_{i=1}^{n} \sum_{j \neq i} C_{ij} Y_{ij} (\widehat{D}_{ii} \widehat{D}_{jj} - 1) \\ &= 2 \sum_{i=1}^{n} \sum_{i<j} C_{ij} Y_{ij} (\widehat{D}_{ii} \widehat{D}_{jj} - 1). \end{aligned}$$

The definition of $\widehat{D}$ and the ordering assumption on $\{Y_{ii}\}$ imply $0 < \widehat{D}_{11} \leq \widehat{D}_{22} \leq \ldots \leq \widehat{D}_{nn} \leq 1$, which in turn means $\widehat{D}_{ii}\widehat{D}_{jj} \geq \widehat{D}_{ii}^2$. Further, since $X \succeq 0$ and $Y = \mathbf{diag}(()^1/\sqrt{\rho}) \cdot X \cdot \mathbf{diag}(()^1/\sqrt{\rho})$, we have $Y \succeq 0$. Therefore $Y_{ii}Y_{jj} \geq Y_{ij}Y_{ji}$. By symmetry of Y and the assumed ordering of $\{Y_{ii}\}_1^n$, this can be simplified to $Y_{ii} \geq \left|Y_{ij}\right|$ for $i < j$. These two facts simplify the above to

$$\begin{aligned} C \bullet (\widehat{X} - Y) - \sum_{i=1}^{n} C_{ii} Y_{ii} (\widehat{D}_{ii}^2 - 1) &\geq 2 \sum_{i=1}^{n} \sum_{i<j} \left|C_{ij}\right| \left|Y_{ij}\right| (\widehat{D}_{ii}^2 - 1) \\ &\geq 2 \sum_{i=1}^{n} \sum_{i<j} \left|C_{ij}\right| Y_{ii} (\widehat{D}_{ii}^2 - 1) \end{aligned}$$

Finally, since $\widehat{D}_{ii} \leq 1$, we have $\widehat{D}_{ii}^2 - 1 \leq 0$. Rearranging the terms in the last inequality, we get

$$\begin{aligned} C \bullet (\widehat{X} - Y) &\geq \sum_{i=1}^{n} C_{ii} Y_{ii} (\widehat{D}_{ii}^2 - 1) + \sum_{i=1}^{n} Y_{ii} (\widehat{D}_{ii}^2 - 1) (\sum_{j>i} \left|C_{ij}\right| + \sum_{j<i} \left|C_{ij}\right|) \\ &= \sum_{i=1}^{n} Y_{ii} (\widehat{D}_{ii}^2 - 1) \underbrace{\left(C_{ii} + \sum_{i>j} \left|C_{ij}\right| + \sum_{i<j} \left|C_{ij}\right| \right)}_{\leq \rho_i} \\ &\geq \sum_{i=1}^{n} Y_{ii} \rho_i (\widehat{D}_{ii}^2 - 1) \\ &= -\sum_{i=1}^{n} \rho_i \left(Y_{ii} - 1\right)^+ \end{aligned}$$

where we used $\widehat{D}_{ii} = \min(1, 1/\sqrt{Y_{ii}})$ in the last step. Rearranging the terms in the last inequality gives

$$C \bullet \widehat{X} \geq C \bullet Y - \sum_{i=1}^{n} \rho_i \left(Y_{ii} - 1\right)^+ = \widehat{C} \bullet X - \sum_{i=1}^{n} (X_{ii} - \rho_i)^+,$$

where the last step is by definition of matrix Y. □

A.2.2 Analysis of Approximate Lazy Mirror Descent

We now derive the convergence bound of approximate lazy mirror descent. The proof closely follows that of Theorem 4.3 in Bubeck's monograph [B+15].

Theorem 2.1.3 (Convergence of Lazy Mirror Descent)**.** *Fix a norm $\|\cdot\|$. Given an α-strongly convex*

mirror map $\Phi : \mathcal{D} \to \mathbb{R}$ *and a convex, G-Lipschitz objective* $f : \mathcal{X} \to \mathbb{R}$, *run Algorithm A.2.1 with step size* η *and* $\mathbb{E}\|x^{(t)} - \widetilde{x}^{(t)}\| \leq \delta$. *Let* $D \stackrel{def}{=} \sup_{x \in \mathcal{X} \cap \mathcal{D}} \Phi(x) - \inf_{x \in \mathcal{X} \cap \mathcal{D}} \Phi(x)$ *and* $x^* = \arg\min_{\mathcal{X}} f(x)$. *Then, Algorithm A.2.1, after T iterations, returns* $\widetilde{x}^{t^*}$, *satisfying*

$$\mathbb{E} f(\widetilde{x}^{(t^*)}) - f(x^*) \leq \frac{D}{T\eta} + \frac{2\eta G^2}{\alpha} + \delta G. \tag{2.1.6}$$

Proof. By convexity of f,

$$\sum_{t=1}^{T} (f(\widetilde{x}^{(t)}) - f(x)) \leq \sum_{t=1}^{T} \left\langle \nabla f(\widetilde{x}^{(t)}), \widetilde{x}^{(t)} - x \right\rangle = \underbrace{\sum_{t=1}^{T} \left\langle \nabla f(\widetilde{x}^{(t)}), \widetilde{x}^{(t)} - x^{(t)} \right\rangle}_{\text{(A)}} + \underbrace{\sum_{t=1}^{T} \left\langle \nabla f(\widetilde{x}^{(t)}), x^{(t)} - x \right\rangle}_{\text{(B)}}. \tag{A.2.1}$$

The term (A) can be bounded by Cauchy-Schwarz inequality and the invariant $\mathbb{E}\|x^{(t)} - \widetilde{x}^{(t)}\| \leq \delta$:

$$\text{(A)} \leq \sum_{t=1}^{T} \|\Delta^{(t)}\| \|\nabla f\left(\widetilde{x}^{(t)}\right)\|_* \leq \delta G T. \tag{A.2.2}$$

Next, recall that Algorithm A.2.1 initializes $x^{(1)} \in \operatorname{argmin}_{\mathcal{X} \cap \mathcal{D}} \Phi(x)$ and $z^{(1)}$ satisfying $\nabla \Phi(z^{(1)}) = 0$, and repeats the following two steps:

$$\begin{aligned} \nabla \Phi(z^{(t)}) &= \nabla \Phi(z^{(t-1)}) - \eta \nabla f(x^{(t)}) \\ x^{(t)} &= \operatorname{argmin}_{\mathcal{X} \cap \mathcal{D}} D_\Phi(x, z^{(t)}). \end{aligned}$$

Now consider the potential function $\widetilde{\Psi}_t(x) \stackrel{\text{def}}{=} \Phi(x) + \eta \left\langle x, \sum_{s=1}^{t} \nabla f(\widetilde{x}^{(s)}) \right\rangle$. Applying the recursive definition of the gradient step, we can express $x^{(t+1)} = \arg \min_{x \in \mathcal{X} \cap \mathcal{D}} \widetilde{\Psi}_t(x)$. Since Φ is α-strongly convex, so is the potential function Ψ_t. We can express these two statements as follows:

$$\begin{aligned} \widetilde{\Psi}_t(x^{(t+1)}) - \widetilde{\Psi}_t(x^{(t)}) &\leq \underbrace{\left\langle \nabla \widetilde{\Psi}_t(x^{(t+1)}), x^{(t+1)} - x^{(t)} \right\rangle}_{\leq 0, \text{ by optimality of } x^{(t+1)}} - \tfrac{\alpha}{2} \|x^{(t+1)} - x^{(t)}\|^2 \\ &\leq -\tfrac{\alpha}{2} \|x^{(t+1)} - x^{(t)}\|^2. \end{aligned} \tag{A.2.3}$$

We can also write a lower bound for the left hand side of Inequality A.2.3 by evaluating the potential function $\widetilde{\Psi}_t$ at points $x^{(t+1)}$ and $x^{(t)}$:

$$\begin{aligned} \widetilde{\Psi}_t(x^{(t+1)}) - \widetilde{\Psi}_t(x^{(t)}) &= \Phi\left(x^{(t+1)}\right) + \eta \sum_{s=1}^{t} \left\langle \nabla f(\widetilde{x}^{(s)}), x^{(t+1)} \right\rangle - \Phi(x^{(t)}) - \eta \sum_{s=1}^{t} \left\langle \nabla f(\widetilde{x}^{(s)}), x^{(t)} \right\rangle \\ &= \underbrace{\widetilde{\Psi}_{t-1}(x^{(t+1)}) - \widetilde{\Psi}_{t-1}(x^{(t)})}_{\geq 0, \text{ since } x^{(t)} \text{ minimizes } \widetilde{\Psi}_{t-1}(x)} + \eta \left\langle \nabla f(\widetilde{x}^{(t)}), x^{(t+1)} - x^{(t)} \right\rangle \\ &\geq \eta \left\langle \nabla f(\widetilde{x}^{(t)}), x^{(t+1)} - x^{(t)} \right\rangle. \end{aligned} \tag{A.2.4}$$

Reverse and chain Inequality A.2.3 and Inequality A.2.4, and apply Cauchy-Schwarz inequality to

get

$$\frac{\alpha}{2}\|x^{(t+1)} - x^{(t)}\|^2 \leq \eta \left\langle \nabla f(\widetilde{x}^{(t)}), x^{(t)} - x^{(t+1)} \right\rangle \leq \eta G \|x^{(t)} - x^{(t+1)}\|. \tag{A.2.5}$$

This shows that

$$\|x^{(t)} - x^{(t+1)}\| \leq \frac{2\eta G}{\alpha}, \tag{A.2.6}$$

and applying this to the second part of Inequality A.2.5 gives

$$\left\langle \nabla f(\widetilde{x}^{(t)}), x^{(t)} - x^{(t+1)} \right\rangle \leq \frac{2\eta G^2}{\alpha}. \tag{A.2.7}$$

We now claim

$$\sum_{t=1}^{T} \left\langle \nabla f(\widetilde{x}^{(t)}), x^{(t)} - x \right\rangle \leq \sum_{t=1}^{T} \left\langle \nabla f(\widetilde{x}^{(t)}), x^{(t)} - x^{(t+1)} \right\rangle + \tfrac{1}{\eta}(\Phi(x) - \Phi(x^{(1)})). \tag{A.2.8}$$

Note that this claim immediately gives the desired error bound; this can be seen as follows: the left-hand side is exactly the term (2) in Inequality A.2.1; the first term of the right-hand side is bounded in Inequality A.2.7, and the second one is bounded by the definition of set size D. Therefore Inequality A.2.8 simplifies to

$$\text{(B)} \leq \frac{2\eta G^2 T}{\alpha} + \frac{D}{\eta}. \tag{A.2.9}$$

Combine Inequality A.2.9 and Inequality A.2.2 with Inequality A.2.1, apply Jensen's inequality, and the fact that t^* is picked uniformly at random from $\{1, 2, \ldots, T\}$, to get the desired error bound. We now prove Inequality A.2.8. First, we rewrite it as

$$\sum_{t=1}^{T} \left\langle \nabla f(\widetilde{x}^{(t)}), x^{(t+1)} \right\rangle + \frac{\Phi(x^{(1)})}{\eta} \leq \sum_{t=1}^{T} \left\langle \nabla f(\widetilde{x}^{(t)}), x \right\rangle + \frac{\Phi(x)}{\eta}.$$

The claim is true for $T = 0$ for all $x \in \mathcal{X}$, by the choice of $x^{(1)}$. Assume it holds for all $x \in \mathcal{X}$ at time

$T = t' - 1$. Therefore in particular, it holds at the point $x = x^{(t'+1)}$. This implies

$$\begin{aligned}
\sum_{t=1}^{t'} \left\langle \nabla f(\widetilde{x}^{(t)}), x^{(t+1)} \right\rangle + \frac{\Phi(x^{(1)})}{\eta} &= \left\langle \nabla f(\widetilde{x}^{(t')}), x^{(t'+1)} \right\rangle + \underbrace{\sum_{t=1}^{t'-1} \left\langle \nabla f(\widetilde{x}^{(t)}), x^{(t+1)} \right\rangle + \frac{\Phi(x^{(1)})}{\eta}}_{\text{Apply induction hypothesis at } x^{(t'+1)}} \\
&\leq \left\langle \nabla f(\widetilde{x}^{(t')}), x^{(t'+1)} \right\rangle + \sum_{t=1}^{t'-1} \left\langle \nabla f(\widetilde{x}^{(t)}), x^{(t'+1)} \right\rangle + \frac{\Phi(x^{(t'+1)})}{\eta} \\
&= \sum_{t=1}^{t'} \left\langle \nabla f(\widetilde{x}^{(t)}), x^{(t'+1)} \right\rangle + \frac{\Phi\left(x^{(t'+1)}\right)}{\eta} \\
&= \frac{1}{\eta} \widetilde{\Psi}_{t'} \left(x^{(t'+1)}\right) \\
&\leq \frac{1}{\eta} \widetilde{\Psi}_{t'}(x) \\
&= \sum_{t=1}^{t'} \left\langle \nabla f\left(\widetilde{x}^{(t)}\right), x \right\rangle + \frac{\Phi(x)}{\eta},
\end{aligned}$$

where the last inequality is by optimality of $x^{(t'+1)}$ in minimizing $\widetilde{\Psi}_{t'}$. This completes the induction, and therefore proves Inequality A.2.8, thus completing the proof of the error bound. □

A.3 Analysis of the Arora-Kale Algorithm

In this section, we display Algorithm A.3.1 in the approximate mirror descent framework and provide its analysis. In Section A.3.1, we derive the values of all parameters; in Section A.3.2, we derive the computational costs of the main steps. We then conclude with the correctness and cost of their algorithm. The main export of this section is the following theorem.

Theorem A.3.1 (Run Time [AK07]). *Given $C \in \mathbb{R}^{n \times n}$ with $m \geq n$ non-zero entries and $0 < \varepsilon \leq \frac{1}{2}$, we can find, in time $\tilde{O}\left(m/\varepsilon^5\right)$, a matrix $Y \in \mathbb{S}^n$ with $O(m)$ non-zero entries and a diagonal matrix $S \in \mathbb{R}^{n \times n}$ such that $\widetilde{X}^* = S \cdot \frac{K \exp(Y)}{\operatorname{Tr} \exp(Y)} \cdot S$ satisfies $\widetilde{X}^* \succeq 0$, $\widetilde{X}^*_{ii} \leq 1$ for all $i \in [n]$, and $\mathbb{E}(C \bullet \widetilde{X}^*) \geq C \bullet X^* - \varepsilon \cdot \sum_{i,j} |C|_{ij}$.*

A.3.1 Parameters

As can be seen in Algorithm A.2.1, approximate lazy mirror descent requires five parameters: the set diameter, Lipschitz constant of the objective, strong convexity of the mirror map, step size, and number of iterations. The first three depend on our choice of mirror map Φ and objective f. The last two can be chosen based on these parameters and Inequality 2.1.6.

Lemma A.3.2 (Set Diameter). *Given $\Phi(X) = X \bullet \log X$ and the domain $\{X : X \succeq 0, \operatorname{Tr} X = n\}$, the set diameter measured by Φ is given by $D = n \log n$.*

Lemma A.3.3 (Lipschitz constant). *The problem objective $\widehat{f}(X) = -\widehat{C} \bullet X + \sum_{i=1}^n (X_{ii} - \rho_i)^+$ (recall that $\rho_i = \sum_{j=1}^n \left|C_{ij}\right|$) is 2-Lipschitz in the nuclear norm. Recall that nuclear norm of a matrix is the sum of its singular values.*

Algorithm A.3.1 Reinterpreting [AK07]

Input: Cost matrix $C \in \mathbb{R}^{n\times n}$, accuracy parameter ε.

Parameters: $T = 256\log n/\varepsilon^2$, $T' = 10240\log n/\varepsilon^2$, $T'' = (1/\varepsilon)\cdot 64\log n$, $\eta = \varepsilon/64$. Set $\widehat{C}$ and ρ as defined in Lemma 2.1.4.

1: Initialize $Y^{(1)} \leftarrow \mathbf{0}$.
2: Define $\nabla f(M) \stackrel{\text{def}}{=} \mathbf{diag}(\mathbf{1})_{M\geq\rho} - \widehat{C}$.
3: **for** $t = 1 \to T$ **do**
4: $\quad \widetilde{\exp}\left(\frac{1}{2}Y^{(t)}\right) \leftarrow \mathbf{TaylorExp}\left(\frac{1}{2}Y^{(t)}, T''\right)$. ▷ Approximate matrix exponential
5: $\quad \widetilde{\exp}Y^{(t)} \leftarrow \mathbf{RandProj}\left(\widetilde{\exp}\left(\frac{1}{2}Y^{(t)}\right), T'\right)$. ▷ Approximate projection
6: $\quad \widetilde{X}^{(t)} \leftarrow n\frac{\widetilde{\exp}(Y^{(t)})}{\mathrm{Tr}\,\widetilde{\exp}Y^{(t)}}$ ▷ Scaling due to the trace constraint
7: $\quad Y^{(t+1)} \leftarrow Y^{(t)} - \eta\nabla f(\widetilde{X}^{(t)})$. ▷ Gradient update.
8: **end for**
9: For $t^* \stackrel{\text{unif.}}{\sim} \{1,2,\ldots,T\}$, return $Y^{(t^*)}$ and S, where S is from Lemma 2.1.4.

Proof. The gradient of the objective at point X is $\nabla\widehat{f}(X) = \mathbf{diag}(\mathbf{1}_{\{X\geq\rho\}}) - \widehat{C}$. By the Gershgorin Disk Theorem, we have

$$\|\mathbf{diag}(()) \tfrac{1}{\rho}C\|_{\mathrm{op}} \leq \max_{i\in[n]}\left(\frac{1}{\rho_i}\cdot|C_{ii}| + \frac{1}{\rho_i}\cdot\sum_{j\neq i}\left|C_{ij}\right|\right) = \max_{i\in[n]}\left(\frac{1}{\rho_i}\cdot\sum_{j=1}^{n}\left|C_{ij}\right|\right) = 1, \tag{A.3.1}$$

where in the last equality we use the choice of $\rho_i = \sum_{j=1}^{n}\left|C_{ij}\right|$. Since the matrices $\mathbf{diag}(1/\rho)\cdot C$ and $\widehat{C} = \mathbf{diag}(1/\sqrt{\rho})\cdot C\cdot\mathbf{diag}(1/\sqrt{\rho})$ are similar, they have the same set of eigenvalues (and therefore, the same operator norm). Therefore

$$\|\mathbf{diag}(\mathbf{1}_{\{X\geq\rho\}}) - \widehat{C}\|_{\mathrm{op}} \leq \|\mathbf{diag}(\mathbf{1}_{\{X\geq\rho\}})\|_{\mathrm{op}} + \|\widehat{C}\|_{\mathrm{op}} = 1 + 1 = 2.$$

When we have $\|\nabla\widehat{f}\| \leq G$ for some G, it implies f is G-Lipschitz in $\|\cdot\|_*$ (the dual norm). Therefore, in our case, we have that $\widehat{f}$ is 2-Lipschitz in the nuclear norm (dual of the operator norm). □

Lemma A.3.4 (Strong Convexity)**.** *([KST09]) The mirror map $\Phi(X) = X\bullet\log X$ is $1/(2n)$-strongly convex with respect to the nuclear norm on the domain $\{X\in\mathbb{S}^n : X\geq 0, \mathrm{Tr}(X) = n\}$.*

Lemma A.3.5. *Choosing $\eta = \varepsilon/64$ and $T = 256\log n/\varepsilon^2$ in Algorithm A.3.1 gives an accuracy of εn.*

Proof. We show in Lemma A.3.11 that Algorithm A.3.1 maintains the invariant $\mathbb{E}\|X^{(t)} - \widetilde{X}^{(t)}\| \leq \delta = \varepsilon n/4$. Therefore we are in the framework of approximate lazy mirror descent and can use its error bound from Inequality 2.1.6 and bound it by εK. We plug in the parameters from Lemma A.3.2, Lemma A.3.3, and Lemma A.3.4 in the bound and get

$$\mathbb{E}f(\overline{x}^{(t^*)}) - f(x^*) \leq \frac{n\log n}{T\eta} + \frac{2\eta\cdot 2^2}{1/2n} + \left(\frac{\varepsilon n}{4}\right)\cdot 2.$$

We optimize for η by setting the first two terms equal, and get

$$\eta = \tfrac{1}{4}\sqrt{\frac{\log n}{T}}. \tag{A.3.2}$$

With this expression for η, setting the bound for the right-hand side above to be εn gives $T \geq 256 \log n/\varepsilon^2$; plug this back in Equation (A.3.2) to get $\eta = \varepsilon/64$. □

A.3.2 Computational Cost

From Algorithm A.3.1, we see that there are three main parts to be computed to get the overall cost of the Arora-Kale algorithm: the number of iterations, the number of JL projections per iteration, and the cost of approximating a matrix exponential and multiplying it with a vector. We derive these values in this section.

Taylor Approximation for Matrix Exponential

In Algorithm A.3.1, before we do the randomized projection to get the diagonal entries, we approximate the matrix exponential $\widetilde{\exp}\left(Y^{(t)}/2\right) = \textbf{TaylorExp}\left(Y^{(t)}/2, T''\right)$. Here we show a bound on $\left|\frac{\exp(Y^{(t)})_{ii}}{\operatorname{Tr}\exp(Y^{(t)})} - \frac{\widetilde{\exp}(Y^{(t)})_{ii}}{\operatorname{Tr}\widetilde{\exp}(Y^{(t)})}\right|$ for any $1 \leq i \leq n$. We do so by first proving a bound on $\left|\frac{A_{ii}}{\operatorname{Tr} A} - \frac{B_{ii}}{\operatorname{Tr} B}\right|$ for a matrix B approximating the general matrix A; then we prove a general result on the number of terms required to approximate a matrix exponential using Taylor series; finally, we combine these results to get an appropriate choice of T_{poly} for approximating $\exp\left(Y^{(t)}/2\right)$.

Lemma A.3.6. *Given positive definite matrices A and B such that $\|A - B\|_{op} \leq \varepsilon$, where $\varepsilon \leq \frac{1}{2n}\operatorname{Tr} A$, we have $\left|\frac{A_{ii}}{\operatorname{Tr} A} - \frac{B_{ii}}{\operatorname{Tr} B}\right| \leq 2\frac{\varepsilon(\operatorname{Tr} A + nA_{ii})}{(\operatorname{Tr} A)^2}$.*

Proof. We have the following chain of inequalities.

$$\left|\frac{B_{ii}}{\operatorname{Tr} B} - \frac{A_{ii}}{\operatorname{Tr} A}\right| \overset{①}{\leq} \left|\frac{A_{ii}+\varepsilon}{\operatorname{Tr} A - n\varepsilon} - \frac{A_{ii}}{\operatorname{Tr} A}\right| = \frac{\varepsilon\,(\operatorname{Tr} A + nA_{ii})}{(\operatorname{Tr} A)\,(\operatorname{Tr} A - \varepsilon n)} \overset{②}{\leq} 2\frac{\varepsilon\,(\operatorname{Tr} A + nA_{ii})}{(\operatorname{Tr} A)^2},$$

where ① is by the worst case values for B_{ii} from the operator norm bound, and ② is by the bound on ε. □

Lemma A.3.7. *For $T \geq e^2\|Y\|_{op}$, we have $\|\exp(Y) - \sum_{j=0}^{T}\frac{Y^j}{j!}\|_{op} \leq \exp(-T)$.*

Proof. We have the following chain:

$$\|\exp(Y) - \sum_{j=0}^{T}\tfrac{1}{j!}Y^j\|_{\text{op}} \overset{①}{=} \|\sum_{j=T+1}^{\infty}\tfrac{1}{j!}Y^j\|_{\text{op}} \overset{②}{\leq} \sum_{j=T+1}^{\infty}\|\tfrac{1}{j!}Y^j\|_{\text{op}} = \sum_{j=T+1}^{\infty}\tfrac{1}{j!}\|Y\|_{\text{op}}^j \overset{③}{\leq} \sum_{j=T+1}^{\infty}\tfrac{e^j}{j^j}\|Y\|_{\text{op}}^j, \tag{A.3.3}$$

where ① is by the Taylor series expansion of the matrix exponential, ② is by triangle inequality of norms, and ③ is by Stirling's approximation, $j! \geq (j/e)^j$. Since the right hand side of the above

inequality is indexed over $j \geq T \geq e^2\|Y\|_{\text{op}}$, we can bound it further to get

$$\|\exp Y - \sum_{j=0}^{T} \tfrac{1}{j!} Y^j\|_{\text{op}} \leq \sum_{j=T+1}^{\infty} e^{-j} = \frac{(e^{-1})^{T+1}}{1-e^{-1}} \leq e^{-T}.$$

□

Lemma A.3.8. *In Algorithm A.3.1, for $n \geq 2$ and $\varepsilon \leq \frac{1}{2}$, set $T_{poly} = \frac{64 \log n}{\varepsilon}$, and let $\widetilde{\exp}\left(Y^{(t)}/2\right) :=$ **TaylorExp**$\left(Y^{(t)}/2, T_{poly}\right)$. Then for each coordinate i, we have $\left|\frac{\exp(Y^{(t)})_{ii}}{\operatorname{Tr}\exp Y^{(t)}} - \frac{(\widetilde{\exp} Y^{(t)})_{ii}}{\operatorname{Tr}\widetilde{\exp} Y^{(t)}}\right| \leq \frac{\varepsilon}{8n}$.*

Proof. Let $\widetilde{\exp}\left(Y^{(t)}/2\right) = \exp\left(Y^{(t)}/2\right) + \Delta$, and $\|\Delta\|_{\text{op}} = \varepsilon_1$. Then

$$\begin{aligned}
\|\exp Y^{(t)} - \widetilde{\exp} Y^{(t)}\|_{\text{op}} &= \|(\exp\left(Y^{(t)}/2\right))^2 - (\widetilde{\exp}(Y^{(t)}/2))^2\|_{\text{op}} \\
&= \|\Delta^2 + \Delta \exp\left(Y^{(t)}/2\right) + \exp\left(Y^{(t)}/2\right)\Delta\|_{\text{op}} \\
&\leq \varepsilon_1^2 + 2\varepsilon_1 \|\exp\left(Y^{(t)}/2\right)\|_{\text{op}}. \qquad \text{(A.3.4)}
\end{aligned}$$

Observe that in each iteration of Algorithm A.3.1, we add $-\eta \nabla f(\widetilde{X}^{(t)})$ to the current $Y^{(t)}$ in the gradient step; therefore at the end of all the T iterations, $\|Y^{(t)}\|_{\text{op}} \leq \left|\eta T\right| \|\nabla f(\widetilde{X}^{(t)})\|_{\text{op}}$. From the values of η and T as set in Algorithm A.3.1 (and explained in Section A.3), the worst-case value is

$$\|Y^{(t)}/2\|_{\text{op}} \leq \frac{1}{2} \cdot \frac{\varepsilon}{64} \cdot \frac{256 \log n}{\varepsilon^2} \cdot 2 = \frac{4 \log n}{\varepsilon}. \qquad \text{(A.3.5)}$$

Next, from Lemma A.3.7, we require the first $\max\left\{e^2\|Y^{(t)}/2\|_{\text{op}}, \log(1/\varepsilon_1)\right\}$ terms of the Taylor series of $\exp\left(Y^{(t)}/2\right)$ to get an ε_1 accuracy in approximation. Since $T_{\text{poly}} = 64 \log n/\varepsilon \geq e^2\|Y^{(t)}/2\|_{\text{op}}$ (from Inequality A.3.5), this choice of number of Taylor series terms corresponds to an accuracy of $\varepsilon_1 = n^{-64/\varepsilon}$. From Inequality A.3.5, we get that

$$\|\exp\left(Y^{(t)}/2\right)\|_{\text{op}} \leq e^{4 \log n/\varepsilon} = n^{4/\varepsilon}. \qquad \text{(A.3.6)}$$

Then we have

$$\begin{aligned}
\varepsilon_1^2 + 2\varepsilon_1 \|\exp\left(Y^{(t)}/2\right)\|_{\text{op}} &\leq n^{-128/\varepsilon} + 2n^{-64/\varepsilon} n^{4/\varepsilon} \\
&\leq 4n^{-60/\varepsilon} \\
&\leq \frac{n^{-4/\varepsilon}}{2} \leq \frac{1}{2n} \operatorname{Tr}\exp\left(Y^{(t)}/2\right), \qquad \text{(A.3.7)}
\end{aligned}$$

where the last inequality is by Inequality A.3.6. Chaining Inequality A.3.4 and Inequality A.3.7, the

condition in Lemma A.3.6 is satisfied. Applying the result of Lemma A.3.6,

$$\begin{aligned}\left|\frac{\left(\exp\left(Y^{(t)}\right)\right)_{ii}}{\operatorname{Tr}\exp\left(Y^{(t)}\right)}-\frac{\left(\widetilde{\exp}\left(Y^{(t)}\right)\right)_{ii}}{\operatorname{Tr}\widetilde{\exp}\left(Y^{(t)}\right)}\right| &\leq 2\left(\varepsilon_1^2+2\varepsilon_1\|\exp\left(Y\right)\|_{\mathrm{op}}\right)\frac{\operatorname{Tr}\exp\left(Y^{(t)}\right)+n\exp\left(Y^{(t)}\right)_{ii}}{\left(\operatorname{Tr}\exp\left(Y^{(t)}\right)\right)^2}.\\ &\leq 2\frac{\left(\varepsilon_1^2+2\varepsilon_1 n^{4/\varepsilon}\right)\left(2n^{1+8/\varepsilon}\right)}{\left(n^{-8/\varepsilon}\right)^2}\\ &\leq 4\left(\frac{\varepsilon^2}{10000n^{41/\varepsilon}}+\frac{\varepsilon}{50n^{4/\varepsilon}}\right)\\ &\leq \frac{\varepsilon}{8n}\end{aligned}$$

□

Randomized Projections

Suppose we approximate each entry of a vector using randomized projections. Then we can state the following result about the accuracy of the function $g(x) = x_i/\|x\|_1$.

Lemma A.3.9. *For $0 \neq X \in \mathbb{S}^n$, let $\widetilde{X} = \textbf{\textit{RandProj}}(X, 10240\log n/\varepsilon^2)$. Then $\left|\frac{\widetilde{X}_{ii}}{\operatorname{Tr}\widetilde{X}}-\frac{X_{ii}^2}{\operatorname{Tr}X^2}\right| \leq \frac{\varepsilon}{8}$.*

To prove this result, we need the Johnson-Lindenstrauss lemma.

Lemma A.3.10 ([JL84])**.** *For any $0 < \varepsilon < 1$, and any integer n, let k be a positive integer such that $k \geq 20\log n/\varepsilon^2$. Then for any set V of n points in $\mathbb{R}^d$ and random matrix $A \in \mathbb{R}^{k\times d}$, with high probability, for all $x \in V$,*

$$(1-\varepsilon)\|x\|_2^2 \leq \|(1/\sqrt{k})Ax\|_2^2 \leq (1+\varepsilon)\|x\|_2^2.$$

Proof of Lemma A.3.9. Applying Lemma A.3.10 to $\widetilde{X} = \textbf{RandProj}\left(X, \frac{10240\log n}{\varepsilon^2}\right)$, we have that with high probability, $\left|X_{ii}^2-\widetilde{X}_{ii}\right| \leq \frac{\varepsilon}{32}\left|X_{ii}^2\right|$. Therefore, $\operatorname{Tr}X^2\left(1-\frac{\varepsilon}{32}\right) \leq \operatorname{Tr}\widetilde{X}^2 \leq \operatorname{Tr}X^2\left(1+\frac{\varepsilon}{32}\right)$. Therefore $\frac{X_{ii}^2(1-\varepsilon/32)}{\operatorname{Tr}X^2(1+\varepsilon/32)} \leq \frac{\widetilde{X}_{ii}}{\operatorname{Tr}\widetilde{X}} \leq \frac{X_{ii}^2(1+\varepsilon/32)}{\operatorname{Tr}X^2(1-\varepsilon/32)}$ which can be simplified to $\frac{X_{ii}^2}{\operatorname{Tr}X^2}(1-\varepsilon/16) \leq \frac{\widetilde{X}_{ii}}{\operatorname{Tr}\widetilde{X}} \leq \frac{X_{ii}^2}{\operatorname{Tr}X^2}(1+\varepsilon/8)$, where the last simplification is by the inequalities $\frac{1-x}{1+x} \geq 1-2x$ and $\frac{1+x}{1-x} \leq 1+4x$ for $x \in \left(0, \frac{1}{2}\right)$. Therefore we have that $\left|\frac{\widetilde{X}_{ii}}{\operatorname{Tr}\widetilde{X}}-\frac{X_{ii}^2}{\operatorname{Tr}X^2}\right| \leq (\varepsilon/8)\frac{X_{ii}^2}{\operatorname{Tr}X^2} \leq \varepsilon/8$. □

Number of Iterations

From Lemma A.3.8 and Lemma A.3.9 proved above, we can infer that the choice of T'' and T' in Algorithm A.3.1 gives us the following overall error in approximating the true primal iterate.

Lemma A.3.11. *In Algorithm A.3.1, we have that $\|\widetilde{X}^{(t)} - X^{(t)}\| \leq \frac{\varepsilon n}{4}$.*

Proof. The quantity we want to bound is $\left\| \frac{n \exp\left(Y^{(t)}\right)}{\operatorname{Tr} \exp\left(Y^{(t)}\right)} - \frac{\widetilde{X}^{(t)}}{\operatorname{Tr} \widetilde{X}^{(t)}} \right\|$. Each term is bounded as:

$$\left| \frac{n \exp\left(Y^{(t)}\right)_{ii}}{\operatorname{Tr} \exp\left(Y^{(t)}\right)} - \frac{\widetilde{X}^{(t)}_{ii}}{\operatorname{Tr} \widetilde{X}^{(t)}} \right| \leq n \underbrace{\left| \frac{\exp\left(Y^{(t)}\right)_{ii}}{\operatorname{Tr} \exp\left(Y^{(t)}\right)} - \frac{\widetilde{\exp}\left(Y^{(t)}\right)_{ii}}{\operatorname{Tr} \widetilde{\exp}\left(Y^{t}\right)} \right|}_{\textbf{TaylorExp error}} + \underbrace{\left| \frac{n\widetilde{\exp}\left(Y^{(t)}\right)_{ii}}{\operatorname{Tr} \widetilde{\exp}\left(Y^{(t)}\right)} - \frac{n\widehat{\exp}\left(Y^{(t)}\right)_{ii}}{\operatorname{Tr} \widehat{\exp}\left(Y^{(t)}\right)} \right|}_{\textbf{RandProj error}}.$$

Apply the results of Lemma A.3.8 and Lemma A.3.9 to the right hand side terms. □

Corollary A.3.12. *The number of iterations for convergence of the Arora-Kale algorithm is* $O(1/\varepsilon^2)$.

Proof. Since the Arora-Kale algorithm only depends on the diagonal entries of X, we can assume that $\widetilde{X}$ and X match on the off-diagonal entries. Then, $\|\widetilde{X}^{(t)} - X^{(t)}\| \leq \frac{\varepsilon n}{4}$ is exactly equivalent to $\|\widetilde{X}^{(t)} - X^{(t)}\|_{\text{nuc}} \leq \frac{\varepsilon n}{4}$. Therefore the algorithm meets the conditions of Algorithm A.2.1 with $\delta = \frac{\varepsilon n}{4}$. Therefore by Theorem 2.1.3, the number of outer iterations required for convergence is $O(1/\varepsilon^2)$. □

Combining All the Costs

Recall from Algorithm A.3.1 that $T' = O(1/\varepsilon^2)$, $T'' = O(1/\varepsilon)$, and the number of iterations is $O(1/\varepsilon^2)$. The cost of a matrix-vector product is $O(m)$. Therefore, multiplying these costs gives $O(m/\varepsilon^5)$, the claimed cost of Arora-Kale algorithm. This completes the analysis.

A.4 Analysis of our Proposed Algorithm

We now analyze Algorithm 2.2.1, organizing this section as follows. In Section A.4.1 we derive the values of parameters that appear in the error bounds. Next, in Section A.4.2, we show how we construct a polynomial to approximate the matrix exponential. In Section A.4.3, we prove properties of the constructed estimators. We derive the number of inner iterations we have in Section A.4.4. In Section A.4.5, we establish the crucial distance invariance between true and estimated iterates, which ensures that our error is always under control. We next show in Section A.4.6 why we do not need to normalize our projection step, which enables us to have a simple projection. Finally, we derive the error bound in Section A.4.7.

A.4.1 Parameters of Mirror Map

As before, there are two parameters of the mirror map that we need to use in Theorem 2.1.3: the diameter of the constraint set as measured by it, and its strong convexity parameter.

Lemma A.4.1 (Set Diameter)**.** *Given* $\Phi(X) = X \bullet \log X - \operatorname{Tr} X$ *and the domain* $\mathcal{D} = \{X : X \succeq 0, \operatorname{Tr} X \leq K\}$, *where* $K \geq n$, *the set diameter measured by* Φ *is given by* $D = K \log K$.

Lemma 2.2.9. *The function* $\Phi(X) = X \bullet \log X - \operatorname{Tr} X$ *is* $\frac{1}{4K}$*-strongly convex with respect to the nuclear norm over the domain* $\mathcal{D} = \{X : X \succeq 0, \operatorname{Tr} X \leq K\}$.

To prove the claimed strong convexity, we need the following tools.

Definition A.4.2. *A function* $f : \mathbb{R}^n \to \mathbb{R}$ *is L-smooth in norm* $\|\cdot\|$ *if it is continuously differentiable and satisfies* $\|\nabla f(x) - \nabla f(y)\|_* \leq L\|x - y\|$ *for all* x *and* y *in* **dom** f.

For functions on symmetric matrices, we use the following equivalent definition of smoothness.

Definition A.4.3. *A function $f : \mathbb{S}^n \to \mathbb{R}$ is L-smooth in $\|\cdot\|$ if and only if for $h : \mathbb{R} \to \mathbb{R}$ defined as $h(t) = f(X + tH)$ for $H \in \mathbb{S}^n$ such that $X + tH \in \mathbf{dom}(f)$, we have $h''(0) \leq L\|H\|^2$.*

Theorem A.4.4 ([KST09])**.** *Assume that f is a closed and convex function. Then f is β-strongly convex with respect to a norm $\|\cdot\|$ if and only if its Fenchel dual, f^*, is $\frac{1}{\beta}$-smooth with respect to the dual norm $\|\cdot\|_*$.*

Theorem A.4.5 ([JN08])**.** *Let Δ be an open interval on the real axis, and f be a twice differentiable function on Δ satisfying, for a certain $\theta \in \mathbb{R}$, for all $a < b$, where $a, b \in \Delta$, $\frac{f'(b)-f'(a)}{b-a} \leq \theta\frac{f''(a)+f''(b)}{2}$. Let $\mathcal{X}_n(\Delta)$ be the set of all $n \times n$ symmetric matrices with eigenvalues belonging to Δ. Then for $X \in \mathcal{X}_n(\Delta)$, the function $F(X) = \operatorname{Tr} f(X)$ is twice differentiable, and for every $H \in \mathbb{S}^n$, we have $D^2F(X)[H, H] \leq \theta \operatorname{Tr}(Hf''(X)H)$.*

Theorem A.4.6 ([Lew95])**.** *Suppose that the function $f : \mathbb{R}^n \to \mathbb{R}$ is symmetric (that is, $f(\sigma x) = f(x)$ for all $x \in \mathbf{dom}\, f$ and all permutations σ). Then if f is convex and lower semicontinuous, the corresponding unitarily invariant function $f \circ \lambda$ is convex and lower semicontinuous on $\mathbb{R}^{n\times n}$*

For our proof, we use definitions from Definition 2.2.10 in the following way. We first show that Ψ satisfies

$$\Psi^*(Y) = \Phi(Y), \text{ on } \{Y : Y \succeq 0, \operatorname{Tr} Y \leq K\}, \tag{A.4.1}$$

where $\Phi(Y) = Y \bullet \log Y - \operatorname{Tr} Y$ is the mirror map, as defined in the statement of the lemma. We then prove that Ψ is β-smooth with respect to the operator norm for a certain $\beta > 0$. Theorem A.4.4 then immediately implies $1/\beta$-strong convexity of Ψ^* with respect to the nuclear norm. Then Equation (A.4.1) implies that Φ is $1/\beta$-strongly convex with respect to the nuclear norm on the domain $\{Y : Y \succeq 0, \operatorname{Tr} Y \leq K\}$, which is to be proved. We accomplish our first goal (Equation (A.4.1)) in the following sequence of steps.

Claim A.4.7 proves that the function ψ and its matrix version, Ψ, are both continuously differentiable at the boundary of definition of the two pieces. Claim A.4.8 then proves that ψ_1 and ψ_2 are convex; in conjuncation with Claim A.4.7, this implies ψ is convex. Applying Theorem A.4.6 extends the property of convexity to Ψ. Claim A.4.9 proves that the vector functions ψ and ϕ satisfy $\psi_1^*(x) = \phi(x)$ for $x \in \mathbb{R}^n_+$. Claim A.4.10 proves that given an input point $x \in \{x : x_i \geq 0, \sum_{i=1}^n x_i \leq K\}$, the point y which attains the optimum in computing $\psi_1^*(x)$ lies in the *interior* of the set $\{y : \psi_1(y) \leq 2K\}$. Claim A.4.11 shows that $\psi^*(x) = \psi_1^*(x)$ for $x \in \{x : x_i \geq 0, \sum_{i=1}^n x_i \leq K\}$. This is obtained by combining the results of Claim A.4.8 and Claim A.4.10.

We then use these results as follows: since on the set $\{x : x_i \geq 0, \sum_{i=1}^n x_i \leq K\}$, we have $\psi^* = \phi$, this implies $\Psi^* = \Phi$ on the corresponding set, $\{X : X \succeq 0, \operatorname{Tr} X \leq K\}$. Next, to show smoothness of Ψ, we use Theorem A.4.5 to compute the smoothness constants of each part of Ψ (in Claim A.4.12 and Claim A.4.13), and then combine with continuous differentiability at the boundary (from Claim A.4.7) to get the overall smoothness constant of Ψ. By the argument at the start of this proof, this immediately proves the desired strong convexity parameter. We now proceed to prove all the claims aluded to above.

Claim A.4.7. *The functions Ψ and ψ are both continuously differentiable at the boundary.*

Proof of Claim. One can check that $\psi_1(y) = \psi_2(y)$ at the boundary. This implies continuity of the function ψ. The derivatives of the two functions are also the same at the boundary. The i-th component of the gradient is given by $\nabla_i\psi_2(y) = \frac{2K\nabla_i\psi_1(y)}{\psi_1(y)}$. At the boundary of the two parts of the function, we have $\psi_1(y) = 2K$. Substituting this into the above gradient gives $\nabla\psi_2(y) = \nabla\psi_1(y)$. This shows that ψ is continuously differentiable at the boundary. We only used chain rule of derivatives here, which applies to matrices as well, so the exactly same reasoning also gives that Ψ is continuously differentiable at the boundary. □

Claim A.4.8. *The functions ψ and Ψ are convex on their domains.*

Proof. The function ψ is a piecewise function, each piece composed of a standard convex function (see [BV04b]). Combine with continuous differentiability from Claim A.4.7 gives convexity of ψ. Applying Theorem A.4.6 implies convexity of Ψ. □

Claim A.4.9. *For any input $x \in \mathbb{R}^n_+$, we have $\psi_1^*(x) = \phi(x)$.*

Proof of Claim. By definition, we have $\psi_1^*(x) = \sup_y(x^\top y - \sum_{i=1}^n \exp(y_i))$. Observe that the domain of ψ_1^* is $\mathbb{R}^n_+$ (because if there exists an input with a negative coordinate, then the corresponding coordinate of the maximizer y^* can be made to go to $-\infty$). Therefore, given an input $x \in \mathbb{R}^n_+$, the supremum is attained at y^* satisfying $x_i = \exp\left(y_i^*\right)$. This means the maximizer is $y_i^* = \log x_i$. Therefore the conjugate is $\psi_1^*(x) = \sum_{i=1}^n x_i \log x_i - \sum_{i=1}^n x_i = \phi(x)$. □

Claim A.4.10. *For any x in the set $\left\{x : x_i \geq 0, \sum_{i=1}^n x_i \leq K\right\}$, the point $y^* = \operatorname{argmax}\left(x^T y - \psi_1(y)\right)$ lies in* **int** *$\{y : \psi_1(y) \leq 2K\}$, where* **int** *denotes the interior of the set.*

Proof of Claim. From the proof of Claim A.4.9, for any $x \in \mathbb{R}^n_+$, we have that $y^* = \operatorname{argmax}\left(x^T y - \psi_1(y)\right)$ satisfies $y_i^* = \log x_i$ for $1 \leq i \leq n$. In addition to this, the statement of the lemma also requires the input x to satisfy $x_i \geq 0, \sum_{i=1}^n x_i \leq K$. Plug in the values of x in terms of y in the above inequality to get $\sum_{i=1}^n \exp y_i^* \leq K$, which is the same as saying $\psi_1(y^*) \leq K < 2K$. This shows that the optimum, y^*, lies in **int** $\{y : \psi_1(y) \leq 2K\}$. □

Claim A.4.11. *We have $\psi^*(x) = \psi_1^*(x)$ on all $x \in \left\{x : x_i \geq 0, \sum_{i=1}^n x_i \leq K\right\}$.*

Proof of Claim. By definition of conjugate and ψ,

$$\begin{aligned} \psi^*(x) &= \sup_y x^T y - \psi(y) \\ &= \sup_y x^T y - \begin{cases} \psi_1(y) & \text{if } \psi_1(y) \leq 2K \\ \psi_2(y) & \textit{otherwise} \end{cases} \end{aligned} \tag{A.4.2}$$

From Claim A.4.8, ψ is convex. Therefore the function to be maximized in Equation (A.4.2) is concave. From Claim A.4.10, for input x in the set $\left\{x : x_i \geq 0, \sum_{i=1}^n x_i \leq K\right\}$, we have that the maximizer $\operatorname{argmax}_y \left(x^T y - \psi_1(y)\right)$ lies in the interior of $\{y : \psi_1(y) \leq 2K\}$. Therefore for input $x \in \left\{x : x_i \geq 0, \sum_{i=1}^n x_i \leq K\right\}$, the maximizer of Equation (A.4.2) is also the same as that of $\psi_1^*(x)$. This gives $\psi^*(x) = \psi_1^*(x)$. □

Claim A.4.12. *The function $\Psi_1(Y)$ defined over $\{Y : \operatorname{Tr} \exp Y \leq 2K\}$ is $4K$-smooth.*

Proof of Claim. Let $g(u) \stackrel{\text{def}}{=} \exp(u)$. The function g is convex and differentiable (any number of times). In particular, g'' is convex. For any a, b, applying the Mean Value theorem to some point $\zeta \in (a, b)$, convexity of g'', and $g'' \geq 0$ (due to convexity of g) gives

$$\frac{g'(b) - g'(a)}{b - a} = g''(\zeta) \leq \max\left(g''(a), g''(b)\right) \leq 2\frac{g''(a) + g''(b)}{2}.$$

This satisfies the right-hand side condition for Theorem A.4.5 with $\theta = 2$; so Theorem A.4.5 implies that on the domain $\{Y : \operatorname{Tr}\exp Y \le K\}$, for $h(t) \stackrel{\text{def}}{=} \sum_{i=1}^{n} g(\lambda_i(Y + tH)) = \operatorname{Tr}\exp(Y + tH)$, we have,

$$\begin{aligned} h''(0) = D^2\Psi_1(Y)[H,H] &\le 2\operatorname{Tr}(Hg''(Y)H) \\ &= 2\operatorname{Tr}\left(\exp(Y)H^2\right) \\ &\le 2\operatorname{Tr}\exp(Y)\cdot\|H\|_{\text{op}}^2 \\ &\le 2\cdot 2K\cdot\|H\|_{\text{op}}^2 \\ &= 4K\|H\|_{\text{op}}^2, \end{aligned} \tag{A.4.3}$$

where we used the domain constraint for Ψ_1 in the last inequality, and the fact that matrix exponential is positive semidefinite in the first (Hölder's) inequality. By Definition A.4.3 then, we have the lemma. □

Claim A.4.13. *The smoothness constant of $\Psi_2(Y)$ over the set $\{Y : \operatorname{Tr}\exp Y \ge 2K\}$ is $4K$.*

Proof. For ease of exposition, let $a \stackrel{\text{def}}{=} 2K$. Consider the same scalar function from Claim A.4.12, $h(t) = \operatorname{Tr}\exp(Y + tH)$ and $\ell(t) \stackrel{\text{def}}{=} a\log(h(t)) + 2K - 2K\log(2K)$. Then $\ell'(t) = a\frac{h'(t)}{h(t)}$ and $\ell''(t) = a\left(\frac{h''(t)}{h(t)} - \left(\frac{h'(t)}{h(t)}\right)^2\right) \le a\frac{h''(t)}{h(t)}$. In particular,

$$\ell''(0) \le a\frac{h''(0)}{h(0)}. \tag{A.4.4}$$

We already showed in Inequality A.4.3 that $h''(0) = D^2\Psi_1(Y)[H,H] \le 4K\|H\|_{\text{op}}^2$. We also have that $h(0) = \operatorname{Tr}\exp(Y) \ge 2K$ (by assumption of the lemma). Plugging these along with the value of a into Equation (A.4.4) gives us $\ell''(0) \le 2K\frac{4K}{2K}\cdot\|H\|_{\text{op}}^2 = 4K\|H\|_{\text{op}}^2$. This implies the claimed smoothness constant. □

Proof of Lemma 2.2.9. For the functions defined in Definition 2.2.10, we can combine Claim A.4.9 and Claim A.4.11 to get that $\psi^*(x) = \phi(x)$ for $x \in \{x : x_i \ge 0, \sum_{i=1}^{n} x_i \le K\}$. This implies the matrix version of this statement, $\Psi^*(X) = \Phi(X)$ for $X \in \{X : X \succeq 0, \operatorname{Tr} X \le K\}$. Next, applying Claim A.4.7, Claim A.4.12, and Claim A.4.13, we get that the function Ψ is continuously differentiable with smoothness constant $4K$. Invoking Theorem A.4.4, we immediately obtain that Ψ^* is strongly convex with parameter $\frac{1}{4K}$. This implies that Φ is strongly convex with the same parameter over the set $\{X : X \succeq 0, \operatorname{Tr} X \le K\}$. □

A.4.2 Chebyshev Approximation of the Matrix Exponential

In this section, we show how to construct a polynomial approximation of our matrix exponential. The standard technique to do so involves truncating the Taylor series of the matrix exponential; however, a quadratically improved bound on the number of terms required for the computation is provided by Sachdeva and Vishnoi [SV+14] using Chebyshev polynomials. We follow their notation and summarize their main results below for the sake of completeness.

A Brief Summary of Chebyshev Approximation

For a non-negative integer d, we denote by $T_d(x)$ the Chebyshev polynomials of degree d, defined recursively as follows:

$$\begin{aligned} T_0(x) &= 1, \\ T_1(x) &= x, \\ T_d(x) &= 2xT_{d-1}(x) - T_{d-2}(x). \end{aligned}$$

Let Y_i be i.i.d. variables taking values 1 and -1 each with probability $1/2$. Let $D_s = \sum_{i=1}^{s} Y_i$, $D_0 \stackrel{\text{def}}{=} 0$, and

$$p_{s,d}(x) \stackrel{\text{def}}{=} \mathbb{E}_{Y_1,Y_2,\ldots,Y_s}\left(T_{D_s}(x)\mathbf{1}_{|D_s|\le d}\right). \tag{A.4.5}$$

We can use these to construct a polynomial with degree roughly $\sqrt{s}$ that can well approximate x^s. The formal statement follows.

Theorem A.4.14 (Theorem 3.3 in [SV+14]). *For any positive integers s and d, the degree d polynomial $p_{s,d}$ defined by Equation (A.4.5) satisfies*

$$\sup_{x\in[-1,1]} \left|p_{s,d}(x) - x^s\right| \le 2\exp\left(-d^2/(2s)\right).$$

Using this result, define the polynomial:

$$q_{\lambda,t,d}(x) \stackrel{\text{def}}{=} \exp(-\lambda)\sum_{i=0}^{t} \frac{(-\lambda)^i}{i!} p_{i,d}(x). \tag{A.4.6}$$

Then we can use q to approximate an exponential with the following error guarantee.

Lemma A.4.15 (Lemma 4.2 of [SV+14]). *For every $\lambda > 0$ and $\delta \in (0, 1/2]$, we can choose $t = \max(\lambda, \log(1/\delta))$ and $d = \sqrt{t\log(1/\delta)}$ such that*

$$\sup_{x\in[-1,1]} \left|\exp(-\lambda - \lambda x) - q_{\lambda,t,d}(x)\right| \le \delta.$$

This is a quadratic improvement over the standard cost (degree) of approximating an exponential using truncated Taylor series. Finally, this lemma can be used to generalize the approximation from the $[-1, 1]$ interval to the interval $[0, b]$, as stated below.

Theorem A.4.16 (Theorem 4.1 of [SV+14]). *For every $b > 0$, and $0 < \delta \le 1$, there exists a polynomial $r_{b,\delta}$ that satisfies*

$$\sup_{x\in[0,b]} \left|\exp(-x) - r_{b,\delta}(x)\right| \le \delta$$

and has degree $O(\sqrt{\max(b, \log(1/\delta)) \cdot \log(1/\delta)})$.

The proof of this theorem uses $\lambda \stackrel{\text{def}}{=} b/2$, and t and d from Lemma A.4.15 and the polynomial

$$r_{b,\delta}(x) \stackrel{\text{def}}{=} q_{\lambda,t,d}\left(\frac{1}{\lambda}(x - \lambda)\right). \tag{A.4.7}$$

Corollary A.4.17 (Our Chebyshev Approximation)**.** *For every $b > 0$, $a < b$, $0 < \delta \le 1$, and $d = \sqrt{\max\left(\frac{1}{2}(b-a), \log\left(\frac{1}{\delta}\right)\right)\log\left(\frac{1}{\delta}\right)}$, there exists a degree-$d$ polynomial $\boldsymbol{ChebyExp}(u, d, \delta)$ such that*

$$\sup_{u\in[a,b]} \left|\exp(u) - \boldsymbol{ChebyExp}(u, d, \delta)\right| \le \delta \exp(b). \tag{A.4.8}$$

Proof. Using a simple linear transformation, Theorem A.4.16 generalizes to:

$$\sup_{z\in[a,b]} \left|\exp\left(-\frac{1}{2}(b-a)\right)\sum_{i=0}^{t} \frac{(-\frac{1}{2}(b-a))^i}{i!} p_{i,d}\left(\frac{z-(b+a)/2}{(b-a)/2}\right) - \exp(-(z-a))\right| \le \delta.$$

By choosing $\lambda = \frac{1}{2}(b-a)$, and transforming $-z + a = u - b$, we get

$$\sup_{u\in[a,b]} \left|q_{\frac{1}{2}(b-a),t,d}\left(\frac{-u+(b+a)/2}{(b-a)/2}\right) - \exp(u-b)\right| \le \delta.$$

Using Equation (A.4.7) above gives

$$\sup_{u\in[a,b]} \left|\exp(b) r_{b-a,\delta}(b-u) - \exp(u)\right| \le \delta \exp(b).$$

Therefore, let $d = \sqrt{\max\left(\frac{1}{2}(b-a), \log\left(\frac{1}{\delta}\right)\right)\log\left(\frac{1}{\delta}\right)}$ and $\mathbf{ChebyExp}(u, d, \delta) = \exp(b) r_{b-a,\delta}(b-u)$. Substitute these into the last inequality to get the statement of the lemma. □

Chebyshev Approximation in Our Algorithm

We can use the above results to approximate a matrix exponential as follows. Observe that

$$\|\exp(Y) - \mathbf{ChebyExp}(Y, d, \delta)\|_{\mathrm{op}} = \max_{i\in[n]} \left|\exp(\lambda_i) - \mathbf{ChebyExp}(\lambda_i, d, \delta)\right|,$$

where λ_i are the eigenvalues of Y and **ChebyExp** is the subroutine described in Corollary A.4.17. We only need the spectrum of Y in order to complete the approximation, and that is what we proceed to derive below. Once we have the spectrum, we simply combine it with the above results to get Lemma A.4.19.

Lemma A.4.18. *The spectrum of Y lies in the range $\left[-\frac{1}{\varepsilon}60\log n, \log K\right]$.*

Proof. Recall that $Y = -\eta\nabla f(X)$. Since we start Algorithm 2.2.1 with $Y^{(1)} = 0$, at the t-th iteration, we have $Y^{(t)} = -\sum_{i=1}^{t} \eta\nabla f\left(X^{(t)}\right)$. Plugging in the parameters displayed in Table 2.1, we get that the total number of iterations of the algorithm is $T_{\text{inner}} \times T_{\text{outer}} = \frac{1}{\varepsilon^3} 24 \times 10^5 \left(\log(n/\varepsilon)\right)^{11} \log n$, the Lipschitz constant of the objective function is $\|\nabla f\|_{\mathrm{op}} \le 2$, and the step size is $\eta = \frac{\varepsilon^2}{8\times 10^4 \left(\log(n/\varepsilon)\right)^{11}}$. Multiplying these gives

$$\|Y^{(t)}\|_{\mathrm{op}} \le 2 \cdot \frac{\varepsilon^2}{8\times 10^4 \times (\log(n/\varepsilon))^{11}} \cdot \frac{24\times 10^5 \times (\log(n/\varepsilon))^{11} \log n}{\varepsilon^3} = \frac{1}{\varepsilon} 60 \log n.$$

Therefore, the spectrum of $Y^{(t)}$ lies in

$$\lambda(Y^{(t)}) \in \left[-\frac{1}{\varepsilon}60\log n, \frac{1}{\varepsilon}60\log n\right]. \tag{A.4.9}$$

We now show a better upper bound on the spectrum. Since our algorithm maintains $\operatorname{Tr} X^{(t)} \le K$ (see Lemma 2.2.8), and $X^{(t)} = \exp\left(Y^{(t)}\right)$, it implies $\operatorname{Tr}\exp\left(Y^{(t)}\right) \le K$. Since the matrix exponential is positive definite, this implies $\|\exp\left(Y^{(t)}\right)\|_{\text{op}} \le K$, and therefore,

$$\lambda_{max}(Y^{(t)}) \le \log K. \tag{A.4.10}$$

Combining the inclusion Equation (A.4.9) and Equation (A.4.10) gives the claimed bound on the spectrum. □

Lemma A.4.19. *In Algorithm A.3.1, for $n \ge 2$ and $\varepsilon \le \frac{1}{2}$, set $\mathrm{T}_{\text{Cheby}} = \frac{150}{\sqrt{\varepsilon}}\log(n/\varepsilon)$, $\delta_{\text{Cheby}} = (\varepsilon/n)^{401}$, and let $\widetilde{\exp}\left(Y^{(t)}/2\right) := \textbf{\textit{ChebyExp}}\left(Y^{(t)}/2, \mathrm{T}_{\text{Cheby}}, \delta_{\text{Cheby}}\right)$. Then for all $1 \le i \le n$,*

$$\left|\exp\left(Y^{(t)}\right)_{ii} - \left(\widetilde{\exp} Y^{(t)}\right)_{ii}\right| \le \delta_{\exp} \stackrel{def}{=} \frac{4800\varepsilon^{401}}{n^{390}}.$$

Proof. We plug into Equation (A.4.8) the following bounds obtained from Lemma A.4.18:

$$a = -\tfrac{60\log n}{\varepsilon}, b = \log K$$
$$u = \lambda = \tfrac{1}{2}(b-a) = \tfrac{\log K}{2} + \tfrac{30\log n}{\varepsilon}$$

Applying Equation (A.4.8), we then get

$$\sup_{\lambda \in \left[-\frac{30\log n}{\varepsilon}, \frac{1}{2}\log K\right]} \left| Kr_{\frac{1}{2}\log K + \frac{30\log n}{\varepsilon}, \delta}\left(\tfrac{1}{2}\log K - \tfrac{1}{2}\lambda\right) - \exp\left(\tfrac{1}{2}\lambda\right)\right| \le \delta K$$

We have $K = 40n(\log n)^{10}$; therefore, if we want the error bound to be roughly $\frac{\varepsilon}{n}$, then we need to pick $\delta = \text{polylog}(\varepsilon, n)$. Because of technical details in Lemma A.4.24, we choose

$$\delta_{\text{Cheby}} = \left(\frac{\varepsilon}{n}\right)^{401}. \tag{A.4.11}$$

This gives us the following result.

$$\|\exp\left(Y^{(t)}/2\right) - \textbf{ChebyExp}(Y^{(t)}/2, \mathrm{T}_{\text{Cheby}}, \delta_{\text{Cheby}})\|_{\text{op}} \le 40\frac{\varepsilon^{401}}{n^{396}}.$$

From Lemma A.4.15, we get that the degree of polynomial required to achieve this guarantee is

$$\text{Required Degree } = \sqrt{\frac{2\times 10^4}{\varepsilon}\log n \log(n/\varepsilon)} \le \frac{150}{\sqrt{\varepsilon}}\log(n/\varepsilon).$$

This is the value of $\mathrm{T}_{\text{Cheby}}$ that we choose. We now bound the quantity we actually care about. We can write $\widetilde{\exp}\left(\frac{1}{2}Y^{(t)}\right) = \exp\left(\frac{1}{2}Y^{(t)}\right) + \Delta$, where $\|\Delta\|_{\text{op}} = 40\frac{\varepsilon^{401}}{n^{396}}$, the error guarantee obtained above.

Simplifying with the application of $\|\exp\left(Y^{(t)}\right)\|_{\text{op}} \leq K$ obtained from Lemma 2.2.8 gives

$$\begin{aligned}
\|\exp\left(Y^{(t)}\right) - \widetilde{\exp}\left(Y^{(t)}\right)\|_{\text{op}} &= \|\left(\exp\left(\tfrac{1}{2}Y^{(t)}\right)\right)^2 - \left(\widetilde{\exp}\left(\tfrac{1}{2}Y^{(t)}\right)\right)^2\|_{\text{op}} \\
&= \|\Delta^2 + \Delta\exp\left(\tfrac{1}{2}Y^{(t)}\right) + \exp\left(\tfrac{1}{2}Y^{(t)}\right)\Delta\|_{\text{op}} \\
&\leq (40\frac{\varepsilon^{401}}{n^{396}})^2 + 2(40\frac{\varepsilon^{401}}{n^{396}})\|\exp\left(\tfrac{1}{2}Y^{(t)}\right)\|_{\text{op}} \\
&\leq (40\frac{\varepsilon^{401}}{n^{396}})^2 + 2(40\frac{\varepsilon^{401}}{n^{396}})K \\
&\leq 3(40\frac{\varepsilon^{401}}{n^{396}})K \\
&\leq 3\cdot\frac{40\varepsilon^{401}}{n^{396}}\cdot 40n\left(\log n\right)^{10} \\
&\leq \frac{4800\varepsilon^{401}}{n^{390}}.
\end{aligned}$$

Substituting our assumption $n \geq 4$ above gives the claimed bound. □

In conclusion, we showed that we can approximate our matrix exponential to ε-accuracy using $O(1/\sqrt{\varepsilon})$ terms in the polynomial approximation.

A.4.3 Properties of Estimators

Since we have an inner loop in Algorithm 2.2.1 with estimated quantities, it is crucial for the convergence that these estimators have a small bias and variance. In this section we show that this is indeed the case. We first prove two technical results about the functions **InvSqrt** and **RandProj** which are "building blocks" of our estimators. We then apply these results in proving properties of $\widehat{\theta}_1$ and $\widehat{\theta}_2$, and subsequently those of the overall estimator $\widehat{\theta}$.

Two Technical Results about Estimators

Lemma 2.2.6. *Consider a positive random variable x sampled from a distribution X with mean μ and variance σ^2. For some integer $k > 0$, construct the distribution $\mathcal{G}(X) = \textbf{\textit{InvSqrt}}(X, k)$ defined in Equation (2.2.2). Then the random variable $g \sim \mathcal{G}(X)$ satisfies*

(1) $|\mathbb{E}g - \mu^{-1/2}| \leq \mathbb{E}\left(\frac{|x-\mu|^k}{\min(\mu,x)^{k+1/2}}\right)$

(2) $\mathbb{E}|g|^2 \leq k\sum_{j=0}^{k-1}\mathbb{E}\left(\frac{\left(\sigma^2+(\mu-x)^2\right)^j}{x^{2j+1}}\right)$.

Proof. Recall that given a distribution $\widetilde{X}$ with a positive support, and integer $N > 0$, we define **InvSqrt** as the approximation for $g(u) = u^{-1/2}$ at x_0 sampled from $\widetilde{X}$:

$$\textbf{InvSqrt}(\widetilde{X}, N) = \sum_{k=0}^{N-1}\frac{1}{k!}g^{(k)}(x_0)\prod_{j=1}^{k}(x_{k,j} - x_0), \text{ where } x_0, x_{k,j} \overset{\text{i.i.d.}}{\sim} \widetilde{X},$$

where $g^{(k)}(u) = \frac{(-1)^k}{2^k}u^{-j-1/2}\prod_{\ell=1}^{j}(2\ell-1)$ denotes the k-th derivative of g evaluated at u. Then the

expected value of g with respect to the distribution $\mathcal{G}(X)$ is

$$\begin{aligned}\mathbb{E}g &= \mathbb{E}\sum_{j=0}^{k-1}\frac{1}{j!}g^{(j)}(x)\prod_{\ell=1}^{j}\left(x_{j,\ell}-x\right)\\ &= \mathbb{E}\sum_{j=0}^{k-1}\frac{1}{j!}g^{(j)}(x)\prod_{\ell=1}^{j}\left(\mathbb{E}x_{j,\ell}-x\right)\\ &= \mathbb{E}\sum_{j=0}^{k-1}\frac{1}{j!}g^{(j)}(x)\left(\mu-x\right)^{j}. \end{aligned} \tag{A.4.12}$$

To see how the term on the right hand side of Equation (A.4.12) differs from the true quantity to be estimated, we apply Taylor's remainder theorem: for some point ζ lying between μ and x, we have

$$\begin{aligned}\left|\sum_{j=0}^{k-1}\frac{1}{j!}g^{(j)}(x)\left(\mu-x\right)^{j}-\mu^{-1/2}\right| &\leq \frac{g^{(k)}(\zeta)}{k!}\left|x-\mu\right|^{k}\\ &\leq \frac{\left|x-\mu\right|^{k}}{\min\left(x,\mu\right)^{k+1/2}},\end{aligned}$$

where the second inequality follows from

$$\left|\frac{g^{(k)}(u)}{k!}\right| \leq u^{-k-\frac{1}{2}}, \tag{A.4.13}$$

and the fact that ζ lies between x and μ. Combining this with Jensen's inequality gives us the final bound on the first moment,

$$\left|\mathbb{E}g-\mu^{-1/2}\right| \leq \mathbb{E}\left|g-\mu^{-1/2}\right| \leq \mathbb{E}\frac{\left|x-\mu\right|^{k}}{\min\left(x,\mu\right)^{k+1/2}}. \tag{A.4.14}$$

To prove the bound on the second moment, we again start with the definition of **InvSqrt**,

$$\begin{aligned}\mathbb{E}\left|g\right|^{2} &= \mathbb{E}\left(\sum_{j=0}^{k-1}\frac{1}{j!}g^{(j)}(x)\prod_{\ell=1}^{j}\left(x_{j,\ell}-x\right)\right)^{2}\\ &\overset{①}{\leq} k\mathbb{E}\sum_{j=0}^{k-1}\left(\frac{g^{(j)}(x)}{j!}\prod_{\ell=1}^{j}\left(x_{j,\ell}-x\right)\right)^{2}\\ &\overset{②}{=} k\sum_{j=0}^{k-1}\mathbb{E}\left(\left(\frac{g^{(j)}(x)}{j!}\right)^{2}\left(\sigma^{2}+(x-\mu)^{2}\right)^{j}\right)\\ &\overset{③}{\leq} k\sum_{j=0}^{k-1}\mathbb{E}\left(\frac{\left(\sigma^{2}+(x-\mu)^{2}\right)^{j}}{x^{2j+1}}\right).\end{aligned} \tag{A.4.15}$$

Here ① is by Cauchy-Schwarz inequality; ② is by using the fact that each $x_{j,\ell}$ is sampled independently and adding and subtracting μ from the term inside the square and using the definition of σ^2; ③ uses Inequality A.4.13. □

Lemma 2.2.7. *Given $u \in \mathbb{R}^n$ such that $\mu \stackrel{def}{=} \|u\|_2^2 \neq 0$, and positive integers $k > 1$ and $N \geq 4k + 6$, the following are true for x sampled from $X =$ **RandProj**(u, N).*

(1) $\mathbb{E}x = \mu$

(2) $\sigma^2 \stackrel{def}{=} \mathbb{E}\,(x-\mu)^2 = \frac{2\mu^2}{N}$

(3) $\mathbb{E}\left(\frac{\left(\sigma^2+(x-\mu)^2\right)^k}{\min(x,\mu)^{2k+1}}\right) \leq \frac{1}{\mu}\left(\frac{e^{N/2}}{2^{N-17k}} + \frac{2^{13k}k^{2k}}{N^k}\right)$

Before diving into this proof, we state below a tool we need about logconcave distributions.

Theorem A.4.20 (Theorem 5.22 in [LV07])**.** *If $X \in \mathbb{R}^n$ is a random point sampled from a logconcave distribution, then $(\mathbb{E}|X|^k)^{1/k} \leq 2k\mathbb{E}|X|$.*

Proof of Lemma 2.2.7. By linearity of the Gaussian distribution, given a $\zeta \sim \mathcal{N}(0, I_n)$ and for some $u \in \mathbb{R}^n$, we have $\zeta^T u \sim \mathcal{N}(0, \|u\|_2^2)$. Therefore **RandProj**$(u, N)$ gives us a scaled chi-squared distribution, $X = \frac{\mu}{N}\chi^2_N$. For a point $x \sim X$, using the parameters of a standard chi-squared distribution gives us the following properties.

$$\mathbb{E}x = \frac{\mu}{N} \cdot N = \mu, \text{ and } \mathbf{Var}\,x = \left(\frac{\mu}{N}\right)^2 N(N+2) - \mu^2 = 2\frac{\mu^2}{N}, \tag{A.4.16}$$

which proves **(1)** and **(2)**. To prove **(3)**, we first scale the random variable x by N/μ to make it of a standard chi-squared distribution; this makes our computations easier, since we later need to use the closed-form expression of the probability density function of x. After the scaling, we have

$$\mathbb{E}_{x\sim\chi^2_N} x = N \qquad \mathop{\mathbf{Var}}_{x\sim\chi^2_N} = 2N. \tag{A.4.17}$$

Therefore,

$$\begin{aligned}\mathbb{E}_{x\sim X}\left(\frac{\left(\sigma^2+(\mu-x)^2\right)^k}{\min(x,\mu)^{2k+1}}\right) &\overset{①}{\leq} 2^k\mathbb{E}_{x\sim X}\left(\frac{\sigma^{2k}+(\mu-x)^{2k}}{\min(x,\mu)^{2k+1}}\right)\\ &\overset{②}{=} 2^k\frac{N}{\mu}\underbrace{\mathbb{E}_{x\sim\chi^2_N}\left(\frac{(2N)^k+(N-x)^{2k}}{\min(x,N)^{2k+1}}\right)}_{Ⓐ}.\end{aligned} \tag{A.4.18}$$

Here ① follows from Jensen's inequality applied to the function $g(x) = x^k$ for $k > 1$ and $x > 0$; the equation ② follows from Equation (A.4.16). We now bound Ⓐ by considering the random

variable in two disjoint intervals as follows.

$$\begin{aligned}
\text{(A)} &= \mathbb{E}_{x\sim\chi^2_N}\left(\frac{(2N)^k+(N-x)^{2k}}{\min(x,N)^{2k+1}}\mathbf{1}_{\{x<\frac{N}{4}\}}\right)+\mathbb{E}_{x\sim\chi^2_N}\left(\frac{(2N)^k+(N-x)^{2k}}{\min(x,N)^{2k+1}}\mathbf{1}_{\{x\geq\frac{N}{4}\}}\right).\\
&\leq \underbrace{\mathbb{E}_{x\sim\chi^2_N}\left(\frac{(2N)^k+(N-x)^{2k}}{x^{2k+1}}\mathbf{1}_{\{x<\frac{N}{4}\}}\right)}_{\text{(B)}}+\frac{1}{(N/4)^{2k+1}}\underbrace{\mathbb{E}_{x\sim\chi^2_N}\left((2N)^k+(N-x)^{2k}\right)}_{\text{(C)}}. \qquad \text{(A.4.19)}
\end{aligned}$$

To bound (B), we divide the region $\{x < N/4\}$ into intervals of geometrically-varying lengths as follows.

$$\begin{aligned}
\text{(B)} &= \sum_{j=2}^{\infty}\mathbb{E}_{x\sim\chi^2_N}\left(\frac{(2N)^k+(N-x)^{2k}}{x^{2k+1}}\mathbf{1}_{\left\{\frac{N}{2^{j+1}}\leq x<\frac{N}{2^j}\right\}}\right)\\
&\leq \sum_{j=2}^{\infty}\frac{N^{2k}5^k}{(N/2^{j+1})^{2k+1}}\underbrace{\textbf{Prob}\left(x<N/2^j\right)}_{\text{(D)}}, \qquad \text{(A.4.20)}
\end{aligned}$$

where the inequality follows from the worst case upper bounds for the numerator and $1+2^k \leq 5^k$ for $k \geq 1$ and the worst case lower bounds for the denominator over each interval $\{N/2^{j+1} \leq x < N/2^j\}$. For $a > 0$ and a random variable $x \sim \chi^2_N$, we have the following cumulative distribution function:

$$\begin{aligned}
\textbf{Prob}\left(x\leq a\right) &= \int_0^a \frac{e^{-x/2}x^{N/2-1}}{2^{(N/2)}\Gamma\left(N/2\right)}dx\\
&\leq \int_0^a \frac{e^{-x/2}x^{N/2-1}}{2^{N/2}(N/2e)^{(N-1)/2}}dx\\
&\leq \frac{2a^{N/2-1}e^{N/2}}{N^{(N-1)/2}},
\end{aligned}$$

where we used the Sterling approximation of Gamma function in the second inequality. Substituting $a = 2^{-j}N$ above and simplifying gives the following bound on the quantity from Equation (A.4.20).

$$\text{(D)} \leq \frac{2^{j+1}}{\sqrt{N}}\left(\frac{e}{2^j}\right)^{\frac{N}{2}}. \qquad \text{(A.4.21)}$$

Substitute into Equation (A.4.20) to get

$$\begin{aligned}
\textcircled{B} &\le \sum_{j=2}^{\infty} N^{2k} 5^k \left(\frac{2^{j+1}}{N}\right)^{2k+1} \frac{2^{j+1}}{\sqrt{N}} \left(\frac{e}{2^j}\right)^{N/2} \\
&= \frac{5^k 2^{2k+2} e^{N/2}}{N^{3/2}} \sum_{j=2}^{\infty} \frac{1}{2^{j(N/2-2k-2)}} \\
&\le \frac{2^{5k+2} e^{N/2}}{N^{3/2}} \frac{2}{2^{N-4k-4}} \\
&\le \frac{e^{N/2}}{N^{3/2} 2^{N-9k-7}},
\end{aligned} \tag{A.4.22}$$

where we used the condition that $N \ge 4k+6$ in the first two inequalities. Next, we bound $\textcircled{C}$.

$$\begin{aligned}
\textcircled{C} &= (2N)^k \left(\mathbb{E}\left|\frac{x-N}{\sqrt{2N}}\right|^{2k} + 1\right) \\
&\overset{\textcircled{1}}{\le} (2N)^k \left(2^{2k} (2k)^{2k} \left(\mathbb{E}\frac{|x-N|}{\sqrt{2N}}\right)^{2k} + 1\right) \\
&\overset{\textcircled{2}}{\le} (2N)^k \left(2^{2k} (2k)^{2k} \left(\frac{\sqrt{\mathbb{E}|x-N|^2}}{\sqrt{2N}}\right)^{2k} + 1\right) \\
&= (2N)^k \left(2^{2k} (2k)^{2k} + 1\right) \\
&\le (2N)^k \left(32k^2\right)^k,
\end{aligned} \tag{A.4.23}$$

where $\textcircled{1}$ is by invoking Theorem A.4.20, which is valid by logconcavity of chi-squared distribution, and $\textcircled{2}$ is by Jensen's inequality. Plugging Inequality A.4.22 and Inequality A.4.23 into Equation (A.4.18) gives:

$$\begin{aligned}
\mathbb{E}_{x\sim X}\left(\frac{\left(\sigma^2 + (x-\mu)^2\right)^k}{\min(x,\mu)^{2k+1}}\right) &\le 2^k \frac{N}{\mu}\left(\frac{e^{N/2}}{N^{3/2}2^{N-9k-7}} + \frac{4^{2k+1}}{N^{2k+1}} (2N)^k \left(32k^2\right)^k\right) \\
&\le \frac{1}{\mu}\left(\frac{e^{N/2}}{2^{N-17k}} + \frac{2^{13k}k^{2k}}{N^k}\right),
\end{aligned}$$

which is what is to be proved. □

Properties of $\widehat{\theta}_1$

We prove the bounds on first and second moments of $\widehat{\theta}_1$. Note that this is where we make our choice of $\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}$ and $\mathrm{T}_{\mathrm{est}_{\mathrm{jl}}}$ for the modules **InvSqrt** and **RandProj** used in estimating θ_1 in the subroutine **Estimator1**.

Lemma 2.2.4. *Given* $\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}} = 1600\log(n/\varepsilon)$, $\mathrm{T}_{\mathrm{est}_{\mathrm{jl}}} = 2^{14}\mathrm{T}^2_{\mathrm{est}_{\mathrm{isq}}}$, $Z \in \mathbb{S}^n$, *and* $\varepsilon \in (0, 1/2)$, *let* $\widetilde{Z^2} =$ ***RandProj***$(Z, \mathrm{T}_{\mathrm{est}_{\mathrm{jl}}})$ *and* $\widehat{\theta}_{1_i} \sim$ ***InvSqrt***$((\widetilde{Z^2})_{ii} + 1, \mathrm{T}_{\mathrm{est}_{\mathrm{isq}}})$ *for* $i \in [n]$. *Then,*

(1) *The first moment satisfies* $\left|\mathbb{E}\widehat{\theta}_{1_i} - \frac{1}{\sqrt{(Z^2)_{ii}+1}}\right| \le \frac{\sqrt{2}(\varepsilon/n)^{400}}{\sqrt{(Z^2)_{ii}+1}}$.

(2) *The second moment satisfies* $\mathbb{E}|\widehat{\theta}_{1_i}|^2 \le \frac{1}{(Z^2)_{ii}} 1630 \log(n/\varepsilon)$.

Proof. Consider a random variable x sampled from the distribution $(\widetilde{Z^2})_{ii}$. Because of Lemma 2.2.7, we have $\mathbb{E}x = (Z^2)_{ii}$. Then $x+1$ satisfies the required bias condition of Lemma 2.2.6 for constructing a polynomial approximation for $1/\sqrt{1+(Z^2)_{ii}}$. Then $\widehat{\theta}_{1_i}$ satisfies

$$\begin{aligned}\left|\mathbb{E}\widehat{\theta}_{1_i} - \frac{1}{\sqrt{1+(Z^2)_{ii}}}\right| &\overset{①}{\le} \mathbb{E}\left(\frac{\left|x-(Z^2)_{ii}\right|^{\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}}}{\min(x+1,(Z^2)_{ii}+1)^{\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}+1/2}}\right)\\ &\overset{②}{\le} \sqrt{\mathbb{E}\frac{(x-(Z^2)_{ii})^{2\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}}}{\min(x+1,(Z^2)_{ii}+1)^{2\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}+1}}}\\ &\overset{③}{\le} \sqrt{\frac{1}{(Z^2)_{ii}+1}\left(\frac{e^{\mathrm{T}_{\mathrm{est}_{\mathrm{jl}}}/2}}{2^{\mathrm{T}_{\mathrm{est}_{\mathrm{jl}}}-17\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}}} + \frac{2^{13\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}}\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}^{2\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}}}{\mathrm{T}_{\mathrm{est}_{\mathrm{jl}}}^{\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}}}\right)}.\end{aligned}$$

where ① is by Lemma 2.2.6, ② is by Jensen's inequality, and ③ is by a slight modification of the proof of (3) in Lemma 2.2.7 (instead of scaling by N/μ, we scale by $N\mu/(\mu+1)$ in the proof). Finally, set $\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}} = 1600\log\left(\frac{n}{\varepsilon}\right)$ and $\mathrm{T}_{\mathrm{est}_{\mathrm{jl}}} = 2^{14}\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}^2$ to get the claimed bias. Next, we can bound the variance as follows.

$$\begin{aligned}\mathbb{E}|\widehat{\theta}_{1_i}|^2 &\overset{①}{\le} \mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}\sum_{k=0}^{\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}-1}\mathbb{E}\left(\frac{\left(\sigma^2+\left(x-(Z^2)_{ii}\right)^2\right)^k}{(x+1)^{2k+1}}\right)\\ &\le \mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}\sum_{k=0}^{\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}-1}\mathbb{E}\left(\frac{\left(\sigma^2+\left(x-(Z^2)_{ii}\right)^2\right)^k}{\min(x+1,(Z^2)_{ii}+1)^{2k+1}}\right)\\ &\overset{②}{\le} \frac{\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}}{(Z^2)_{ii}}\sum_{k=0}^{\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}-1}\left(\frac{e^{\mathrm{T}_{\mathrm{est}_{\mathrm{jl}}}/2}}{2^{\mathrm{T}_{\mathrm{est}_{\mathrm{jl}}}-17k}} + \frac{2^{13k}k^{2k}}{\mathrm{T}_{\mathrm{est}_{\mathrm{jl}}}^k}\right)\\ &\overset{③}{=} \frac{\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}}{(Z^2)_{ii}}\sum_{k=0}^{\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}-1}\left(2^{17k}\left(\frac{\sqrt{e}}{2}\right)^{2^{14}\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}^2} + \frac{k^{2k}}{2^k\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}^{2k}}\right)\end{aligned}$$

where ① is by (2) in Lemma 2.2.6, ② is by (3) in Lemma 2.2.7, and ③ is by writing $\mathrm{T}_{\mathrm{est}_{\mathrm{jl}}}$ in terms of $\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}$. We have the simplications, $\sum_{k=0}^{\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}-1} 2^{17k}\left(\frac{\sqrt{e}}{2}\right)^{2^{14}\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}^2} \le \frac{2^{17\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}}}{1.2^{2^{14}\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}}2^{16}}$ and

$$\sum_{k=0}^{\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}-1}\left(\frac{k^2}{2\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}^2}\right)^k \le 1 + \frac{1}{2\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}^2} + \frac{4}{\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}^4} + \sum_{k=3}^{\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}/2}\left(\frac{k^2}{2\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}^2}\right)^k + \sum_{k>\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}/2}\left(\frac{k^2}{2\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}^2}\right)^k.$$

Finally, plug in the values of $\mathrm{T}_{\mathrm{est}_{\mathrm{isq}}}$ to get the desired bound. □

In Algorithm 2.2.1, we construct the matrix Z as an approximation to $\exp\left(\frac{1}{2}\left(Y^{(t)}+s\Delta\right)\right)$ by the subroutine $\mathbf{ChebyExp}\left(\frac{1}{2}\left(Y^{(t)}+s\Delta\right), T_{\mathrm{Cheby}}, \delta_{\mathrm{Cheby}}\right)$, with details as provided in Lemma A.4.19. With this value of Z and the same rest of the notation as in the above lemma, we therefore wish to compare $\mathbb{E}\widehat{\theta}_{1_i}$ with $\frac{1}{\sqrt{\exp\left(Y^{(t-1)}+s\Delta\right)_{ii}+1}}$. Note that the above lemma only tells us that we are close to $\frac{1}{\sqrt{(Z^2)_{ii}+1}}$, but Z, as defined above in Lemma A.4.19, is only an approximation to $\exp\left(\frac{1}{2}\left(Y^{(t-1)}+s\Delta\right)\right)$. We therefore obtain the following corollary which gives us a precise bound on the bias we care about.

Corollary A.4.21 (Bias of $\widehat{\theta}_{1_i}$)**.** *The estimator $\widehat{\theta}_{1_i}$ described in Algorithm 2.2.2 satisfies*

$$\left|\mathbb{E}\widehat{\theta}_{1_i}-\frac{1}{\sqrt{\exp\left(Y^{(t-1)}+s\Delta\right)_{ii}+1}}\right| \leq b_{1_i} \stackrel{def}{=} \frac{(1+2\delta_{\exp})\sqrt{2}(\frac{\varepsilon}{n})^{400}+2\delta_{\exp}}{\sqrt{\exp\left(Y^{(t-1)}+s\Delta\right)_{ii}+1}},$$

where $\delta_{\exp}=4800\frac{\varepsilon^{401}}{n^{390}}$.

Proof. From Lemma A.4.19, we know that $Z=\mathbf{ChebyExp}\left(\frac{1}{2}\left(Y^{(t-1)}+s\Delta\right), \mathrm{T}_{\mathrm{Cheby}}, \delta_{\mathrm{Cheby}}\right)$ satisfies

$$\left|\left(\exp\left(Y^{(t-1)+s\Delta}\right)-Z^2\right)_{ii}\right| \leq \frac{4800\varepsilon^{401}}{n^{390}}.$$

For ease of notation, let $\delta_{\exp} \stackrel{\mathrm{def}}{=} \frac{4800\varepsilon^{401}}{n^{390}}$. Given $a-\delta \leq b \leq a+\delta$, we use the Taylor series approximation to compute the error $\frac{1}{\sqrt{a}}-\frac{1}{\sqrt{b}}$. We have:

$$\begin{aligned}\left|\frac{1}{\sqrt{a}}-\frac{1}{\sqrt{b}}\right| &\leq \left|\frac{1}{\sqrt{a}}-\frac{1}{\sqrt{-\delta+a}}\right| \\ &= \frac{1}{\sqrt{a}}\left|1-\frac{1}{\sqrt{1-\delta/a}}\right| \\ &\leq \frac{1}{\sqrt{a}}\frac{2\delta}{a}=\frac{2\delta}{a^{3/2}},\end{aligned}$$

where we used the Taylor approximation of $\frac{1}{\sqrt{1-x}}$ for small x. Thus, we have, from the above and Lemma 2.2.4,

$$\begin{aligned}\left|\mathbb{E}\widehat{\theta}_{1_i}-\frac{1}{\sqrt{\exp\left(Y^{(t-1)}+s\Delta\right)_{ii}+1}}\right| &\leq \frac{\sqrt{2}(\varepsilon/n)^{400}}{\sqrt{Z_{ii}^2+1}}+\frac{2\delta}{\sqrt{\exp\left(Y^{(t-1)}+s\Delta\right)_{ii}+1}} \\ &\leq \frac{(1+2\delta)\sqrt{2}(\varepsilon/n)^{400}+2\delta}{\sqrt{\exp\left(Y^{(t-1)}+s\Delta\right)_{ii}+1}},\end{aligned}$$

which proves the claim. □

Properties of $\widehat{\theta}_2$

Lemma 2.2.5. *Consider Z_1, Z_2, Z, and Δ all in $\mathbb{S}^n$. Sample $\zeta \sim \mathcal{N}(\mathbf{0}, I_n)$, and define $\widehat{\theta}_2 \in \mathbb{R}^n$ as $\widehat{\theta}_{2_i} = (Z_1 \Delta Z_2 \zeta)_i (Z\zeta)_i$. Define $\theta_{2_i} \stackrel{def}{=} (Z_1 \Delta Z_2 Z)_{ii}$. Then for $i \in [n]$:*

(1) *The first moment satisfies $\mathbb{E}\widehat{\theta}_{2_i} = \theta_{2_i}$*
(2) *The second moment satisfies $\mathbb{E}|\widehat{\theta}_{2_i}|^2 \leq 3\left(Z_1 \Delta Z_2^2 \Delta Z_1\right)_{ii}\left(Z^2\right)_{ii}$.*

Proof. The bias is defined as

$$\begin{aligned}\mathbb{E}\widehat{\theta}_{2_i} &= \mathbf{1}_i^T Z_1 \Delta Z_2 \left(\mathbb{E}\zeta\zeta^T\right) Z\mathbf{1}_i \\ &= (Z_1 \Delta Z_2 Z)_{ii} = \theta_{2_i},\end{aligned}$$

where the second step is from the fact that $\zeta \sim \mathcal{N}(0, I_n)$ and linearity of expectation, and the last is by definition of θ_2. Next, from Lemma A.5.1, given $a, b \in \mathbb{R}^n$ and $\zeta \sim \mathcal{N}(0, I_n)$, we conclude that $\mathbb{E}((\zeta^T a)^2(\zeta^T b)^2) \leq 3\|a\|_2^2\|b\|_2^2$. Therefore,

$$\begin{aligned}\mathbb{E}\left|\widehat{\theta}_{2_i}\right|^2 &= \mathbb{E}(\mathbf{1}_i^T Z_1 \Delta Z_2 \zeta)^2)(\zeta^T Z\mathbf{1}_i)^2 \\ &\leq 3\|Z_2 \Delta Z_1 \mathbf{1}_i\|^2 \|Z\mathbf{1}_i\|^2 \\ &= 3(Z_1 \Delta Z_2^2 \Delta Z_1)_{ii}(Z^2)_{ii}.\end{aligned}$$

This proves the bound on the second moment. □

As before, we can obtain, as a corollary of this result, a comparison of the mean of our estimator with the quantity we actually are trying to compute.

Corollary A.4.22 (Bias of $\widehat{\theta}_{2_i}$)**.** *The estimator $\widehat{\theta}_{2_i}$ described in Algorithm 2.2.2 satisfies*

$$\left|\mathbb{E}\widehat{\theta}_{2_i} - \left(\exp\left(\bar{\tau}(Y^{(t-1)} + s\Delta)\right)\Delta \exp\left((\tau - \tfrac{1}{2})(Y^{(t-1)} + s\Delta)\right)\exp\left(\tfrac{1}{2}(Y^{(t-1)} + s\Delta)\right)\right)_{ii}\right| \leq 15\delta_{\exp}\eta K$$

where $\delta_{\exp} = \frac{4800\varepsilon^{401}}{n^{390}}$.

Proof. This proof simply involves writing out some matrix products and bounds on the diagonal entries of the products (using the operator norm of the individual matrices). We show this below. Let $Z_1 = \exp\left(\bar{\tau}\left(Y^{(t-1)} + s\Delta\right)\right) + U_1$, $Z_2 = \exp\left((\tau - 1/2)\left(Y^{(t-1)} + s\Delta\right)\right) + U_2$, and $Z = \exp\left(\frac{1}{2}\left(Y^{(t-1)} + s\Delta\right)\right) + U$. From Lemma 2.2.5, we have that $\mathbb{E}\widehat{\theta}_{2_i} = \theta_{2_i}$. We now express θ_{2_i} in terms of the matrix exponentials we care about. For ease of notation, we use $Y_s = Y^{(t-1)} + s\Delta$.

$$\begin{aligned}\mathbb{E}\widehat{\theta}_{2_i} - \left(\exp\left(\bar{\tau}Y_s\right)\Delta \exp\left((\tau - 1/2)Y_s\right)\exp\left(\tfrac{1}{2}Y_s\right)\right)_{ii} &= \left(\exp\left(\bar{\tau}Y_s\right)\Delta \exp\left((\tau - 1/2)Y_s\right)U\right)_{ii} \\ &+ \left(\exp\left(\bar{\tau}Y_s\right)\Delta U_2 \exp\left(\tfrac{1}{2}Y_s\right)\right)_{ii} + \left(\exp\left(\bar{\tau}Y_s\right)\Delta U_2 U\right)_{ii} \\ &+ \left(U_1 \Delta \exp\left((\tau - 1/2)Y_s\right)\exp\left(\tfrac{1}{2}Y_s\right)\right)_{ii} \\ &+ \left(U_1 \Delta \exp\left((\tau - 1/2)Y_s\right)U\right)_{ii} \\ &+ \left(U_1 \Delta U_2 \exp\left(\tfrac{1}{2}Y_s\right)\right)_{ii} + \left(U_1 \Delta U_2 U\right)_{ii}.\end{aligned}$$

We can bound this by bounding the operator norm of each of the terms. Matrix norm is sub-multiplicative, so this in turn is bounded by the operator norm of the individual terms in the matrices. From Equation (A.4.10), we know that $\|\exp(\alpha Y_s)\|_{\text{op}} \le K^\alpha$, $\|\Delta\|_{\text{op}} \le \eta G$, $\|U_1\|_{\text{op}} \le \delta_{\text{exp}}$, $\|U_2\|_{\text{op}} \le \delta_{\text{exp}}$, and $\|U\|_{\text{op}} \le \delta_{\text{exp}}$, where $\delta_{\text{exp}} = \frac{4800\varepsilon^{401}}{n^{390}}$. Substituting these values here and bounding each term by the largest of all terms gives us the bound to be proved. □

Properties of the Overall Estimator, $\widehat{\theta}$

Lemma 2.2.3. *The estimator $\widehat{\theta}^{(t)}$ has the following bounds on its first and second moments.*

(1) $|\mathbb{E}\widehat{\theta}_i - \int_{s=0}^1 \int_{\tau=0}^1 \theta_{1_i}\theta_{2_i} ds d\tau| \le b_{1_i}\theta_{2_i} + b_{2_i}\theta_{1_i} + b_{1_i}b_{2_i}$ *for* $i \in [n]$.
(2) $\mathbb{E}\|\widehat{\theta}\|_2^2 \le 19600\log(n/\varepsilon)K\eta^2 + 147000K^2\eta^2\delta_{\text{exp}}$.

Proof. We can get the bound on the bias by applying the results of Corollary A.4.21 and Corollary A.4.22 in $\mathbb{E}\widehat{\theta}_i = \mathbb{E}\widehat{\theta}_{1_i}\mathbb{E}\widehat{\theta}_{2_i}$. We need the following definition to concisely write out expressions in this proof.

Definition A.4.23. *Let* $\theta_{1_i} = \frac{1}{\sqrt{\exp(Y_s)_{ii}+1}}$, $\theta_{2_i} = \frac{1}{2}\left(\exp(\bar{\tau}Y_s)\,\Delta\exp\left((\tau-\frac{1}{2})Y_s\right)\exp\left(\frac{1}{2}Y_s\right)\right)_{ii}$, $b_{1_i} = \theta_{1_i}(2\delta_{\text{exp}} + (1+2\delta_{\text{exp}})\sqrt{2}(\varepsilon/n)^{400})$, *and* $b_{2_i} = 15\delta_{\text{exp}}\eta K$ *for* $Y_s = Y^{(t-1)} + s\Delta$.

We have the following error bound.

$$\begin{aligned}\left|\mathbb{E}\widehat{\theta}_i - \int_{s=0}^1 \theta_{1_i}\int_{\tau=0}^1 \theta_{2_i} d\tau ds\right| &= \left|\int_{s=0}^1 \mathbb{E}\widehat{\theta}_{1_i}\int_{\tau=0}^1 \mathbb{E}\widehat{\theta}_{2_i} d\tau ds - \int_{s=0}^1 \theta_{1_i}\int_{\tau=0}^1 \theta_{2_i} d\tau ds\right| \\ &\le \int_{s=0}^1\int_{\tau=0}^1 \left|\mathbb{E}\widehat{\theta}_{1_i}\mathbb{E}\widehat{\theta}_{2_i} - \theta_{1_i}\theta_{2_i}\right| d\tau ds \\ &\le \left|\mathbb{E}\widehat{\theta}_{1_i}\mathbb{E}\widehat{\theta}_{2_i} - \theta_{1_i}\theta_{2_i}\right|.\end{aligned}$$

From Corollary A.4.21, we have $\mathbb{E}\widehat{\theta}_{1_i} \in [\theta_{1_i} \pm b_{1_i}]$. From Corollary A.4.22, we have $\mathbb{E}\widehat{\theta}_{2_i} \in [\theta_{2_i} \pm b_{2_i}]$. Therefore, the right hand side above is bounded by:

$$\left|\mathbb{E}\widehat{\theta}_i - \int_{s=0}^1 \theta_{1_i}\int_{\tau=0}^1 \theta_{2_i} ds d\tau\right| \le b_{1_i}\theta_{2_i} + b_{2_i}\theta_{1_i} + b_{1_i}b_{2_i}.$$

We now compute a quantity which will be useful later:

$$\sum_{i=1}^n \left(\mathbb{E}\widehat{\theta}_i - \int_{s=0}^1 \theta_{1_i}\int_{\tau=0}^1 \theta_{2_i} ds d\tau\right)^2 \le b_{1_i}^2 \sum_{i=1}^n \theta_{2_i}^2 + (2b_{1_i}b_{2_i})(1+b_{1_i})\sum_{i=1}^n \theta_{2_i} + nb_{2_i}^2(1+b_{1_i})^2. \quad \text{(A.4.24)}$$

Here we used the fact that $\theta_{1_i} = \frac{1}{\sqrt{\exp(Y_s)_{ii}+1}} \leq 1$. We compute each of these terms separately.

$$\begin{aligned}
\sum_{i=1}^{n} \theta_{2_i}^2 &= \sum_{i=1}^{n} \left(\left(\exp\left(\bar{\tau} Y_s\right) \Delta \exp\left((\tau - 1/2) Y_s\right) \exp\left(\tfrac{1}{2} Y_s\right) \right)_{ii} \right)^2 \\
&\overset{\text{(A)}}{\leq} \sum_{i=1}^{n} \left(\exp\left(\bar{\tau} Y_s\right) \Delta \exp\left((\tau - 1/2) Y_s\right) \exp\left(\tfrac{1}{2} Y_s\right) \exp\left(\bar{\tau} Y_s\right) \Delta \exp\left((\tau - 1/2) Y_s\right) \exp\left(\tfrac{1}{2} Y_s\right) \right)_{ii} \\
&= \operatorname{Tr}\left(\exp\left(\bar{\tau} Y_s\right) \Delta \exp\left(Y_s\right) \Delta \exp\left(\tau Y_s\right) \right) \\
&= \operatorname{Tr}\left(\exp\left(Y_s\right) \Delta \exp\left(Y_s\right) \Delta \right) \\
&\leq K^2 \eta^2 G^2. \qquad \text{(A.4.25)}
\end{aligned}$$

Here, (A) was because $\sum_{i=1}^{n} (A_{ii})^2 \leq \sum_{i=1}^{n} (A^2)_{ii}$, which can be checked by a simple computation. Similarly, the sum in the cross-term can be computed as follows.

$$\begin{aligned}
\sum_{i=1}^{n} \theta_{2_i} &= \sum_{i=1}^{n} \left(\exp\left(\bar{\tau} Y_s\right) \Delta \exp\left((\tau - 1/2) Y_s\right) \exp\left(\tfrac{1}{2} Y_s\right) \right)_{ii} \\
&= \operatorname{Tr}\left(\exp\left(\bar{\tau} Y_s\right) \Delta \exp\left((\tau - 1/2) Y_s\right) \exp\left(\tfrac{1}{2} Y_s\right) \right) \\
&= \operatorname{Tr}\left(\exp\left(\bar{\tau} Y_s\right) \Delta \exp\left(\tau Y_s\right) \right) \\
&= \operatorname{Tr}\left(\exp\left(Y_s\right) \Delta \right) \\
&\leq K \eta G. \qquad \text{(A.4.26)}
\end{aligned}$$

Substituting Equation (A.4.25) and Equation (A.4.26) into Equation (A.4.24), and using $\frac{1}{\sqrt{\exp(Y_s)_{ii}+1}} \leq 1$ gives us:

$$\begin{aligned}
\sum_{i=1}^{n} \left(\mathbb{E}\widehat{\theta_i} - \int_{s=0}^{1} a_1 \int_{\tau=0}^{1} a_2 \, ds \, d\tau \right)^2 &\leq (2\delta_{\exp} + (1 + 2\delta_{\exp}) \sqrt{2}(\varepsilon/n)^{400})^2 K^2 \eta^2 G^2 \\
&\quad + 900 n \delta^2 \eta^2 K^2 \\
&\quad + 60 \eta \delta K (2\delta_{\exp} + (1 + 2\delta_{\exp}) \sqrt{2}(\varepsilon/n)^{400}) K \eta G \\
&\leq 6 K^2 \eta^2 (\sqrt{2}(\varepsilon/n)^{400} + 2\delta_{\exp}) \\
&\leq 400 n K^2 \eta^2 (\sqrt{2}(\varepsilon/n)^{400} + 2\delta_{\exp}). \qquad \text{(A.4.27)}
\end{aligned}$$

We now prove the final variance bound.

$$\begin{aligned}
\mathbb{E}_{s,\tau,\zeta_1,\zeta_2} \|\widehat{\theta}\|_2^2 &= \mathbb{E}_{s,\tau,\zeta_1,\zeta_2} \sum_{i=1}^{n} |\widehat{\theta_i}|^2 \\
&= \int_{s=0}^{1} \int_{\tau=0}^{1} \sum_{i=1}^{n} \mathbb{E}_{\zeta_1} |\widehat{\theta_{1_i}}|^2 \mathbb{E}_{\zeta_2} |\widehat{\theta_{2_i}}|^2 \, ds \, d\tau.
\end{aligned}$$

Combining Lemma 2.2.4 and Lemma 2.2.5, we get:

$$\mathbb{E}_{s,\tau,\zeta_1,\zeta_2}\|\widehat{\theta}\|_2^2 = \int_{s=0}^{1}\int_{\tau=0}^{1}\sum_{i=1}^{n}\underbrace{\frac{1630\log(n/\varepsilon)}{(Z^2)_{ii}}}_{①}\cdot 3\underbrace{\left(Z_2\Delta Z_1^2\Delta Z_2\right)_{ii}\left(Z^2\right)_{ii}}_{②}dsd\tau,$$

$$\overset{Ⓐ}{=}\sum_{i=1}^{n}\int_{s=0}^{1}\int_{\tau=0}^{1}4890\log(n/\varepsilon)(Z_2\Delta Z_1^2\Delta Z_2)_{ii}dsd\tau$$

$$= 4890\log(n/\varepsilon)\int_{s=0}^{1}\int_{\tau=0}^{1}\mathrm{Tr}\left(Z_2^2\Delta Z_1^2\Delta\right)dsd\tau, \tag{A.4.28}$$

where $Z_1 = \exp\left((\tau-1/2)\left(Y^{(t-1)}+s\Delta\right)\right)+U_1$ and $Z_2 = \exp\left(\bar{\tau}\left(Y^{(t-1)}+s\Delta\right)\right)+U_2$ as defined in Corollary A.4.22. The term Ⓐ shows the significance of carefully choosing the split in the estimator $\widehat{\theta}_2$, which enabled the cancellation of $\frac{1}{(Z^2)_{ii}}$ and $(Z^2)_{ii}$. We now bound $\mathrm{Tr}\left(Z_2^2\Delta Z_1^2\Delta\right)$. In Lemma A.4.19 we showed how to construct Z_1 and Z_2 as $\delta_{\exp} = 4800\varepsilon^{401}/n^{390}$ approximations to the respective matrix exponentials. Thus, writing $\|U_1\|_{\mathrm{op}} = \|U_2\|_{\mathrm{op}} = \delta_{\exp}$ and expanding out the product $Z_2^2\Delta Z_1^2\Delta$ in terms of the true matrix exponentials and the error matrices, we get the following:

$$\mathrm{Tr}\Big(Z_2^2\Delta Z_1^2\Delta\Big) \le \mathrm{Tr}\Big(\exp\Big(2\bar{\tau}(Y^{(t-1)}+s\Delta)\Big)\Delta\exp\Big((2\tau-1)(Y^{(t-1)}+s\Delta)\Big)\Delta\Big) + 30\eta^2\delta_{\exp}K^2.$$

Choosing $A = \exp\left(Y^{(t-1)}+s\Delta\right)$ and $B = \Delta$ and combining with the fact that matrix exponential is positive semidefinite, and Δ is a symmetric matrix since the gradient of the objective is symmetric, invoking Fact 2.1.1 gives:

$$\mathrm{Tr}\left(Z_2^2\Delta Z_1^2\Delta\right) \le \mathrm{Tr}\Big(\exp\Big(Y^{(t-1)}+s\Delta\Big)\Delta^2\Big) + 30\eta^2\delta_{\exp}K^2 \le 4K\eta^2 + 30\eta^2\delta_{\exp}K^2,$$

where the last inequality follows from applying Holder's inequality with the nuclear norm and operator norm. Plugging this back into Equation (A.4.28) and completing the integration gives

$$\mathbb{E}_{s,\tau,\zeta_1,\zeta_2}\|\widehat{\theta}\|_2^2 \le 4890\log(n/\varepsilon)\left(4K\eta^2 + 30K^2\eta^2\delta_{\exp}\right) \le 19600\log(n/\varepsilon)K\eta^2 + 147000K^2\eta^2\delta_{\exp}.$$

□

A.4.4 Number of Inner Iterations

We can use the general expression for overall running time to choose a value for number of 'low-accuracy' iterations. The total computational cost of the algorithm is

$$T_{\text{outer}}\times\frac{10^5\,(\log n)^{21}}{\varepsilon^2}T_{\exp} + T_{\text{outer}}\times T_{\text{inner}}\times 2^{30}\left(\log\left(\frac{1}{\varepsilon}\right)\right)^4 T_{\exp}, \tag{A.4.29}$$

where the first term is the total cost of exact computations, and the second term is the total cost of approximate computations (done inside the inner loop); $T_{\exp}$ is the cost of approximating the products of matrix exponentials with a vector. This is optimal (ignoring polylogarithmic terms) when setting $T_{\text{inner}} = \tilde{O}(1/\varepsilon^2)$. We set $T_{\text{inner}} = 1/\varepsilon^2$ due to technical reasons arising in Lemma A.4.24.

A.4.5 Distance Bound Between Estimated and True Iterates

Since the estimators in the inner loop iterations are constructed to have a low variance, the estimated and true iterates aren't far apart, as we show now. This is also where we choose the step size η.

Lemma A.4.24. *In Algorithm 2.2.1, after $t \le T_{inner}$ iterations, we have $\mathbb{E}\|X^{(t)} - \widetilde{X}^{(t)}\|_{\text{nuc}} \le 1.132n\varepsilon$. Recall, $\widetilde{X}^{(t)}$ is the approximate primal iterate, while $X^{(t)}$ is the exact iterate.*

Proof. By the definition of $\|\!|\cdot|\!\|$ and some algebra, we have

$$\begin{aligned}
\mathbb{E}|\!|\!|X^{(t)} - \widetilde{X}^{(t)}|\!|\!| &= \mathbb{E}\sum_{i=1}^{n}\left|X_{ii}^{(t)} - \widetilde{X}_{ii}^{(t)}\right| \\
&= \mathbb{E}\sum_{i=1}^{n}\left|\left(\sqrt{X_{ii}^{(t)}+1}\right)^2 - \left(\sqrt{\widetilde{X}_{ii}^{(t)}+1}\right)^2\right| \\
&= \mathbb{E}\sum_{i=1}^{n} 2\sqrt{X_{ii}^{(t)}+1}\left|\sqrt{X_{ii}^{(t)}+1} - \sqrt{\widetilde{X}_{ii}^{(t)}+1}\right| + \mathbb{E}\sum_{i=1}^{n}\left|\sqrt{X_{ii}^{(t)}+1} - \sqrt{\widetilde{X}_{ii}^{(t)}+1}\right|^2.
\end{aligned}$$

Next, apply Cauchy-Schwarz inequality and Lemma 2.2.8 to get

$$\begin{aligned}
\mathbb{E}|\!|\!|X^{(t)} - \widetilde{X}^{(t)}|\!|\!| &\le 2\mathbb{E}\sqrt{\operatorname{Tr} X^{(t)} + n}\sqrt{\sum_{i=1}^{n}\left(\sqrt{X_{ii}^{(t)}+1} - \sqrt{\widetilde{X}_{ii}^{(t)}+1}\right)^2} + \mathbb{E}\sum_{i=1}^{n}\left(\sqrt{X_{ii}^{(t)}+1} - \sqrt{\widetilde{X}_{ii}^{(t)}+1}\right)^2 \\
&\le 2\sqrt{K+n}\,\underbrace{\mathbb{E}\sqrt{\sum_{i=1}^{n}\left(\sqrt{X_{ii}^{(t)}+1} - \sqrt{\widetilde{X}_{ii}^{(t)}+1}\right)^2}}_{\text{(A)}} + \underbrace{\mathbb{E}\sum_{i=1}^{n}\left(\sqrt{X_{ii}^{(t)}+1} - \sqrt{\widetilde{X}_{ii}^{(t)}+1}\right)^2}_{\text{(B)}}.
\end{aligned} \tag{A.4.30}$$

We first bound (B). We can write a recursive formulation for as follows.

$$\sqrt{\widetilde{X}_{ii}^{(t)}+1} - \sqrt{X_{ii}^{(t)}+1} = \underbrace{\left(\sqrt{\widetilde{X}_{ii}^{(0)}+1} - \sqrt{X_{ii}^{(0)}+1}\right)}_{\text{(C)}} + \underbrace{\sum_{s=1}^{t}\left(\widetilde{\theta}_i^{(s)} - \sqrt{X_{ii}^{(s)}+1} + \sqrt{X_{ii}^{(s-1)}+1}\right)}_{\text{(D)}}.$$

We invoke Johnson-Lindenstrauss lemma (restated in Lemma A.3.10 for completeness) and choose the accuracy parameter for it to be such that $\left|X_{ii}^{(0)} - \widetilde{X}_{ii}^{(0)}\right| \le \widetilde{\varepsilon}X_{ii}^{(0)} = \frac{\varepsilon}{100(\log n)^{10}}X_{ii}^{(0)}$. Therefore,

$\text{C} \le \frac{\widetilde{\varepsilon}}{2}\sqrt{X^{(0)}_{ii}+1} = \frac{\varepsilon}{200(\log n)^{10}}\sqrt{X^{(0)}_{ii}+1}$. Summing over all indices and taking expectations gives

$$
\begin{aligned}
\text{B} &\le \mathbb{E}\sum_{i=1}^{n}\left(\frac{\varepsilon}{200\,(\log n)^{10}}\sqrt{X^{(0)}_{ii}+1} + \sum_{s=1}^{t}\left(\widehat{\theta}^{(s)}_i - \sqrt{X^{(s)}_{ii}+1} + \sqrt{X^{(s-1)}_{ii}+1}\right)\right)^2 \\
&\overset{1}{\le} 2\frac{\varepsilon^2}{40000\,(\log n)^{20}}(\operatorname{Tr} X^{(0)} + n) + 2\mathbb{E}\|\sum_{s=1}^{t}\left(\widehat{\theta}^{(s)} - \sqrt{\mathbf{diag}(())\,X^{(s)}+\mathbf{1}} + \sqrt{\mathbf{diag}(())\,X^{(s-1)}+\mathbf{1}}\right)\|_2^2 \\
&\overset{2}{\le} \frac{K\varepsilon^2}{10000\,(\log n)^{20}} + 2\underbrace{\mathbb{E}\|\sum_{s=1}^{t}\left(\widehat{\theta}^{(s)} - \sqrt{\mathbf{diag}(())\,X^{(s)}+\mathbf{1}} + \sqrt{\mathbf{diag}(())\,X^{(s-1)}+\mathbf{1}}\right)\|_2^2}_{\text{E}},
\end{aligned}
$$

where 1 is by Cauchy-Schwarz inequality, and 2 by Lemma 2.2.8. A subtle point here is that even though the very first iterate in the algorithm satisfies a stronger inequality, namely, $\operatorname{Tr} X^{(0)} \le n$, we *cannot* use this stronger bound because we care about *all* iterations, and this stronger bound doesn't hold later on. We now bound E below. Note that since the random variable $\widehat{\theta}^{(s)}$ is not entirely unbiased, the term E is not the variance. Let $\theta^{(s)} \overset{\text{def}}{=} \mathbb{E}\widehat{\theta}^{(s)}$ and $d^{(s)} = \sqrt{\mathbf{diag}(())\,X^{(s)}+\mathbf{1}} - \sqrt{\mathbf{diag}(())\,X^{(s-1)}+\mathbf{1}}$. Then,

$$
\begin{aligned}
\text{E} &= \mathbb{E}\|\sum_{s=1}^{t}\left(\widehat{\theta}^{(s)} - \left(\sqrt{\mathbf{diag}(())\,X^{(s)}+\mathbf{1}} - \sqrt{\mathbf{diag}(())\,X^{(s-1)}+\mathbf{1}}\right)\right)\|_2^2 \\
&= \mathbb{E}\|\sum_{s=1}^{t}\left(\widehat{\theta}^{(s)} - \theta^{(s)} + \theta^{(s)} - d^{(s)}\right)\|_2^2 \\
&= \mathbb{E}\sum_{i=1}^{n}\left(\sum_{s=1}^{t}\left(\widehat{\theta}^{(s)}_i - \theta^{(s)}_i\right)^2 + \sum_{s=1}^{t}\left(\theta^{(s)}_i - d^{(s)}_i\right)^2 + 2\sum_{s\neq\ell}\left(\widehat{\theta}^{(s)}_i - \theta^{(s)}_i\right)\left(\theta^{(\ell)}_i - d^{(\ell)}_i\right)\right) \\
&= \sum_{s=1}^{t}\mathbb{E}\|\widehat{\theta}^{(s)} - \theta^{(s)}\|_2^2 + \sum_{s=1}^{t}\underbrace{\sum_{i=1}^{n}\left(\theta^{(s)}_i - d^{(s)}_i\right)^2}_{\text{F}} + 0 \\
&\le \sum_{s=1}^{t}\left(\mathbb{E}\|\widehat{\theta}^{(s)}\|^2 + \text{F}\right),
\end{aligned}
$$

where the last step is by the bound on variance by its second moment. Recall that we already have from Equation (A.4.27), $\text{F} \le 400nK^2\eta^2(\sqrt{2}(\varepsilon/n)^{400} + 2\delta_{\exp})$. Substitute this into the bound for E

and Ⓑ, and apply the result of Lemma 2.2.3 to bound $\mathbb{E}\|\widehat{\theta}^{(s)}\|_2^2$; we choose $t = T_{\text{inner}} = \frac{1}{\varepsilon^2}$ and get

$$\text{Ⓑ} \le \underbrace{\frac{K\varepsilon^2}{10000\,(\log n)^{20}} + \frac{1}{\varepsilon^2}\left(\underbrace{19600\log(n/\varepsilon)K\eta^2 + 147000K^2\eta^2\delta_{\text{exp}}}_{\text{second-moment bound from Lemma}\textit{Lemma 2.2.3}} + \underbrace{400nK^2\eta^2\left(\sqrt{2}(\varepsilon/n)^{400} + 2\delta\right)}_{\text{squared error in bias}}\right)}_{\text{Ⓖ}}. \tag{A.4.31}$$

Next, we bound Ⓐ using Jensen's inequality, and use Inequality A.4.31 in Inequality A.4.30 to get

$$\mathbb{E}\|X^{(t)} - \widetilde{X}^{(t)}\| \le 2\sqrt{K+n}\sqrt{\text{Ⓖ}} + \text{Ⓖ}. \tag{A.4.32}$$

Note that to bound Ⓖ, we only need to take care of the second term in Inequality A.4.31, because the first term is already fixed, and the remaining can be fixed by appropriate choices of δ_{exp}. We choose the step size to be

$$\eta = \varepsilon^2\frac{1}{8\times 10^4(\log(n/\varepsilon))^{11}}. \tag{A.4.33}$$

Substituting this in Inequality A.4.31 gives

$$\text{Ⓖ} \le \frac{K\varepsilon^2}{10^4\,(\log n)^{20}} + \frac{K\varepsilon^2}{6\times 10^5\,(\log(n/\varepsilon))^{21}} + \frac{K\varepsilon^2 n\delta_{\text{exp}}}{2500\,(\log(n/\varepsilon))^{12}} + \frac{K\varepsilon^2 n^2\left(\sqrt{2}(\varepsilon/n)^{400} + 2\delta_{\text{exp}}\right)}{4\times 10^5\times(\log(n/\varepsilon))^{12}}.$$

Plugging this back into Inequality A.4.32 with the value of δ_{exp} from Definition 2.2.2 gives:

$$\begin{aligned}
\text{Ⓖ} &\le \frac{K\varepsilon^2}{10^4\,(\log n)^{20}} + \frac{K\varepsilon^2}{6\times 10^5\,(\log n)^{21}} + \frac{2K\varepsilon^{403}}{(\log(n/\varepsilon))^{12}\,n^{389}} + \frac{3K\varepsilon^{402}}{41\,(\log(n/\varepsilon))^{12}\,n^{388}} \\
&\le K\varepsilon^2\left(\frac{1}{10^4\,(\log n)^{20}} + \frac{1}{6\times 10^5\,(\log(n/\varepsilon))^{21}} + \frac{2\varepsilon^{401}}{(\log(n/\varepsilon))^{12}\,n^{389}} + \frac{3\varepsilon^{402}}{41n^{388}\,(\log(n/\varepsilon))^{12}}\right) \\
&\le K\varepsilon^2\left(\frac{1}{5\times 10^3\,(\log n)^{20}} + \frac{6\varepsilon^{401}}{(\log n)^{20}\,n^{380}}\right) \\
&\le \frac{K\varepsilon^2}{4999\,(\log n)^{20}}
\end{aligned}$$

Plugging this back into Inequality A.4.32 and using $K = 40n\,(\log n)^{10}$ gives $\mathbb{E}\|X^{(t)} - \widetilde{X}^{(t)}\| \le 1.132n\varepsilon$. Since Algorithm 2.2.1 only uses the diagonal entries of $\widetilde{X}^{(t)}$ at any iteration t, we can assume the off-diagonal entries exactly equal those in $X^{(t)}$. Therefore $\widetilde{X}^{(t)} - X^{(t)}$ is a diagonal matrix. For a diagonal matrix A, we can see that $\|A\| = \|A\|_{\text{nuc}}$. Therefore, we have $\mathbb{E}\|X^{(t)} - \widetilde{X}^{(t)}\|_{\text{nuc}} \le 1.132n\varepsilon$. □

A.4.6 The Expanded Domain Trick for Projection

The goal of this section is two-fold: first, we show that if the trace constraint is inactive, the projection step is simple and requires no trace normalization; second, we prove that the trace

constraint remains inactive throughout the run of our algorithm. We remark that this is also the lemma where we choose the optimal number of iterations in the outer loop of Algorithm 2.2.1.

Lemma A.4.25. *Consider the mirror map* $\Phi(X) = X \bullet \log X - \operatorname{Tr} X$ *over the domain* $\{X : X \succeq 0, \operatorname{Tr} X \leq K\}$. *Assuming that the trace inequality is never active, we have that* $\exp Y = \operatorname{argmin}_{X \succeq 0, \operatorname{Tr} X \leq K} \Phi(X) - Y \bullet X$.

Proof. We wish to solve

$$\min X \bullet \log X - \operatorname{Tr} X - X \bullet Y, \text{ subject to } X \succeq 0, \operatorname{Tr} X \leq K. \tag{A.4.34}$$

By diagonalizing X as $X = U \Lambda U^\top$ and Y as $Y = V \Sigma V^\top$, we can rewrite this problem as

$$\min \sum_{i=1}^n \lambda_i \log \lambda_i - \sum_{i=1}^n \lambda_i - \sum_{i=1}^n \lambda_i \widetilde{y}_i, \text{ subject to } \lambda_i \geq 0, \sum_{i=1}^n \lambda_i \leq K, \tag{A.4.35}$$

where $\widetilde{y}_i$ is the i'th diagonal entry of the matrix $U^\top Y U$. The Lagrangian is given by $\mathcal{L}(\lambda_i, \nu) = \sum_{i=1}^n \lambda_i \log \lambda_i - \sum_{i=1}^n \lambda_i - \sum_{i=1}^n \lambda_i \widetilde{y}_i + \nu \left(\sum_{i=1}^n \lambda_i - K \right)$. Setting the gradient to zero gives $\nabla_\Lambda \mathcal{L} = \mathbf{1} + \log \lambda^* - \mathbf{1} - \widetilde{y} + \nu \mathbf{1} = 0$, which gives $\lambda_i^* = \exp(\widetilde{y}_i - \nu)$ for all i. Since we assumed that the trace constraint is *not* active, it means, by complementary slackness, $\nu = 0$ (note that this assumption is justified because we prove it in Lemma 2.2.8). This gives $\lambda_i^* = \exp(\widetilde{y}_i)$ which translates to $X^* = \exp(Y)$, as claimed. □

Before we start the second proof, we need the following result.

Lemma A.4.26. *Fix a norm* $\|\cdot\|$. *Given an* α*-strongly convex mirror map* $\Phi : \mathcal{D} \to \mathbb{R}$, *a convex, G-Lipschitz objective* $f : \mathcal{X} \to \mathbb{R}$, *the diameter of* $\mathcal{X} \cap \mathcal{D}$ *denoted by* $D \stackrel{def}{=} \sup_{x \in \mathcal{X} \cap \mathcal{D}} \Phi(X) - \inf_{x \in \mathcal{X} \cap \mathcal{D}} \Phi(x)$, *step size* η, *and parameter* δ' *where* $\mathbb{E}\|x^{(t)} - \widetilde{x}^{(t)}\| \leq \delta'$, *running mirror descent for* T *iterations gives iterates* $\{\widetilde{x}^{(t)}\}_{t=1}^T$ *that satisfy the inequality*

$$f\left(\frac{1}{T-1} \sum_{t=1}^{T-1} \widetilde{x}^{(t)} \right) - f(x^*) \leq \frac{\eta G^2}{2\alpha} + \frac{1}{\eta (T-1)} (D_\Phi(x^*, \widetilde{x}^{(1)}) - D_\Phi(x^*, \widetilde{x}^{(T)})) + \delta' G.$$

This can be derived the same way as Theorem 4.2 in [B+15], by incorporating the error in iterate, just as we did in the proof of Theorem 2.1.3.

Lemma 2.2.8. *For any iteration* t *of Algorithm 2.2.1,* $\widetilde{X}^{(t)}$ *satisfies* $\operatorname{Tr} \widetilde{X}^{(t)} < K$ *for* $K = 40n(\log n)^{10}$.

Proof. We prove this by induction on the iteration count.

Induction Hypothesis. We assume that for any iteration t, the primal iterate is not too far from the optimal point, satisfying $\|\widetilde{X}^{(t)} - X^*\| \leq 38n (\log n)^{10}$.

Base Case. Since $Y^{(1)} = 0$, the primal iterate $\widetilde{X}^{(1)} = I$. We also know that the optimal point satisfies $\operatorname{Tr} X^* = n$. Therefore, $\|\widetilde{X}^{(1)} - X^*\| \leq 2n \leq 38n (\log n)^{10}$. The hypothesis is thus true for the base case, $t = 1$.

Induction. Suppose that the hypothesis is true for some $t = t'$. We prove that this would make it true for $t = t' + 1$ as well. Our technique is to first prove a weak bound for $\|\widetilde{X}^{(t)} - X^*\|$ using triangle inequality of norms; then we boost our bound (and obtain the stronger guarantee of the induction

hypothesis) by invoking strong convexity of Bregman Divergence. We now show the details.

$$\begin{aligned}\|\widetilde{X}^{(t'+1)} - X^*\| &\leq \|\widetilde{X}^{(t'+1)} - \widetilde{X}^{(t')}\| + \|\widetilde{X}^{(t')} - X^*\| \\ &\leq \underbrace{\|\widetilde{X}^{(t'+1)} - \widetilde{X}^{(t')}\|_{\text{nuc}}}_{\text{Equation (A.2.6)}} + \underbrace{\|\widetilde{X}^{(t')} - X^*\|}_{\text{induction hypothesis}} . \\ &\leq \underbrace{\frac{2\eta G}{\alpha}}_{\text{(A)}} + 38n\,(\log n)^{10} . \end{aligned} \tag{A.4.36}$$

The first step here used the fact that $\|M\| \leq \|M\|_{\text{nuc}}$(We can show this by Hölder's Inequality, $\langle X, Y\rangle \leq \|Y\|_{\text{op}}\|X\|_{\text{nuc}}$. Select $Y = \textbf{diag}(())\,\text{sgn}\,(\textbf{diag}(X))$, that is, Y is a diagonal matrix with $Y_{ii} = \text{sgn}\,(X_{ii})$). We can plug in parameters of the mirror map and the step size, as displayed in Table 2.1, to obtain:

$$\text{(A)} = 2 \cdot \frac{\varepsilon^2}{80000(\log(n/\varepsilon))^{11}} \cdot 2 \cdot 4(40n(\log n)^{10}) \leq \frac{n\varepsilon^2}{125}.$$

Plugging this back into Inequality A.4.36 while using $\varepsilon < 1/2$ and $K = 40n(\log n)^{10}$ gives $\|\widetilde{X}^{(t'+1)} - X^*\| \leq \frac{n\varepsilon^2}{125} + 38n\,(\log n)^{10}$, which implies that $\text{Tr}\big(\widetilde{X}^{(t')}\big) < (n(\varepsilon^2/125 + 38(\log n)^{10}) + n) < 40n(\log n)^{10} = K$, which says that the trace constraint on the iterates is not active on the first t' iterations.

Since the trace constraint is not active on the first t' iterations, the projection step does not require a normalization. This implies that Algorithm A.2.1 now is identical to Approximate Mirror Descent with this mirror map and objective. We now recall Lemma A.4.26 for $T = t' + 1$:

$$f\left(\frac{1}{t'}\sum_{t=1}^{t'}\widetilde{X}^{(t)}\right) - f\,(X^*) \leq \frac{\eta G^2}{2\alpha} + \frac{1}{\eta t'}(D_\Phi(X^*, \widetilde{X}^{(1)}) - D_\Phi(X^*, \widetilde{X}^{(t'+1)})) + \delta' G.$$

Multiplying throughout by $\eta t'$ and rearranging the terms gives

$$D_\Phi(X^*, \widetilde{X}^{(t'+1)}) \leq \frac{\eta^2 G^2 t'}{2\alpha} + D_\Phi(X^*, \widetilde{X}^{(1)}) - \eta t' \underbrace{\left(f\left(\frac{1}{t'}\sum_{t=1}^{t'}\widetilde{X}^{(t)}\right) - f\,(X^*)\right)}_{\text{positive}} + \eta t'\delta' G \tag{A.4.37}$$

Since Φ is α-strongly convex in the nuclear norm, we have $D_\Phi(X^*, \widetilde{X}) \geq \frac{\alpha}{2}\|X^* - \widetilde{X}\|_{\text{nuc}}^2$. Since this is at least $\frac{\alpha}{2}\|X^* - \widetilde{X}\|^2$. Chaining this with Inequality A.4.37 gives

$$\|\widetilde{X}^{(t'+1)} - X^*\|^2 \leq \underbrace{\frac{\eta^2 G^2 t'}{\alpha^2}}_{\text{(B)}} + \underbrace{\frac{2D_\Phi(X^*, \widetilde{X}^{(1)})}{\alpha}}_{\text{(C)}} + \underbrace{\frac{2}{\alpha}\eta t'\delta' G}_{\text{(D)}}, \tag{A.4.38}$$

We now bound each of the terms on the right-hand side. We remark that this is actually where we

choose the appropriate value of T_{outer}.

$$\begin{aligned}
\textcircled{B} &= \frac{\eta^2 G^2 T_{\text{inner}} T_{\text{outer}}}{\alpha^2} \\
&= \frac{\varepsilon^4}{64 \times 10^8 \left(\log(n/\varepsilon)\right)^{22}} \cdot 4 \cdot \frac{1}{\varepsilon^2} \cdot \frac{1}{\varepsilon} 24 \times 10^5 \left(\log(n/\varepsilon)\right)^{11} \log n \cdot 16 \left(40n (\log n)^{10}\right)^2 \\
&\leq 40\varepsilon n^2 (\log n)^{10}
\end{aligned}$$

To bound the second term $\textcircled{C} = \frac{2D_\Phi(\widetilde{X}^{(1)}, X^*)}{\alpha}$, we need to compute $D_\Phi(\widetilde{X}^{(1)}, X^*)$. Recall that $\widetilde{X}^{(1)} = I$ by our algorithm. Therefore, $\Phi(\widetilde{X}^{(1)}) = -n$ and $\nabla\Phi(\widetilde{X}^{(1)}) = 0$. Applying Hölder's inequality gives $\Phi(X^*) \leq \operatorname{Tr} X^* \log \|X^*\|_{\text{op}} \leq n \log n$. Therefore $D_\Phi(X^*, \widetilde{X}^{(1)}) \leq n \log n$. Now we go back to the quantity we were trying to bound:

$$\textcircled{C} \leq 2 \cdot n \log n \cdot 4(40n(\log n)^{10}) \leq 320n^2 (\log n)^{11}.$$

Finally, the last term is:

$$\textcircled{D} = \frac{2}{\alpha} \eta T_{\text{inner}} T_{\text{outer}} \delta' G \leq 2 \cdot 4K \cdot \frac{30 \log n}{\varepsilon} \cdot 1.132 n\varepsilon \cdot 2 = 21735 n^2 (\log n)^{11}$$

Summing these terms and plugging back into Equation (A.4.38) gives

$$\begin{aligned}
\|\widetilde{X}^{(t'+1)} - X^*\|^2 &\leq n^2(40\varepsilon(\log n)^{10} + 320(\log n)^{11} + 21735(\log n)^{11}). \\
&< n^2(0.77(\log n)^{20} + 17(\log n)^{20} + 1150(\log n)^{20}) \\
&\leq 1168 n^2 (\log n)^{20} \leq 35n (\log n)^{10},
\end{aligned}$$

which completes the induction. Therefore we have $\|\widetilde{X}^{(t)} - X^*\| \leq 38n (\log n)^{10}$ for all t. Since $\operatorname{Tr} X^* = n$, this proves $\operatorname{Tr} \widetilde{X}^{(t)} < 40n (\log n)^{10} = K$. □

A.4.7 Error bound

Finally, we put together all the parameters derived above to obtain our claimed error bound.

Lemma A.4.27. *Running Algorithm 2.2.1 gives an output for (2.1.2) that has an error bound of $K\varepsilon$.*

Our algorithm is in the framework of approximate lazy mirror descent, with error bound given by Theorem 2.1.3, restated below.

Theorem 2.1.3 (Convergence of Lazy Mirror Descent). *Fix a norm $\|\cdot\|$. Given an α-strongly convex mirror map $\Phi : \mathcal{D} \to \mathbb{R}$ and a convex, G-Lipschitz objective $f : \mathcal{X} \to \mathbb{R}$, run Algorithm A.2.1 with step size η and $\mathbb{E}\|x^{(t)} - \widetilde{x}^{(t)}\| \leq \delta$. Let $D \stackrel{def}{=} \sup_{x \in \mathcal{X} \cap \mathcal{D}} \Phi(x) - \inf_{x \in \mathcal{X} \cap \mathcal{D}} \Phi(x)$ and $x^* = \arg\min_{\mathcal{X}} f(x)$. Then, Algorithm A.2.1, after T iterations, returns $\widetilde{x}^{t^*}$, satisfying*

$$\mathbb{E} f(\widetilde{x}^{(t^*)}) - f(x^*) \leq \frac{D}{T\eta} + \frac{2\eta G^2}{\alpha} + \delta G. \tag{2.1.6}$$

Proof. Our proof involves plugging in the values of the parameters (from Table 2.1) in the above

bound. Since we assume $n \geq 4$, we use $\log n \leq \sqrt{n}$ in one of the calculations below.

$$\frac{D}{T\eta} = K\varepsilon \frac{\log K}{30 \log n} \leq K\varepsilon \frac{\log 40 + 6 \log n}{30 \log n} \leq 0.29K\varepsilon.$$
$$\frac{2\eta G^2}{\alpha} = \frac{32\varepsilon^2 K}{8 \times 10^4 (\log n)^{11}} = \frac{K\varepsilon}{2500 (\log n)^{11}} \leq 2 \times 10^{-5} K\varepsilon$$
$$\delta G = 1.132 n\varepsilon \leq \frac{K\varepsilon}{35 (\log n)^{10}} \leq 11 \times 10^{-4} K\varepsilon$$

Summing these quantities gives the upper bound on the error to be εK, as claimed. □

A.5 General Technical Results

Lemma A.5.1. *Given $a, b \in \mathbb{R}^n$, we have that $\mathbb{E}_{\zeta \sim \mathbb{N}(0,I)} \left((\zeta^T a)^2 (\zeta^T b)^2 \right) \leq 3\|a\|_2^2 \|b\|_2^2$.*

Proof. By Cauchy-Schwarz inequality, the functions f_1 and f_2 satisfy $\mathbb{E}_{\zeta \sim \mathbb{N}(0,I)}(f_1(\zeta) f_2(\zeta)) \leq \sqrt{\mathbb{E}_\zeta (f_1(\zeta))^2 \mathbb{E}_\zeta (f_2(\zeta))^2}$. Choose $f_1(\zeta) = (\zeta^T a)^2$ and $f_2(\zeta) = (\zeta^T b)^2$. Since $\zeta \sim \mathbb{N}(0, I)$ and all the coordinates of ζ are independent, $\mathbf{Var}(\zeta^T a) = \sum_{i=1}^n \mathbf{Var}(\zeta_i a_i) = \sum_{i=1}^n a_i^2 = \|a\|_2^2$. Therefore $\zeta^T a \sim \mathbb{N}(0, \|a\|_2^2)$. For $X \sim \mathbb{N}(0, \sigma^2)$, we have $\mathbb{E}X^4 = 3\sigma^4$. Applying this to $\zeta^T a$ and $\zeta^T b$ proves the desired inequality. □

Appendix B

Appendix for Chapter 3

This chapter contains details and proofs from Chapter 3.

B.1 Notation and Preliminaries

For any integer d, we use $[d]$ to denote the set $\{1, 2, \cdots, d\}$. We use $\mathbb{S}^{n\times n}$ to denote the set of symmetric $n \times n$ matrices, $\mathbb{S}^{n\times n}_{\geq 0}$ for the set of $n \times n$ positive semidefinite matrices, and $\mathbb{S}^{n\times n}_{>0}$ for the set of $n \times n$ positive definite matrices. For two matrices $A, B \in \mathbb{S}^{n\times n}$, the notation $A \preceq B$ means that $B - A \in \mathbb{S}^{n\times n}_{\geq 0}$. When clear from the context, we use 0 to denote the all-zeroes matrix (e.g. $A \succeq 0$). For a vector $v \in \mathbb{R}^n$, we use $\mathbf{diag}(v)$ to denote the diagonal $n \times n$ matrix with $\mathbf{diag}(v)_{i,i} = v_i$. For $A, B \in \mathbb{S}^{n\times n}$, we define the inner product to be the trace product of A and B, defined as $\langle A, B\rangle := \mathrm{tr}[A^\top B] = \sum_{i,j\in[n]} A_{i,j}B_{i,j}$. For two matrices $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{k\times \ell}$, the *Kronecker product* of A and B, denoted as $A \otimes B$, is defined as the $mk \times n\ell$ block matrix whose (i, j) block is $A_{i,j}B$, for all $(i, j) \in [m] \times [n]$.

Throughout this chapter, unless otherwise specified, m denotes the number of constraints for the primal SDP (3.1.1), and the variable matrix X is of size $n \times n$. The number of non-zero entries in all the A_i and C of (3.1.1) is denoted by $\mathrm{nnz}(A)$.

B.1.1 Useful Facts

Linear algebra. Some matrix norms we frequently use in this chapter are the Frobenius and operator norms, defined as follows. The Frobenius norm of a matrix $A \in \mathbb{R}^{n\times n}$ is defined to be $\|A\|_F := \sqrt{\mathrm{tr}[A^\top A]}$. The operator (or spectral) norm $\|A\|_{\mathrm{op}}$ of $A \in \mathbb{R}^{n\times n}$ is defined to be the largest singular value of A. In the case of symmetric matrices (which is what we encounter in this chapter), this can be shown to equal the largest absolute eigenvalue of the matrix. A property of trace we frequently use is the following: given matrices $A_1 \in \mathbb{R}^{m\times n_1}, A_2 \in \mathbb{R}^{n_1\times n_2}, \ldots, A_k \in \mathbb{R}^{n_{k-1}\times n_k}$, the trace of their product is invariant under cyclic permutation $\mathrm{tr}[A_1A_2 \ldots A_k] = \mathrm{tr}[A_2A_3 \ldots A_kA_1] = \cdots = \mathrm{tr}[A_kA_1 \ldots A_{k-2}A_{k-1}]$. A matrix $A \in \mathbb{R}^{n\times n}$ is called *normal* if A commutes with its transpose, i.e. $AA^\top = A^\top A$. We note that all symmetric $n \times n$ matrices are normal. Two matrices $A, B \in \mathbb{R}^{n\times n}$ are said to be similar if there exists a nonsingular matrix $S \in \mathbb{R}^{n\times n}$ such that $A = S^{-1}BS$. In particular, if matrices A and B are similar, then they have the same set of eigenvalues. We use the following simple fact involving Loewner ordering: given two positive definite matrices A and B satisfying $\frac{1}{\alpha}B \preceq A \preceq \alpha B$ for some $\alpha > 0$, we have $\frac{1}{\alpha}B^{-1} \preceq A^{-1} \preceq \alpha B^{-1}$. We further need the following facts.

Fact B.1.1 (Generalized Lieb-Thirring Inequality [Eld13, ALO16b, JLL+20b])**.** *Given a symmetric matrix B, a positive semi-definite matrix A and $\alpha \in [0, 1]$, we have*

$$\mathrm{tr}\left[A^\alpha BA^{1-\alpha}B\right] \leq \mathrm{tr}\left[AB^2\right].$$

Fact B.1.2 (Hoffman-Wielandt Theorem, [AH53, HJ12])**.** *Let $A, E \in \mathbb{R}^{n\times n}$ such that A and $A+E$ are both normal matrices. Let $\lambda_1, \lambda_2, \ldots, \lambda_n$ be the eigenvalues of A, and let $\widehat{\lambda}_1, \widehat{\lambda}_2, \ldots, \widehat{\lambda}_n$ be the eigenvalues of $A+E$ in any order. There is a permutation σ of the integers $1, \ldots, n$ such that $\sum_{i\in[n]} |\widehat{\lambda}_{\sigma(i)} - \lambda_i|^2 \le \|E\|_F^2$.*

Fact B.1.3 (Corollary of Fact B.1.2, [HJ12])**.** *Let $A, E \in \mathbb{R}^{n\times n}$ such that A is Hermitian and $A+E$ is normal. Let $\lambda_1, \ldots, \lambda_n$ be the eigenvalues of A arranged in increasing order $\lambda_1 \le \ldots \le \lambda_n$. Let $\widehat{\lambda}_1, \ldots, \widehat{\lambda}_n$ be the eigenvalues of $A+E$, ordered so that $\mathrm{Re}(\widehat{\lambda}_1) \le \ldots \le \mathrm{Re}(\widehat{\lambda}_n)$. Then, $\sum_{i\in[n]} |\widehat{\lambda}_i - \lambda_i|^2 \le \|E\|_F^2$.*

Fact B.1.4 (Woodbury matrix identity, [Woo49, Woo50])**.** *Given matrices $A \in \mathbb{R}^{n\times n}$, $U \in \mathbb{R}^{n\times k}$, $C \in \mathbb{R}^{k\times k}$, and $V \in \mathbb{R}^{k\times n}$, such that A, C, and $A+UCV$ are invertible, we have*

$$(A+UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}.$$

B.2 Matrix Multiplication

The main goal of this section is to derive upper bounds on the time to perform the following two rectangular matrix multiplication tasks (Lemma B.2.9, Lemma B.2.10, and Lemma B.2.11):

- Multiplying a matrix of dimensions $m \times n^2$ with one of dimensions $n^2 \times m$,
- Multiplying a matrix of dimensions $n \times mn$ with one of dimensions $mn \times n$.

Besides being crucial to the runtime analysis of our interior point method in Section B.6, these results (as well as several intermediate results) might be of independent interest.

B.2.1 Exponent of Matrix Multiplication

We need the following definitions.

Definition B.2.1. *Define $\mathcal{T}_{\mathrm{mat}}(n,r,m)$ to be the number of operations needed to compute the product of matrices of dimensions $n\times r$ and $r\times m$.*

Definition B.2.2. *We define the function $\omega(k)$ to be the minimum value such that $\mathcal{T}_{\mathrm{mat}}(n, n^k, n) = n^{\omega(k)+o(1)}$. We overload notation and use ω to denote the cost of multiplying two $n\times n$ matrices. Thus, we have $\omega(1) = \omega$.*

The following is a basic property of $\mathcal{T}_{\mathrm{mat}}$ that we frequently use.

Lemma B.2.3 ([BCS97, Blä13])**.** *For any three positive integers n, m, r, we have*

$$\mathcal{T}_{\mathrm{mat}}(n,r,m) = O(\mathcal{T}_{\mathrm{mat}}(n,m,r)) = O(\mathcal{T}_{\mathrm{mat}}(m,n,r)).$$

We refer to Table 3 in [GU18] for the latest upper bounds on $\omega(k)$ for different values of k. In particular, we need the following upper bounds in our paper.

Lemma B.2.4 ([GU18])**.** *We have $\omega = \omega(1) \le 2.372927$, $\omega(1.5) \le 2.79654$, $\omega(1.75) \le 3.02159$, and $\omega(2) \le 3.251640$.*

B.2.2 Technical Results for Matrix Multiplication

In this section, we derive some technical results on $\mathcal{T}_{\mathrm{mat}}$ and ω that we extensively use.

Lemma B.2.5 (Sub-linearity)**.** *For any $p \ge q \ge 1$, we have*

$$\omega(p) \le p - q + \omega(q).$$

Proof. We assume that n^p and n^q are integers for notational simplicity. Consider multiplying an $n \times n^p$ matrix with an $n^p \times n$ matrix. One can cut the $n \times n^p$ matrix into n^{p-q} rectangular blocks of size $n \times n^q$ and the $n^p \times n$ matrix into n^{p-q} rectangular blocks of size $n^q \times n$, and compute the multiplication of the corresponding blocks. This approach takes time $n^{p-q+\omega(q)+o(1)}$, from which the desired inequality immediately follows. □

Key to our analysis is the following lemma, which establishes the convexity of $\omega(k)$.

Lemma B.2.6 (Convexity)**.** *The fast rectangular matrix multiplication time exponent $\omega(k)$ as defined in Definition B.2.2 is convex in k.*

Proof. Let $k = \alpha \cdot p + (1-\alpha) \cdot q$ for $\alpha \in (0,1)$. For notational simplicity, we assume that n^p, n^q and n^k are all integers. Consider a rectangular matrix of dimensions $n \times n^k$. Since $\alpha p \le k$, we can tile this rectangular matrix with matrices of dimensions $n^\alpha \times n^{\alpha p}$. Then, the product of this tiled matrix with another similarly tiled matrix of dimensions $n^k \times n$ can be obtained by viewing it as a multiplication of a matrix of dimensions $n/n^\alpha \times n^k/n^{\alpha p}$ with one of dimensions $n^k/n^{\alpha p} \times n^{1/\alpha}$, where each "element" of these two matrices is itself a matrix of dimensions $n^\alpha \times n^{\alpha p}$. With this recursion in tow, we obtain the following upper bound.

$$\begin{aligned}
\mathcal{T}_{\mathrm{mat}}(n, n^k, n) \le& \mathcal{T}_{\mathrm{mat}}(n^\alpha, n^{\alpha p}, n^\alpha) \cdot \mathcal{T}_{\mathrm{mat}}(n/n^\alpha, n^k/n^{\alpha p}, n/n^\alpha) \\
=& \mathcal{T}_{\mathrm{mat}}(n^\alpha, n^{\alpha p}, n^\alpha) \cdot \mathcal{T}_{\mathrm{mat}}(n^{(1-\alpha)}, n^{(1-\alpha)q}, n^{(1-\alpha)}) \\
\le& n^{\alpha\cdot\omega(p)+o(1)} \cdot n^{(1-\alpha)\cdot\omega(q)+o(1)}.
\end{aligned}$$

The final step above follows from denoting $m = n^\alpha$ and observing that multiplying matrices of dimensions $n^\alpha \times n^{\alpha \cdot p}$ costs, by Definition B.2.2, $m^{\omega(p)+o(1)}$, which is exactly $n^{\alpha(\omega(p)+o(1))}$. Applying Definition B.2.2 and comparing exponents, this implies that

$$\omega(k) \le \alpha \cdot \omega(p) + (1-\alpha) \cdot \omega(q),$$

which proves the convexity of the function $\omega(k)$. □

Claim B.2.7. $\omega(1.68568) \le 2.96370$.

Proof. We can upper bound $\omega(1.68568)$ in the following sense

$$\begin{aligned}
\omega(1.68568) &= \omega(0.25728 \cdot 1.5 + (1 - 0.25728) \cdot 1.75) \\
&\le 0.25728 \cdot \omega(1.5) + (1 - 0.25728) \cdot \omega(1.75) \\
&\le 0.25728 \cdot 2.79654 + (1 - 0.25728) \cdot 3.02159 \\
&\le 2.96370,
\end{aligned}$$

where the first step follows from convexity of ω (Lemma B.2.6), the third step follows from $\omega(1.5) \le 2.79654$ and $\omega(1.75) \le 3.02159$ (Lemma B.2.4). □

Lemma B.2.8. *Let $\mathcal{T}_{\mathrm{mat}}$ be defined as in Definition B.2.1. Then for any positive integers h, ℓ, and k, we have*

$$\mathcal{T}_{\mathrm{mat}}(h, \ell k, h) \le O(\mathcal{T}_{\mathrm{mat}}(hk, \ell, hk)).$$

Proof. Given any matrices $A, B^\top \in \mathbb{R}^{h,\ell k}$, by Definition B.2.1, the cost of computing the matrix product AB is $\mathcal{T}_{\mathrm{mat}}(h, \ell k, h)$. We now show how to compute this product in time $O(\mathcal{T}_{\mathrm{mat}}(hk, \ell, hk))$.

We cut A and $B^\top$ into k sub-matrices each of size $h \times \ell$, i.e. $A = (A_1, \cdots, A_k)$ and $B^\top = (B_1^\top, \cdots, B_k^\top)$, where each $A_i, B_i^\top \in \mathbb{R}^{h \times \ell}$ for all $i \in [k]$. By performing matrix multiplication blockwise, we can write $AB = \sum_{i=1}^{k} A_i B_i$. Next, we stack the k matrices $A_1, \cdots, A_k$ vertically to form a matrix $A' \in \mathbb{R}^{hk,\ell}$. Similarly, we stack the k matrices $B_1, \cdots, B_k$ horizontally to form a matrix $B' = (B_1, \cdots, B_k) \in \mathbb{R}^{\ell,hk}$. By Definition B.2.1, we can compute $A'B' \in \mathbb{R}^{hk,hk}$ in time $\mathcal{T}_{\text{mat}}(hk, \ell, hk)$. To complete the proof, we note that we can derive AB from $A'B'$ as follows: for each $j \in [k]$, the jth diagonal block of $A'B'$ of size $h \times h$ is exactly $A_j B_j$, and summing up the k diagonal $h \times h$ blocks of $A'B'$ gives AB. □

B.2.3 General Upper Bound on $\mathcal{T}_{\text{mat}}(n, mn, n)$ and $\mathcal{T}_{\text{mat}}(m, n^2, m)$

Lemma B.2.9. *Let $\mathcal{T}_{\text{mat}}$ be defined as in Definition B.2.1. If $m \geq n$, then we have $\mathcal{T}_{\text{mat}}(n, mn, n) \leq O(\mathcal{T}_{\text{mat}}(m, n^2, m))$. If $m \leq n$, then we have $\mathcal{T}_{\text{mat}}(m, n^2, m) \leq O(\mathcal{T}_{\text{mat}}(n, mn, n))$.*

Proof. We only prove the case of $m \geq n$, as the proof of the other case is similar. This is an immediate consequence of Lemma B.2.8 by taking $h = n$, $\ell = n^2$, and $k = \lfloor m/n \rfloor$, where k is a positive integer because $m \geq n$. □

In the next lemma, we derive upper bounds on the term $\mathcal{T}_{\text{mat}}(m, n^2, m)$ when $m \geq n$ and $\mathcal{T}_{\text{mat}}(n, mn, n)$ when $m < n$, which is crucial to our runtime analysis.

Lemma B.2.10. *Let $\mathcal{T}_{\text{mat}}$ be defined as in Definition B.2.1 and ω be defined as in Definition B.2.2. Then we have $\mathcal{T}_{\text{mat}}(n, mn, n) \leq O(mn^{\omega+o(1)})$ and $\mathcal{T}_{\text{mat}}(m, n^2, m) \leq O\left(\sqrt{n}\left(mn^2 + m^\omega\right)\right)$.*

Proof. Recall from Definition B.2.1 that $\mathcal{T}_{\text{mat}}(n, mn, n)$ is the cost of multiplying a matrix of size $n \times mn$ with one of size $mn \times n$. We can cut each of the matrices into m sub-matrices of size $n \times n$ each. The product in question then can be obtained by multiplying these sub-matrices. Since there are m of them, and each product of an $n \times n$ submatrix with another $n \times n$ submatrix costs, by definition, $n^{\omega+o(1)}$, we get $\mathcal{T}_{\text{mat}}(n, mn, n) \leq O(mn^{\omega+o(1)})$, as claimed.

Let $m = n^a$, where $a \in (0, \infty)$. By definition, $\mathcal{T}_{\text{mat}}(m, n^2, m)$ is the cost of multiplying a matrix of size $m \times n^2$ with one of size $n^2 \times m$. Expressing n^2 as $m^{2/a}$ then gives, by Definition B.2.2, that

$$\mathcal{T}_{\text{mat}}(m, n^2, m) = m^{\omega(2/a)+o(1)} = n^{a \cdot \omega(2/a)+o(1)}. \tag{B.2.1}$$

The claimed inequality follows immediately from the following, which we prove next:

$$\omega(2/a) < \max(1 + 2.5/a, \omega(1) + 0.5/a) \qquad \forall a \in (0, \infty). \tag{B.2.2}$$

Define $b = 2/a \in (0, \infty)$. Then the desired inequality in Equation (B.2.2) can be expressed as

$$\omega(b) < \max(1 + 5b/4, \omega(1) + b/4) \qquad \forall b \in (0, \infty). \tag{B.2.3}$$

Notice that the RHS of Inequality B.2.3 is a maximum of two linear functions of b and these intersect at $b^* = \omega(1) - 1$. By the convexity of $\omega(\cdot)$ as proved in Lemma B.2.6, it suffices to verify Inequality B.2.3 at the endpoints $b \to 0$, $b \to \infty$ and $b = b^*$. In the case where $b = \delta$ for any $\delta < 1$, Inequality B.2.3 follows immediately from the observation that $\omega(\delta) < \omega(1)$. We next argue about the case $b \to \infty$. By Lemma B.2.4 we have $\omega(2) \leq 3.252$. Using Lemma B.2.5, we have $\omega(b) \leq b - 2 + \omega(2)$. Combining these two facts implies that for any $b > 2$, we have

$$\omega(b) \leq b - 2 + \omega(2) \leq 1 + 5b/4,$$

which again satisfies Inequality B.2.3. The final case is $b = b^* = \omega(1) - 1$, for which Inequality B.2.3 is equivalent to

$$\omega(\omega(1) - 1) < 5\omega(1)/4 - 1/4. \tag{B.2.4}$$

By Lemma B.2.4, we have that $\omega(1) - 2 \in [0, 0.372927]$. Then to prove Inequality B.2.4, it is sufficient to show that

$$\omega(t + 1) < 5t/4 + 9/4 \qquad \forall t \in [0, 0.372927]. \tag{B.2.5}$$

By the convexity of $\omega(\,\cdot\,)$ as proved in Lemma B.2.6, the upper bound of $\omega(2) \le 3.251640$ in Lemma B.2.4, and recalling that $\omega(1) = t + 2$ for $t \in [0, 0.372927]$, we have for $k \in [1, 2]$,

$$\omega(k) \le \omega(1) + (k - 1) \cdot (3.251640 - (t + 2)) = t + 2 + (k - 1) \cdot (1.251640 - t).$$

In particular, using this inequality for $k = t + 1$, we have

$$\omega(t + 1) - 5t/4 - 9/4 \le (t + 2) + t \cdot (1.251640 - t) - 5t/4 - 9/4 = -t^2 + 1.00164t - 1/4,$$

which is negative on $t \in [0, 0.372927]$. This establishes Inequality B.2.5 and finishes the proof. □

B.2.4 Improved Upper Bound on $\mathcal{T}_{\text{mat}}(m, n^2, m)$

Lemma B.2.11. *For any two positive integers n and m, we have $\mathcal{T}_{\text{mat}}(m, n^2, m) = o(m^3 + mn^{2.37})$.*

Proof. Let $m = n^a$ where $a \in (0, \infty)$. Recall from Equation (B.2.1) that $\mathcal{T}_{\text{mat}}(m, n^2, m) = m^{\omega(2/a)+o(1)} = n^{a\omega(2/a)+o(1)}$. We consider the following two cases according to the range of a.

Case 1: $a \in [1.18647, \infty)$. In this case, we have $\omega(2/a) \le \omega(2/1.18647) \le \omega(1.68568) < 3$, where the last inequality follows from Claim B.2.7. This implies that

$$\mathcal{T}_{\text{mat}}(m, n^2, m) = o(n^{3a}) = o(m^3). \tag{B.2.6}$$

Case 2: $a \in (0, 1.18647]$. In this case, we have $2/a \in [1.68567, \infty)$. Consider the linear function

$$y(t) = 1 + 2.37 \cdot \frac{t}{2}. \tag{B.2.7}$$

By Claim B.2.7, we have

$$\omega(1.68567) < 2.997 \le y(1.68567). \tag{B.2.8}$$

By Lemma B.2.4, we have

$$\omega(2) < 3.37 = y(2). \tag{B.2.9}$$

An application of Lemma B.2.5 then gives, for any $t \ge 2$, the inequality

$$\omega(t) \le t - 2 + \omega(2) < t - 2 + y(2) \le y(t), \tag{B.2.10}$$

where the last inequality is by definition of $y(t)$ from Equation (B.2.7). Therefore, combining the convexity of $\omega(\,\cdot\,)$, as proved in Lemma B.2.6, with Inequality B.2.8, Inequality B.2.9, and

Inequality B.2.10, we conclude that for any $t \in [1.68567, \infty)$, the function ω is bounded from above by the affine function y, expressed as follows.

$$\omega(t) < y(t) = 1 + 2.37 \cdot \frac{t}{2}.$$

This implies that

$$\mathcal{T}_{\text{mat}}(m, n^2, m) = n^{a \cdot \omega(2/a)+o(1)} = o(n^{a+2.37}) = o(mn^{3.27}). \tag{B.2.11}$$

Combining Equation (B.2.6) and Equation (B.2.11) finishes the proof of the lemma. □

B.3 Our Main Theorem

In this section, we give the formal statement of our main result.

Theorem B.3.1 (Main Result)**.** *Consider a semidefinite program with variable size $n \times n$ and m constraints (assume there are no redundant constraints):*

$$\max C \bullet X \text{ subject to } X \succeq 0, A_i \bullet X = b_i \text{ for all } i \in [m]. \tag{B.3.1}$$

Assume that any feasible solution $X \in \mathbb{S}^{n\times n}_{\geq 0}$ satisfies $\|X\|_{\text{op}} \leq R$. Then for any error parameter $0 < \delta \leq 0.01$, there is an interior point method that outputs in time $O^*(\sqrt{n}(mn^2 + m^\omega + n^\omega)\log(n/\delta))$ a positive semidefinite matrix $X \in \mathbb{R}^{n\times n}_{\geq 0}$ such that

$$C \bullet X \geq C \bullet X^* - \delta \cdot \|C\|_{\text{op}} \cdot R \quad \textit{and} \quad \sum_{i\in[m]} \left|A_i \bullet \widehat{X} - b_i\right| \leq 4n\delta \cdot (R \sum_{i\in[m]} \|A_i\|_1 + \|b\|_1),$$

where ω is the exponent of matrix multiplication, X^* is any optimal solution to the semidefinite program in Equation (B.3.1), and $\|A_i\|_1$ is the Schatten 1-norm of matrix A_i.

The proof of Theorem B.3.1 is given in the subsequent sections.

B.4 Approximate Central Path via Approximate Hessian

Our main result of this section is the following statement about the dual variable y following the approximate central path (in the sense that the objective value is additively close to the optimal values on the central path). We also show, importantly, that our step size is small enough that the next slack matrix S_{new} is not too far (in Frobenius norm) from the current slack matrix S.

Theorem B.4.1 (Approximate Central Path)**.** *Consider a semidefinite program as in (3.1.1) with no redundant constraints. Assume that any feasible solution $X \in \mathbb{S}^{n\times n}_{\geq 0}$ satisfies $\|X\|_{\text{op}} \leq R$. Then for any error parameter $0 < \delta \leq 0.01$ and Newton step size ϵ_N satisfying $\sqrt{\delta} < \epsilon_N \leq 0.1$, Algorithm 3.2.1 outputs, in $T = \frac{40}{\epsilon_N}\sqrt{n}\log(n/\delta)$ iterations, a positive semidefinite matrix $X \in \mathbb{R}^{n\times n}_{\geq 0}$ that satisfies*

$$C \bullet X \geq C \bullet X^* - \delta \cdot \|C\|_{\text{op}} \cdot R \quad \textit{and} \quad \sum_{i\in[m]} \left|A_i \bullet \widehat{X} - b_i\right| \leq 4n\delta \cdot (R \sum_{i\in[m]} \|A_i\|_1 + \|b\|_1), \tag{B.4.1}$$

where X^* is any optimal solution to the semidefinite program in (3.1.1), and $\|A_i\|_1$ is the Schatten 1-norm of matrix A_i. Further, in each iteration of Algorithm 3.2.1, the following invariant holds for $\alpha_H = 1.03$:

$$\|S^{-1/2} S_{\text{new}} S^{-1/2} - I\|_F \leq \alpha_H \cdot \epsilon_N. \tag{B.4.2}$$

Proof. At the start of Algorithm 3.2.1, Lemma B.8.1 is called to modify the semidefinite program to obtain an initial dual solution y for the modified SDP that is close to the dual central path at $\eta = 1/(n+2)$. This ensures that the invariant $g_\eta(y)^\top H(y)^{-1} g_\eta(y) \le \epsilon_N^2$ holds at the start of the algorithm. Therefore, by Lemma B.4.4 and Lemma B.4.5, this invariant continues to hold throughout the run of the algorithm. Therefore, after $T = \frac{40}{\epsilon_N}\sqrt{n}\log\left(\frac{n}{\delta}\right)$ iterations, the step size η in Algorithm 3.2.1 grows to $\eta = (1 + \frac{\epsilon_N}{20\sqrt{n}})^T/(n+2) \ge 2n/\delta^2$. It then follows from Lemma B.4.6 that

$$b^\top y \le b^\top y^* + \frac{n}{\eta}\cdot(1+2\epsilon_N) \le b^\top y^* + \delta^2.$$

Thus when the algorithm stops, the dual solution y has duality gap at most δ^2 for the modified SDP. Lemma B.8.1 then shows how to obtain an approximate solution to the original SDP that satisfies the guarantees in Equation (B.4.1).

To prove Inequality B.4.2, define $\Delta_S = S_{\text{new}} - S \in \mathbb{R}^{n\times n}$ and $\delta_y = y_{\text{new}} - y \in \mathbb{R}^m$. For each $i \in [n]$, we use $\delta_{y,i}$ to denote the i-th coordinate of vector δ_y. Then $\Delta_S = \sum_{i=1}^m \delta_{y,i} A_i$ by definition. We rewrite $\|S^{-1/2} S_{\text{new}} S^{-1/2} - I\|_F^2$ as

$$\|S^{-1/2} S_{\text{new}} S^{-1/2} - I\|_F^2 = \operatorname{tr}\left[(S^{-1/2}(\Delta_S)S^{-1/2})^2\right] = \sum_{i,j=1}^m \delta_{y,i}\delta_{y,j}\operatorname{tr}\left[S^{-1}A_i S^{-1}A_j\right] = (\delta_y)^\top H(y)\delta_y. \quad \text{(B.4.3)}$$

Now using $\delta_y = \widetilde{H}(y)^{-1} g_\eta(y)$ simplifies the above expression to $g_\eta(y)^\top \widetilde{H}(y)^{-1} H(y) \widetilde{H}(y)^{-1} g_\eta(y)$. It then follows from Lemma B.4.4 and the invariant $g_\eta(y)^\top H(y)^{-1} g_\eta(y) \le \epsilon_N^2$ on the Newton decrement that

$$g_\eta(y)^\top \widetilde{H}(y)^{-1} H(y) \widetilde{H}(y)^{-1} g_\eta(y) \le \alpha_H^2 \cdot \epsilon_N^2, \quad \text{(B.4.4)}$$

where $\alpha_H = 1.03$. Combining Equation (B.4.3) with Inequality B.4.4 finishes the proof. □

Table B.1: Summary of parameters in approximate central path.

Notation	Choice	Appearance	Meaning
α_H	1.03	Lemma B.4.4	Spectral approximation factor $\alpha_H^{-1}\cdot H \preceq \widetilde{H} \preceq \alpha_H \cdot H$
ϵ_N	0.1	Lemma B.4.5	Upper bound on the Newton step size $(g_\eta^\top H^{-1} g_\eta)^{1/2}$
ϵ_S	0.01	Lemma B.4.2	Spectral approximation error $(1-\epsilon_S)\cdot S \preceq \widetilde{S} \preceq (1+\epsilon_S)\cdot S$

B.4.1 Approximate Slack Update

In this section, we show that the approximate slack matrix $\widetilde{S}$ we maintain is in the operator norm close enough to the true slack matrix S.

Lemma B.4.2. *Given positive definite matrices $S_{\text{new}}, \widetilde{S} \in \mathbb{S}^{n\times n}_{>0}$ and any parameter $0 < \epsilon_S < 0.01$, there is an algorithm that takes $O(n^{\omega+o(1)})$ time to output a positive definite matrix $\widetilde{S}_{\text{new}} \in \mathbb{S}^{n\times n}_{>0}$ such that*

$$\|S_{\text{new}}^{-1/2}\widetilde{S}_{\text{new}}S_{\text{new}}^{-1/2} - I\|_{\text{op}} \le \epsilon_S. \quad \text{(B.4.5)}$$

Proof. The operator norm bound of Inequality B.4.5 is achieved by design in Algorithm 3.2.2. Specifically, we notice that λ_{new} are the eigenvalues of $S_{\text{new}}^{-1/2}\widetilde{S}_{\text{new}}S_{\text{new}}^{-1/2} - I$ and by the algorithm

description (lines 6 - 13), the upper bound $(\lambda_{\text{new}})_i \leq \epsilon_S$ holds for each $i \in [n]$. The claimed runtime of $O(n^{\omega+o(1)})$ is by the spectral decomposition $Z = U \cdot \Lambda \cdot U^\top$, the costliest step in the algorithm. □

B.4.2 Tracking the True Slack Implies Tracking the True Hessian

In this section we show that the spectral closeness of the true and approximate slack matrices, as shown in Section B.4.1, implies that the true and approximate Hessian matrices are also spectrally close. We show this in Lemma B.4.3 and apply it in Lemma B.4.4.

Lemma B.4.3. *Given symmetric matrices $A_1, \cdots, A_m \in \mathbb{S}^{n\times n}$, and positive definite matrices $\widetilde{S}, S \in \mathbb{S}^{n\times n}_{>0}$, define matrices $\widetilde{H} \in \mathbb{R}^{m\times m}$ and $H \in \mathbb{R}^{m\times m}$ as $\widetilde{H}_{j,k} = \text{tr}\left[\widetilde{S}^{-1}A_j\widetilde{S}^{-1}A_k\right]$ and $H_{j,k} = \text{tr}\left[S^{-1}A_jS^{-1}A_k\right]$. Then $\widetilde{H}$ and H are positive semidefinite. For accuracy parameter $\alpha_S \geq 1$, if $\alpha_S^{-1} \cdot S \preceq \widetilde{S} \preceq \alpha_S \cdot S$, then*

$$\alpha_S^{-2} \cdot H \preceq \widetilde{H} \preceq \alpha_S^2 \cdot H.$$

Proof. For any vector $v \in \mathbb{R}^n$, we define $A(v) = \sum_{i=1}^m v_i A_i$. We can rewrite $v^\top Hv$ as follows.

$$v^\top Hv = \sum_{i=1}^m \sum_{j=1}^m v_i v_j H_{i,j} = \sum_{i=1}^m \sum_{j=1}^m v_i v_j \,\text{tr}\left[S^{-1}A_iS^{-1}A_j\right] = \text{tr}\left[S^{-1/2}A(v)S^{-1}A(v)S^{-1/2}\right]. \tag{B.4.6}$$

Similarly, we have

$$v^\top \widetilde{H}v = \text{tr}\left[\widetilde{S}^{-1/2}A(v)\widetilde{S}^{-1}A(v)\widetilde{S}^{-1/2}\right]. \tag{B.4.7}$$

As the RHS of Equation (B.4.6) and Equation (B.4.7) are non-negative, both $\widetilde{H}$ and H are positive semidefinite. Since $\widetilde{S} \preceq \alpha_S \cdot S$, we have $S^{-1} \preceq \alpha_S \cdot \widetilde{S}^{-1}$ (see Section B.1.1), which gives the following inequalities

$$\begin{aligned}\text{tr}\left[S^{-1/2}A(v)S^{-1}A(v)S^{-1/2}\right] &\leq \alpha_S \cdot \text{tr}\left[S^{-1/2}A(v)\widetilde{S}^{-1}A(v)S^{-1/2}\right] \\ &\leq \alpha_S^2 \cdot \text{tr}\left[\widetilde{S}^{-1/2}A(v)\widetilde{S}^{-1}A(v)\widetilde{S}^{-1/2}\right],\end{aligned} \tag{B.4.8}$$

where the first inequality follows from viewing $\text{tr}\left[S^{-1/2}A(v)S^{-1}A(v)S^{-1/2}\right]$ as $\sum_{i=1}^n u_i^\top S^{-1}u_i$ for $u_i = A(v)S^{-1/2}e_i$ and the second inequality follows similarly, after using the cyclic permutation property of trace. Similarly, using $\alpha_S^{-1} \cdot S \preceq \widetilde{S}$, we have

$$\text{tr}\left[S^{-1/2}A(v)S^{-1}A(v)S^{-1/2}\right] \geq \alpha_S^{-2} \cdot \text{tr}\left[\widetilde{S}^{-1/2}A(v)\widetilde{S}^{-1}A(v)\widetilde{S}^{-1/2}\right]. \tag{B.4.9}$$

Combining Equation (B.4.8) and Equation (B.4.9) with Equation (B.4.6) and Equation (B.4.7) along with the fact that v can be any arbitrary n-dimensional vector finishes the proof of the lemma. □

Lemma B.4.4. *Throughout Algorithm 3.2.1, for $\alpha_H = 1.03$, the approximate Hessian $\widetilde{H}(y)$ satisfies*

$$\alpha_H^{-1}H(y) \preceq \widetilde{H}(y) \preceq \alpha_H \cdot H(y).$$

Proof. By Lemma B.4.2, given as input two positive definite matrices S_{new} and $\widetilde{S}$, Algorithm 3.2.2 outputs a matrix $\widetilde{S}_{\text{new}}$ such that $\|S_{\text{new}}^{-1/2}\widetilde{S}_{\text{new}}S_{\text{new}}^{-1/2} - I\|_{\text{op}} \leq \epsilon_S$, where $\epsilon_S = 0.01$. By definition of operator norm, this implies that in each iteration of Algorithm 3.2.1, we have, for $\alpha_S = 1.011$,

that $-\epsilon_S I \preceq S^{-1/2}\widetilde{S}S^{-1/2} - I \preceq \epsilon_S I$, and by pre-multiplying and post-multiplying throughout by the positive definite matrix S, we get $-\epsilon_S S \preceq \widetilde{S} - S \preceq \epsilon_S S$, and by rearranging the terms, we get $\alpha_S^{-1} \cdot S \preceq \widetilde{S} \preceq \alpha_S \cdot S$. The statement of this lemma then follows from Lemma B.4.3. □

B.4.3 Standard Results from Interior-Point Methods

The following results are standard in the theory of interior point methods (e.g. see [Ren01b]).

Lemma B.4.5 (Invariance of the Newton Decrement [Ren01b])**.** *Given any parameters $1 \le \alpha_H \le 1.03$ and $0 < \epsilon_N \le 1/10$, suppose that $g_\eta(y)^\top H(y)^{-1} g_\eta(y) \le \epsilon_N^2$ holds for some feasible dual solution $y \in \mathbb{R}^m$ and parameter $\eta > 0$, and positive definite matrix $\widetilde{H} \in \mathbb{S}^{n\times n}_{>0}$ satisfies $\alpha_H^{-1} H(y) \preceq \widetilde{H} \preceq \alpha_H H(y)$. Then $\eta_{\text{new}} = \eta(1 + \frac{\epsilon_N}{20\sqrt{n}})$ and $y_{\text{new}} = y - \widetilde{H}^{-1} g_{\eta_{\text{new}}}(y)$ satisfy*

$$g_{\eta_{\text{new}}}(y_{\text{new}})^\top H(y_{\text{new}})^{-1} g_{\eta_{\text{new}}}(y_{\text{new}}) \le \epsilon_N^2.$$

Lemma B.4.6 (Approximate Optimality [Ren01b])**.** *Suppose $0 < \epsilon_N \le 1/10$, dual feasible solution $y \in \mathbb{R}^m$, and parameter $\eta \ge 1$ satisfy the following bound on the Newton decrement:*

$$g_\eta(y)^\top H(y)^{-1} g_\eta(y) \le \epsilon_N^2.$$

Let y^* be an optimal solution to the dual formulation (3.2.1). Then we have

$$b^\top y \le b^\top y^* + \frac{n}{\eta} \cdot (1 + 2\epsilon_N).$$

B.5 Low-Rank Update

Crucial to being able to efficiently approximate the Hessian in each iteration is the condition that the rank of the update be not too large. We formalize this idea in the following theorem, essential to the runtime analysis in Section B.6.

Theorem B.5.1 (Rank inequality)**.** *Let $r_0 = n$ and r_i be the rank of the update to the approximate slack matrix $\widetilde{S}$ when calling Lemma B.4.2 in iteration i of Algorithm 3.2.1. Then, over T iterations of Algorithm 3.2.1, the ranks r_i satisfy the inequality*

$$\sum_{i=0}^{T} \sqrt{r_i} \le O(T \log^{1.5} n).$$

The rest of this section is devoted to proving Theorem B.5.1. To this end, we define the "error" matrix $Z \in \mathbb{R}^{n\times n}$ as follows

$$Z = S^{-1/2}\widetilde{S}S^{-1/2} - I \tag{B.5.1}$$

and the potential function $\Phi : \mathbb{R}^{n\times n} \to \mathbb{R}$

$$\Phi(Z) = \sum_{i=1}^{n} \frac{|\lambda(Z)|_{[i]}}{\sqrt{i}}, \tag{B.5.2}$$

where $|\lambda(Z)|_{[i]}$ denotes the i'th entry in the list of absolute eigenvalues of Z sorted in descending

order. We show a certain maximum limit in the change in the above potential when S is updated to S_{new} and a certain minimum potential change when $\widetilde{S}$ is changed to $\widetilde{S}_{\text{new}}$. We combine these two bounds and sum over all iterations to claim the above bound on total rank change. The following lemma give the potential change when S is updated to S_{new}.

Lemma B.5.2 (Potential change when S changes)**.** *Suppose matrices S, S_{new} and $\widetilde{S}$ satisfy the inequalities*

$$\|S^{-1/2}S_{\text{new}}S^{-1/2} - I\|_F \le 0.02 \qquad \textit{and} \qquad \|S^{-1/2}\widetilde{S}S^{-1/2} - I\|_{\text{op}} \le 0.01. \tag{B.5.3}$$

Define matrices $Z = S^{-1/2}\widetilde{S}S^{-1/2} - I$ and $Z_{\text{mid}} = (S_{\text{new}})^{-1/2}\widetilde{S}(S_{\text{new}})^{-1/2} - I$. Then we have

$$\Phi(Z_{\text{mid}}) - \Phi(Z) \le \sqrt{\log n}.$$

Proof. As an intermediate goal, we first prove

$$\sum_{i=1}^{n}(\lambda(Z)_{[i]} - \lambda(Z_{\text{mid}})_{[i]})^2 \le 10^{-3}. \tag{B.5.4}$$

We first show that the lemma statement is implied by Inequality B.5.4.

First, we arrange the eigenvalues of Z_{mid} and Z in descending order of value. Let $\lambda(Z_{\text{mid}})_i$ and $\lambda(Z)_i$ be the ith largest eigenvalues of Z_{mid} and Z, respectively. For each $i \in [n]$, denote $\Delta_i = \lambda(Z_{\text{mid}})_i - \lambda(Z)_i$. Then Inequality B.5.4 is equivalent to $\|\Delta\|_2^2 \le 10^{-3}$. Let τ be the descending order of the magnitudes of eigenvalues of Z_{mid}, i.e. $\left|\lambda(Z_{\text{mid}})_{\tau(1)}\right| \ge \cdots \ge \left|\lambda(Z_{\text{mid}})_{\tau(n)}\right|$. Then the potential $\Phi(Z_{\text{mid}})$ can be upper bounded as

$$\begin{aligned}
\Phi(Z_{\text{mid}}) &= \sum_{i=1}^{n}\frac{1}{\sqrt{i}}\left|\lambda(Z_{\text{mid}})_{\tau(i)}\right| \\
&\le \sum_{i=1}^{n}\left(\frac{1}{\sqrt{i}}\left|\lambda(Z)_{\tau(i)}\right| + \frac{1}{\sqrt{i}}|\Delta_{\tau(i)}|\right) \\
&\le \Phi(Z) + \left(\sum_{i=1}^{n}\frac{1}{i}\right)^{1/2}\left(\sum_{i=1}^{n}|\Delta_i|^2\right)^{1/2} \\
&\le \Phi(Z) + \sqrt{\log n},
\end{aligned}$$

where the third line follows from the "rearrangement inequality",

$$\sum_{i=1}^{n}\frac{1}{\sqrt{i}}\left|\lambda(Z)_{\tau(i)}\right| \le \sum_{i=1}^{n}\frac{1}{\sqrt{i}}|\lambda(Z)|_{[i]}$$

and Cauchy-Schwarz inequality, and the final step uses Inequality B.5.4. This proves the lemma.

The remaining part of this proof is therefore devoted to proving Inequality B.5.4. Define $W =$

$S_{\text{new}}^{-1/2} S^{1/2}$. Then, we can express Z_{mid} in terms of Z and W in the following way.

$$\begin{aligned} Z_{\text{mid}} &= (S_{\text{new}})^{-1/2} \widetilde{S} (S_{\text{new}})^{-1/2} - I \\ &= (S_{\text{new}})^{-1/2} S^{1/2} S^{-1/2} \widetilde{S} S^{-1/2} S^{1/2} (S_{\text{new}})^{-1/2} - I \\ &= WZW^\top + WW^\top - I. \end{aligned} \tag{B.5.5}$$

Let $\lambda(M)_{[i]}$ denote the i'th (ordered) eigenvalue of a matrix M. We then have

$$\sum_{i=1}^{n} (\lambda(Z_{\text{mid}})_{[i]} - \lambda(WZW^\top)_{[i]})^2 \le \|Z_{\text{mid}} - WZW^\top\|_F^2 = \|W^\top W - I\|_F^2, \tag{B.5.6}$$

where the first inequality is by Fact B.1.3 (which is applicable here because Z_{mid} and $WZW^\top$ are both normal matrices) and the second step is by Equation (B.5.5). Denote the eigenvalues of $S^{-1/2} S_{\text{new}} S^{-1/2}$ by $\{\nu_i\}_{i=1}^n$. Then the first assumption in Inequality B.5.3 implies that $\sum_{i\in[n]} (\nu_i - 1)^2 \le 4 \times 10^{-4}$. It follows that

$$\|W^\top W - I\|_F^2 = \|S^{1/2} S_{\text{new}}^{-1} S^{1/2} - I\|_F^2 = \sum_{i\in[n]} (1/\nu_i - 1)^2 \le 5 \times 10^{-4}, \tag{B.5.7}$$

where the last inequality is because the first assumption from Inequality B.5.3 implies $\nu_i \ge 0.98$ for all $i \in [n]$. Plugging Equation (B.5.7) into the right hand side of Inequality B.5.6, we have

$$\sum_{i=1}^{n} (\lambda(Z_{\text{mid}})_{[i]} - \lambda(WZW^\top)_{[i]})^2 \le 5 \times 10^{-4}. \tag{B.5.8}$$

Let $W = U\Sigma V^\top$ be the singular value decomposition of W, with U and V being $n \times n$ unitary matrices. Because of the invariance of the Frobenius norm under unitary transformation, Equation (B.5.7) is then equivalent to

$$\|\Sigma^2 - I\|_F = \sum_{i=1}^{n} (\sigma_i^2 - 1)^2 \le 5 \times 10^{-4}. \tag{B.5.9}$$

Since U and V are unitary, the matrix $WZW^\top = U\Sigma V^\top Z V \Sigma U^\top$ is similar to $\Sigma V^\top Z V \Sigma$, and the matrix $Z' = V^\top Z V$ is similar to Z. Therefore,

$$\begin{aligned} \sum_{i=1}^{n} (\lambda(WZW^\top)_{[i]} - \lambda(Z)_{[i]})^2 &= \sum_{i=1}^{n} (\lambda(\Sigma Z' \Sigma)_{[i]} - \lambda(Z')_{[i]})^2 \\ &\le \|\Sigma Z' \Sigma - Z'\|_F^2, \end{aligned} \tag{B.5.10}$$

where the last inequality is by Fact B.1.3. We rewrite the Frobenius norm as

$$\|\Sigma Z' \Sigma - Z'\|_F = \|(\Sigma - I) Z' (\Sigma - I) + (\Sigma - I) Z' + Z' (\Sigma - I)\|_F \le \|(\Sigma - I) Z' (\Sigma - I)\|_F + 2\|(\Sigma - I) Z'\|_F. \tag{B.5.11}$$

The first term can be bounded as:

$$\begin{aligned}\|(\Sigma - I)Z'(\Sigma - I)\|_F^2 &= \mathrm{tr}\Big[(\Sigma - I)Z'(\Sigma - I)^2 Z'(\Sigma - I)\Big] \\ &\le \mathrm{tr}\Big[(\Sigma - I)^4 \cdot (Z')^2\Big] \\ &\le 0.01^2 \cdot \mathrm{tr}\Big[(\Sigma - I)^4\Big] \\ &= \sum_{i=1}^{n} (\sigma_i - 1)^4 \le 5 \times 10^{-8}. \end{aligned} \tag{B.5.12}$$

The first inequality above uses Fact B.1.1, the second used the observation that $\|Z'\|_{\mathrm{op}} = \|Z\|_{\mathrm{op}} \le 0.01$, and the last inequality follows from Inequality B.5.9 and the fact that $\sum_{i=1}^{n}(\sigma_i - 1)^4 \le \sum_{i=1}^{n}(\sigma_i^2 - 1)^2$. Similarly, we can bound the second term as

$$\|(\Sigma - I)Z'\|_F^2 = \mathrm{tr}\Big[(\Sigma - I)(Z')^2(\Sigma - I)\Big] \le \mathrm{tr}\Big[(\Sigma - I)^2(Z')^2\Big] \le 0.01^2 \cdot \mathrm{tr}\Big[(\Sigma - I)^2\Big] \le 10^{-7}. \tag{B.5.13}$$

It follows from Inequality B.5.10, Inequality B.5.11 and Inequality B.5.13 that

$$\sum_{i=1}^{n} (\lambda(WZW^\top)_{[i]} - \lambda(Z)_{[i]})^2 \le 10^{-6}. \tag{B.5.14}$$

Combining Inequality B.5.8 and Inequality B.5.14, we get that $\sum_{i=1}^{n}(\lambda(Z)_{[i]} - \lambda(Z_{\mathrm{mid}})_{[i]})^2 \le 10^{-3}$ which establishes Inequality B.5.4. This completes the proof of the lemma. □

Lemma B.5.3 (Potential change when $\widetilde{S}$ changes)**.** *Given positive definite matrices $S_{\mathrm{new}}, \widetilde{S} \in \mathbb{S}^n_{>0}$, let $\widetilde{S}_{\mathrm{new}}$ and r be generated during the run of Lemma B.4.2 when the inputs are S_{new} and $\widetilde{S}$. Define the matrices $Z_{\mathrm{mid}} = (S_{\mathrm{new}})^{-1/2}\widetilde{S}(S_{\mathrm{new}})^{-1/2} - I$ and $Z_{\mathrm{new}} = (S_{\mathrm{new}})^{-1/2}\widetilde{S}_{\mathrm{new}}(S_{\mathrm{new}})^{-1/2} - I$. Then we have*

$$\Phi(Z_{\mathrm{mid}}) - \Phi(Z_{\mathrm{new}}) \ge \frac{10^{-4}}{\log n}\sqrt{r}.$$

Proof. The setup of the lemma considers the eigenvalues of Z when $\widetilde{S}$ changes. For the sake of notational convenience, we define $y = |\lambda(Z_{\mathrm{mid}})|$, the vector of absolute values of eigenvalues of $Z_{\mathrm{mid}} = S_{\mathrm{new}}^{-1/2}\widetilde{S}S_{\mathrm{new}}^{-1/2} - I$. Recall from Table B.1 that $\epsilon_S = 0.01$. We consider two cases below.

Case 1. There does not exist an $i \le n/2$ that satisfies the two conditions $y_{[2i]} < \epsilon_S$ and $y_{[2i]} < (1 - 1/10\log n)y_{[i]}$. In this case, we have $r = n/2$. We consider two sub-cases.

- Case (a). For all $i \in [n]$, we have $y_{[i]} \ge \epsilon_S$. In this case, we change all n coordinates of y, and the change in each coordinate contributes to a potential decrease of at least $\epsilon_S/\sqrt{n}$. Therefore, we have $\Phi(Z_{\mathrm{mid}}) - \Phi(Z_{\mathrm{new}}) \ge \epsilon_S\sqrt{n} \ge \frac{10^{-4}}{\log n}\sqrt{r}$.
- Case (b). There exists a minimum index $i \le n/2$ such that $y_{[2j]} < \epsilon_S$ holds for all j in the range $i \le j \le n/2$. In this case, for all j in the above range, we have that $y_{[2j]} \ge (1 - 1/10\log n)y_{[j]}$. In particular, picking $j = i, 2i, \cdots$ gives

$$y_{[n]} \ge y_{[i]} \cdot (1 - 1/(10\log n))^{\lceil \log n \rceil} \ge \epsilon_S/10.$$

Recalling that our notation $y_{[i]}$ denotes the i'th absolute eigenvalue in decreasing order, we use the above inequality and repeat the argument from the previous sub-case to conclude that $\Phi(Z_{\text{mid}}) - \Phi(Z_{\text{new}}) \geq \epsilon_S/10 \cdot \sqrt{n} \geq \frac{10^{-4}}{\log n} \cdot \sqrt{r}$.

Case 2. There exists an index i for which both the conditions $y_{[2i]} < \epsilon_S$ and $y_{[2i]} < (1 - 1/10 \log n) y_{[i]}$ are satisfied. By definition, $r \leq n/2$ is the smallest such index. Consider the index j such that for all $j' < j$, we have $y_{[j']} \geq \epsilon_S$ and for all $j' \geq j$, we have $y_{[j]} < \epsilon_S$. By the same argument as in Case 1(b), we can prove $y_{[r]} \geq \epsilon_S/10$. Moreover, $y_{[2r]} < (1 - 1/10 \log n) y_{[r]}$ by definition of r. Denote by y^{new} the vector of magnitudes of the eigenvalues of Z_{new}. Since $y^{\text{new}}_{[i]}$ is set to 0 for each $i \in [2r]$, we have $y^{\text{new}}_{[i]} = y_{[i+2r]} \leq y_{[i]}$. Further, $y_{[2r]} < (1 - 1/10 \log n) y_{[r]}$ implies that for each $i \in [r]$, we have

$$y_{[i]} - y^{\text{new}}_{[i]} \geq \frac{1}{10 \log n} \cdot y_{[r]} \geq \frac{10^{-2} \epsilon_S}{\log n} = \frac{10^{-4}}{\log n},$$

where $\epsilon_S = 0.01$ by Table B.1. Therefore, we can bound, from below, the decrease in potential function as

$$\Phi(Z_{\text{mid}}) - \Phi(Z_{\text{new}}) \geq \sum_{i=1}^{r} \frac{y_{[i]} - y^{\text{new}}_{[i]}}{\sqrt{i}} \geq \frac{10^{-4}}{\log n} \sqrt{r}.$$

This finishes the proof of the lemma. □

Proof of Theorem B.5.1. Recall the definition of the potential function in Equation (B.5.2) for an error matrix $Z \in \mathbb{S}^{n \times n}$:

$$\Phi(Z) = \sum_{i=1}^{n} \frac{|\lambda(Z)|_{[i]}}{\sqrt{i}}.$$

Let $S^{(i)}$ and $\widetilde{S}^{(i)}$ be the true and approximate slack matrices in the ith iteration of Algorithm 3.2.1. Define $Z^{(i)} = (S^{(i)})^{-1/2} \widetilde{S}^{(i)} (S^{(i)})^{-1/2} - I$ and $Z^{(i)}_{\text{mid}} = (S^{(i+1)})^{-1/2} \widetilde{S}^{(i)} (S^{(i+1)})^{-1/2} - I$. First, throughout Algorithm 3.2.1, we satisfy Inequality B.4.2 (as stated in Theorem B.4.1) and Inequality B.4.5 (as stated in Lemma B.4.2). As a result, the assumptions in Lemma B.5.2 are satisfied, and we have that

$$\Phi(Z^{(i)}_{\text{mid}}) - \Phi(Z^{(i)}) \leq \sqrt{\log n}.$$

From Lemma B.5.3, we have the following potential decrease:

$$\Phi(Z^{(i)}_{\text{mid}}) - \Phi(Z^{(i+1)}) \geq \frac{10^{-4}}{\log n} \sqrt{r_i}.$$

These together imply that

$$\Phi(Z^{(i+1)}) - \Phi(Z^{(i)}) \leq \sqrt{\log n} - \frac{10^{-4}}{\log n} \sqrt{r_i}. \tag{B.5.15}$$

We note that $\Phi(Z^{(0)}) = 0$ as we initialized $\widetilde{S} = S$ in the beginning of the algorithm, and that the potential function $\Phi(Z)$ is always non-negative. The theorem then follows by summing up Equation (B.5.15) over all T iterations. □

B.6 Runtime Analysis

Our main result of this section is the following bound on the runtime of Algorithm 3.2.1.

Theorem B.6.1 (Runtime bound)**.** *The total runtime of Algorithm 3.2.1 for solving an SDP with variable size $n \times n$ and m constraints is at most $O^*\left(\sqrt{n}\left(mn^2 + \max(m,n)^\omega\right)\right)$, where ω is the matrix multiplication exponent as defined in Definition B.2.2.*

To prove Theorem B.6.1, we first upper bound the runtime in terms of fast rectangular matrix multiplication times. The iteration complexity of Algorithm 3.2.1 is $T = \widetilde{O}(\sqrt{n})$.

Lemma B.6.2 (Total cost)**.** *The total runtime of Algorithm 3.2.1 over T iterations is upper bounded as*

$$\mathcal{T}_{\text{Total}} \leq O^*\left(\min\left(n \cdot \text{nnz}(A), mn^{2.5}\right) + \sqrt{n}\max(m,n)^\omega + \sum_{i=0}^{T}\left(\mathcal{T}_{\text{mat}}(n, mr_i, n) + \mathcal{T}_{\text{mat}}(m, nr_i, m)\right)\right), \tag{B.6.1}$$

where $\text{nnz}(A)$ is the total number of non-zero entries in all the constraint matrices, r_i, as defined in Theorem B.5.1, is the rank of the update to the approximation slack matrix $\widetilde{S}$ in iteration i, and ω and $\mathcal{T}_{\text{mat}}$ are defined in Definition B.2.2 and Definition B.2.1, respectively.

Remark B.6.3. *A more careful analysis can improve the first term in the RHS of Lemma B.6.2 to $\sqrt{n} \cdot \text{nnz}(A)^{1-\gamma} \cdot (mn^2)^\gamma$ for $\gamma = \frac{1}{2(3-\omega(1))}$. For the purpose of this chapter, however, we will only need the simpler bound given in Lemma B.6.2.*

Proof. The total runtime of Algorithm 3.2.1 consists of two parts:

- **Part 1.** The time to compute the approximate Hessian $\widetilde{H}(y)$ (which we abbreviate as $\widetilde{H}$) in Line 6.
- **Part 2.** The total cost of operations other than computing the approximate Hessian.

Part 1.

We analyze the cost of computing the approximate Hessian $\widetilde{H}$.

Part 1a. Initialization.

We start with computing $\widetilde{H}$ in the first iteration of the algorithm. Each entry of $\widetilde{H}$ involves the computation

$$\widetilde{H}_{j,k} = \text{tr}\left[(\widetilde{S}^{-1/2}A_j\widetilde{S}^{-1/2})(\widetilde{S}^{-1/2}A_k\widetilde{S}^{-1/2})\right].$$

It first costs $O^*(n^\omega)$ to invert $\widetilde{S}$. Then the cost of computing the key module of the approximate Hessian, $\widetilde{S}^{-1/2}A_j\widetilde{S}^{-1/2}$ for all $j \in [m]$, is obtained by stacking the matrices A_j together:

$$\mathcal{T}_{\widetilde{S}^{-1/2}A_j\widetilde{S}^{-1/2} \text{ for all } j\in[m]} \leq O(\mathcal{T}_{\text{mat}}(n, mn, n)). \tag{B.6.2}$$

Vectorizing the matrices $\widetilde{S}^{-1/2}A_j\widetilde{S}^{-1/2}$ into row vectors of length n^2, for each $j \in [m]$, and stacking these rows vertically to form a matrix B of dimensions $m \times n^2$, one observes that $\widetilde{H} = BB^\top$. We therefore have,

$$\mathcal{T}_{\text{computing } \widetilde{H} \text{ from } B} \leq O(\mathcal{T}_{\text{mat}}(m, n^2, m)). \tag{B.6.3}$$

Combining Equation (B.6.2), Equation (B.6.3), and the initial cost of inverting $\widetilde{S}$ gives the following cost for computing $\widetilde{H}$ for the first iteration:

$$\mathcal{T}_{\text{part 1a}} \leq O^*(\mathcal{T}_{\text{mat}}(m, n^2, m) + \mathcal{T}_{\text{mat}}(n, mn, n) + n^\omega). \tag{B.6.4}$$

Part 1b. Accumulating low-rank changes over all the iterations

Once the approximate Hessian in the first iteration has been computed, every next iteration has the approximate Hessian computed using a rank r_i update to the approximate slack matrix $\widetilde{S}$ (see Line 11 of Lemma B.4.2). If the update from $\widetilde{S}$ to $\widetilde{S}_{\text{new}}$ has rank r_i, Fact B.1.4 implies that we can compute, in time $O(n^{\omega+o(1)})$, the $n \times r_i$ matrices V_+ and V_- satisfying $\widetilde{S}_{\text{new}}^{-1} = \widetilde{S}^{-1} + V_+V_+^\top - V_-V_-^\top$. The cost of updating $\widetilde{H}$ is then dominated by the computation of $\text{tr}\left[\widetilde{S}^{-1/2}A_jVV^\top A_k\widetilde{S}^{-1/2}\right]$, where $V \in \mathbb{R}^{n\times r_i}$ is either V_+ or V_-. We note that

$$\mathcal{T}_{A_jV \text{ for all } j\in[m]} \leq O^*\left(\min\left(r_i \cdot \text{nnz}(A), mn^2 r_i^{\omega-2+o(1)}\right)\right), \tag{B.6.5}$$

where $\text{nnz}(A)$ is the total number of non-zero entries in all the constraint matrices, and the second term in the minimum is obtained by stacking the matrices A_j together and splitting it and V into matrices of dimensions $r_i \times r_i$. Further, pre-multiplying $\widetilde{S}^{-1/2}$ with A_jV for all $j \in [m]$ essentially involves computing the matrix product of an $n \times n$ matrix and an $n \times mr_i$ matrix, which, by Definition B.2.1, costs $\mathcal{T}_{\text{mat}}(n, mr_i, n)$. This, together with Equation (B.6.5), gives

$$\mathcal{T}_{\widetilde{S}^{-1/2}A_jV \text{ for all } j\in[m]} \leq O^*\left(\mathcal{T}_{\text{mat}}(n, mr_i, n) + \min\left(r_i \cdot \text{nnz}(A), mn^2 r_i^{\omega-2+o(1)}\right)\right). \tag{B.6.6}$$

The final step is to vectorize all the matrices $\widetilde{S}^{-1/2}A_jV$, for each $j \in [m]$, and stack these vertically to get an $m \times nr_i$ matrix B, which gives the update to Hessian to be computed as $BB^\top$. This costs, by definition, $\mathcal{T}_{\text{mat}}(m, nr_i, m)$. Combining this with Equation (B.6.6) gives the following run time for one update to the approximate Hessian:

$$\mathcal{T}_{\text{rank } r_i \text{ Hessian update}} \leq O^*\left(\mathcal{T}_{\text{mat}}(n, mr_i, n) + \min\left(r_i \cdot \text{nnz}(A), mn^2 r_i^{\omega-2}\right) + \mathcal{T}_{\text{mat}}(m, nr_i, m) + n^\omega\right). \tag{B.6.7}$$

Applying this over all $T = \widetilde{O}(\sqrt{n})$ iterations and $\sum_{i=0}^{T}\sqrt{r_i} \leq \widetilde{O}(\sqrt{n})$ from Theorem B.5.1, gives

$$\mathcal{T}_{\text{part 1b}} \leq O^*\left(\min(n \cdot \text{nnz}(A), mn^{2.5}) + \sqrt{n} \cdot n^\omega + \sum_{i=1}^{T}(\mathcal{T}_{\text{mat}}(n, mr_i, n) + \mathcal{T}_{\text{mat}}(m, nr_i, m))\right). \tag{B.6.8}$$

Combining Part 1a and 1b.

Combining Equation (B.6.4) and Equation (B.6.8), we have

$$\begin{aligned}\mathcal{T}_{\text{part 1}} &\leq \mathcal{T}_{\text{part 1a}} + \mathcal{T}_{\text{part 1b}} \\ &\leq O^*\left(\min(n \cdot \text{nnz}(A), mn^{2.5}) + \sqrt{n} \cdot n^\omega + \sum_{i=0}^{T}(\mathcal{T}_{\text{mat}}(n, mr_i, n) + \mathcal{T}_{\text{mat}}(m, nr_i, m))\right),\end{aligned} \tag{B.6.9}$$

where we incorporated the bound from Equation (B.6.4) into the $i = 0$ case.

Part 2.

Observe that there are four operations performed in Algorithm 3.2.1 other than computing $\widetilde{H}$:

- **Part 2a.** computing the gradient $g_\eta(y)$
- **Part 2b.** inverting the approximate Hessian $\widetilde{H}$
- **Part 2c.** updating the dual variables y_{new} and $S(y_{\text{new}})$
- **Part 2d.** computing the new approximate slack matrix $\widetilde{S}(y_{\text{new}})$

Part 2a. The i'th coordinate of the gradient is expressed as $g_\eta(y)_i = \eta b_i - \text{tr}\left[S^{-1}A_i\right]$. The cost per iteration of computing this quantity equals $O(\text{nnz}(A) + n^{\omega+o(1)})$, where the second term comes from inverting the matrix S.

Part 2b. The cost of inverting the approximate Hessian $\widetilde{H}$ is $O(m^{\omega+o(1)})$ per iteration.

Part 2c. The cost of updating the dual variable $y_{\text{new}} = y - \widetilde{H}^{-1}g_{\eta_{\text{new}}}(y)$, given $\widetilde{H}^{-1}$ and $g_{\eta_{\text{new}}}(y)$, is $O(m^2)$ per iteration. The cost of computing the new slack matrix $S_{\text{new}} = \sum_{i\in[m]}(y_{\text{new}})_i A_i - C$ is $O(\text{nnz}(A))$ per iteration.

Part 2d. The per iteration cost of updating the approximate slack matrix $\widetilde{S}_{\text{new}}$ is $O(n^{\omega+o(1)})$ by Lemma B.4.2.

Combining Part 2a, 2b, 2c and 2d.

The total cost of operations other than computing the Hessian over the $T = \widetilde{O}(\sqrt{n})$ iterations is therefore bounded by

$$\mathcal{T}_{\text{part 2}} \leq \mathcal{T}_{\text{part 2a}} + \mathcal{T}_{\text{part 2b}} + \mathcal{T}_{\text{part 2c}} + \mathcal{T}_{\text{part 2d}} \leq O^*(\sqrt{n}(\text{nnz}(A) + \max(m,n)^\omega)). \tag{B.6.10}$$

Combining Part 1 and Part 2.

Combining Inequality B.6.9 and Inequality B.6.10 and using $r_0 = n$ finishes the proof of the lemma.

$$\begin{aligned}\mathcal{T}_{\text{total}} &\leq \mathcal{T}_{\text{part 1}} + \mathcal{T}_{\text{part 2}} \\ &\leq O^*\left(\min\left(n\cdot\text{nnz}(A), mn^{2.5}\right) + \sqrt{n}\max(m,n)^\omega + \sum_{i=0}^{T}\left(\mathcal{T}_{\text{mat}}(n, mr_i, n) + \mathcal{T}_{\text{mat}}(m, nr_i, m)\right)\right).\end{aligned}$$

□

Lemma B.6.4. *Let $\mathcal{T}_{\text{mat}}$ be as defined in Definition B.2.1. Let $T = \widetilde{O}(\sqrt{n})$ and $\{r_1, \cdots, r_T\}$ be a sequence that satisfies*

$$\sum_{i=1}^{T}\sqrt{r_i} \leq O(T\log^{1.5} n)$$

Property I. We have

$$\sum_{i=1}^{T}\mathcal{T}_{\text{mat}}(m, nr_i, m) \leq O^*(\sqrt{n}\max(m^\omega, n^\omega) + \mathcal{T}_{\text{mat}}(m, n^2, m)),$$

Property II. We have

$$\sum_{i=1}^{T} \mathcal{T}_{\text{mat}}(n, mr_i, n) \le O^*(\sqrt{n}\max(m^\omega, n^\omega) + \mathcal{T}_{\text{mat}}(n, mn, n)).$$

Proof. We give only the proof of Property I, as the proof of Property II is similar. Let $m = n^a$. For each $i \in [T]$, let $r_i = n^{b_i}$, where $b_i \in [0, 1]$. Then

$$\mathcal{T}_{\text{mat}}(m, nr_i, m) = \mathcal{T}_{\text{mat}}(n^a, n^{1+b_i}, n^a) = n^{a\omega((1+b_i)/a)+o(1)}. \tag{B.6.11}$$

For each number $k \in \{0, 1, \cdots, \log n\}$, define the set of iterations

$$I_k = \{i \in [T] \ : \ 2^k \le r_i \le 2^{k+1}\}.$$

Then our assumption on the sequence $\{r_1, \cdots, r_T\}$ can be expressed as $\sum_{k=0}^{\log n} |I_k| \cdot 2^{k/2} \le O(T \log^{1.5} n)$. This implies that for each $k\{0, 1, \cdots, \log n\}$, we have $|I_k| \le O(T \log^{1.5} n/2^{k/2})$. Next, taking the summation of Eq. Equation (B.6.11) over all $i \in [T]$, we have

$$\begin{aligned}
\sum_{i=1}^{T} \mathcal{T}_{\text{mat}}(m, nr_i, m) &= \sum_{i=1}^{T} n^{a\cdot\omega((1+b_i)/a)} \\
&= \sum_{k=0}^{\log n} \sum_{i\in I_k} n^{a\cdot\omega((1+b_i)/a)} \\
&\le O(\log n) \cdot \max_k \max_{i\in I_k} \frac{T\log^{1.5} n}{2^{k/2}} \cdot n^{a\cdot\omega((1+b_i)/a)} \\
&\le \widetilde{O}(1) \cdot \max_k \max_{2^k \le n^{b_i} \le 2^{k+1}} \frac{\sqrt{n}}{2^{k/2}} \cdot n^{a\cdot\omega((1+b_i)/a)} \\
&\le \widetilde{O}(1) \cdot \max_{b_i\in[0,1]} n^{1/2-b_i/2+a\cdot\omega((1+b_i)/a)},
\end{aligned}$$

where the fourth step follows from $T = \widetilde{O}(\sqrt{n})$. To bound the exponent on n above, we define the function g,

$$g(b_i) = 1/2 - b_i/2 + a \cdot \omega((1 + b_i)/a). \tag{B.6.12}$$

This function is convex in b_i due to the convexity of the function ω (Lemma B.2.6). Therefore, over the interval $b_i \in [0, 1]$, the maximum of g is attained at one of the end points. We simply evaluate this function at the end points.

Case 1. Consider the case $b_i = 0$. In this case, we have $g(0) = 1/2 + a\omega(1/a)$. We consider the following two subcases.

Case 1a. If $a \ge 1$, then we have

$$g(0) = 1/2 + a \cdot \omega(1/a) \le 1/2 + a\omega(1) = 1/2 + a\omega$$

Case 1b. If $a \in (0, 1)$, then we define $k = 1/a > 1$. It follows from Lemma Lemma B.2.5 and $\omega > 1$,

that

$$g(0) = 1/2 + a \cdot \omega(1/a) = 1/2 + \omega(k)/k \leq 1/2 + (k - 1 + \omega)/k \leq 1/2 + \omega.$$

Combining both Case 1a and Case 1b, we have that

$$n^{g(0)} \leq \max(n^{1/2+a\omega}, n^{1/2+\omega}) \leq \sqrt{n} \cdot \max(m^\omega, n^\omega).$$

Case 2 Consider the other case of $b_i = 1$. In this case, $g(1) = 1/2 - 1/2 + a\omega(2/a) = a\omega(2/a)$.

We now finish the proof by combining Case 1 and Case 2 as follows.

$$\max_{b_i \in [0,1]} n^{1/2 - b_i + a \cdot \omega((1+b_i)/a)} \leq \sqrt{n} \max(m^\omega, n^\omega) + n^{a \cdot \omega(2/a)}.$$

□

Proof of Theorem B.6.1. In light of Lemma B.6.4, the upper bound on runtime given in Lemma B.6.2 can be written as

$$\mathcal{T}_{\text{Total}} \leq O^* \left(\min \left\{ n \cdot \text{nnz}(A), mn^{2.5} \right\} + \sqrt{n} \max(m, n)^\omega + \mathcal{T}_{\text{mat}}(n, mn, n) + \mathcal{T}_{\text{mat}}(m, n^2, m) \right). \quad \text{(B.6.13)}$$

Combining this with Lemma B.2.10, we have the following upper bound on the total runtime of Algorithm 3.2.1:

$$\begin{aligned} \mathcal{T}_{\text{Total}} &\leq O^* \left(\min \left\{ n \cdot \text{nnz}(A), mn^{2.5} \right\} + \sqrt{n} \max(m, n)^\omega + \sqrt{n} \left(mn^2 + m^\omega \right) \right) \\ &\leq O^* \left(\sqrt{n} \left(mn^2 + \max(m, n)^\omega \right) \right). \end{aligned}$$

This finishes the proof of the theorem. □

B.7 Comparison with Cutting Plane Method

In this section, we prove Theorem 3.1.3, restated below.

Theorem 3.1.3 (Comparison with Cutting Plane Method)**.** *When $m \geq n$, there is an interior point method that solves an SDP with $n \times n$ matrices, m constraints, and $\text{nnz}(A)$ input size, faster than the current best cutting plane method [LSW15], over all regimes of $\text{nnz}(A)$.*

Remark B.7.1. *In the dense case with $\text{nnz}(A) = \Theta(mn^2)$, Algorithm 3.2.1 is faster than the cutting plane method whenever $m \geq \sqrt{n}$.*

Proof of Theorem 3.1.3. Recall that the current best runtime of the cutting plane method for solving an SDP (3.1.1) is $\mathcal{T}_{\text{CP}} = O^*(m \cdot \text{nnz}(A) + mn^{2.372927} + m^3)$ [LSW15], where 2.372927 is the current best upper bound on the exponent of matrix multiplication ω. By Lemma B.6.2 and Lemma B.6.4, we have the following upper bound on the total runtime of Algorithm 3.2.1:

$$\mathcal{T}_{\text{Total}} \leq O^* \left(\min \left\{ n \cdot \text{nnz}(A), mn^{2.5} \right\} + \sqrt{n} \max(m, n)^\omega + \mathcal{T}_{\text{mat}}(n, mn, n) + \mathcal{T}_{\text{mat}}(m, n^2, m) \right)$$

Since $m \geq n$ by assumption, Lemma B.2.9 and Lemma B.2.9 further simplify the runtime to

$$\mathcal{T}_{\text{Total}} \leq O^* \left(\min \left\{ n \cdot \text{nnz}(A), mn^{2.5} \right\} + \sqrt{n} m^\omega + \mathcal{T}_{\text{mat}}(m, n^2, m) \right) \quad \text{(B.7.1)}$$

Note that $\min\left\{n \cdot \mathrm{nnz}(A), mn^{2.5}\right\} \le m \cdot \mathrm{nnz}(A) \le O(\mathcal{T}_{\mathrm{CP}})$ and that $\sqrt{n}m^{\omega} = o(m^3) \le o(\mathcal{T}_{\mathrm{CP}})$ since $m \ge n$. Furthermore, Lemma B.2.11 states that $\mathcal{T}_{\mathrm{mat}}(m, n^2, m) = o(m^3 + mn^{2.37}) \le o(\mathcal{T}_{\mathrm{CP}})$. Since each term on the RHS of Equation (B.7.1) is upper bounded by $\mathcal{T}_{\mathrm{CP}}$, we make the stated conclusion. □

B.8 Initialization

Lemma B.8.1 (Initialization)**.** *Consider a semidefinite program as in (3.1.1) of dimension $n \times n$ with m constraints, and assume that it has the following properties.*

1. *Bounded diameter: for any $X \succeq 0$ with $\langle A_i, X\rangle = b_i$ for all $i \in [m]$, we have $\|X\|_{\mathrm{op}} \le R$.*
2. *Lipschitz objective: $\|C\|_{\mathrm{op}} \le L$.*

For any $0 < \delta \le 1$, the following modified *semidefinite program*

$$\begin{aligned} \max_{\overline{X} \succeq 0} \ & \langle \overline{C}, \overline{X}\rangle \\ \text{s.t. } & \langle \overline{A}_i, \overline{X}\rangle = \overline{b}_i, \forall i \in [m+1], \end{aligned}$$

where

$$\overline{A}_i = \begin{bmatrix} A_i & 0_n & 0_n \\ 0_n^\top & 0 & 0 \\ 0_n^\top & 0 & \frac{b_i}{R} - \mathrm{tr}[A_i] \end{bmatrix}, \quad \forall i \in [m],$$

$$\overline{A}_{m+1} = \begin{bmatrix} I_n & 0_n & 0_n \\ 0_n^\top & 1 & 0 \\ 0_n^\top & 0 & 0 \end{bmatrix}, \ \overline{b} = \begin{bmatrix} \frac{1}{R} b \\ n+1 \end{bmatrix}, \ \overline{C} = \begin{bmatrix} C \cdot \frac{\delta}{L} & 0_n & 0_n \\ 0_n^\top & 0 & 0 \\ 0_n^\top & 0 & -1 \end{bmatrix},$$

satisfies the following statements.

1. *The following are feasible primal and dual solutions:*

$$\overline{X} = I_{n+2}, \ \overline{y} = \begin{bmatrix} 0_m \\ 1 \end{bmatrix}, \ \overline{S} = \begin{bmatrix} I_n - C \cdot \frac{\delta}{L} & 0_n & 0 \\ 0_n^\top & 1 & 0 \\ 0_n^\top & 0 & 1 \end{bmatrix}.$$

2. *For any feasible primal and dual solutions $(\overline{X}, \overline{y}, \overline{S})$ with duality gap at most δ^2, the matrix $\widehat{X} = R \cdot \overline{X}_{[n]\times[n]}$, where $\overline{X}_{[n]\times[n]}$ is the top-left $n \times n$ block submatrix of $\overline{X}$, is an approximate solution to the original semidefinite program in the following sense:*

$$\begin{aligned} \langle C, \widehat{X}\rangle &\ge \langle C, X^*\rangle - LR \cdot \delta, \\ \widehat{X} &\succeq 0, \\ \sum_{i\in[m]} \left|\langle A_i, \widehat{X}\rangle - b_i\right| &\le 4n\delta \cdot (R \sum_{i\in[m]} \|A_i\|_1 + \|b\|_1), \end{aligned}$$

where X^* is any optimal solution to the original SDP and $\|A\|_1$ denotes the Schatten 1-norm of a matrix A.

Proof. For the first result, straightforward calculations show that $\langle \overline{A}_i, \overline{X} \rangle = \overline{b}_i$ for all $i \in [m+1]$, and that $\sum_{i\in[m+1]} \overline{y}_i \overline{A}_i - \overline{S} = \overline{C}$. Now we prove the second result. Denote OPT and $\overline{\mathsf{OPT}}$ the optimal values of the original and modified SDP respectively. Our first goal is to establish a lower bound for $\overline{\mathsf{OPT}}$ in terms of OPT. For any optimal solution $X \in \mathbb{S}^{n\times n}$ of the original SDP, consider the following matrix $\overline{X} \in \mathbb{R}^{(n+2)\times(n+2)}$

$$\overline{X} = \begin{bmatrix} \frac{1}{R}X & 0_n & 0_n \\ 0_n^\top & n+1-\frac{1}{R}\operatorname{tr}[X] & 0 \\ 0_n^\top & 0 & 0 \end{bmatrix}.$$

Notice that $\overline{X}$ is a feasible primal solution to the modified SDP, and that

$$\overline{\mathsf{OPT}} \geq \langle \overline{C}, \overline{X} \rangle = \frac{\delta}{LR} \cdot \langle C, X \rangle = \frac{\delta}{LR} \cdot \mathsf{OPT},$$

where the first step follows because the modified SDP is a maximization problem, and the final step is because X is an optimal solution to the original SDP.

Given a feasible primal solution $\overline{X} \in \mathbb{R}^{(n+2)\times(n+2)}$ of the modified SDP with duality gap δ^2, we could assume $\overline{X} = \begin{bmatrix} \overline{X}_{[n]\times[n]} & 0_n & 0_n \\ 0_n^\top & \tau & 0 \\ 0_n^\top & 0 & \theta \end{bmatrix}$ without loss of generality, where $\tau, \theta \geq 0$. This is because if the entries of $\overline{X}$ other than the diagonal and the top-left $n \times n$ block are not 0, then we could zero these entries out and the matrix remains feasible and positive semidefinite. We thus immediately have $\widehat{X} \geq 0$. Notice that

$$\frac{\delta}{L} \cdot \langle C, \overline{X}_{[n]\times[n]} \rangle - \theta = \langle \overline{C}, \overline{X} \rangle \geq \overline{\mathsf{OPT}} - \delta^2 \geq \frac{\delta}{LR} \cdot \mathsf{OPT} - \delta^2. \tag{B.8.1}$$

Therefore, we can lower bound the objective value for $\overline{X}_{[n]\times[n]}$ in the original SDP as

$$\langle C, \widehat{X} \rangle = R \cdot \langle C, \overline{X}_{[n]\times[n]} \rangle \geq \mathsf{OPT} - LR \cdot \delta,$$

where the last inequality follows from Equation (B.8.1). By matrix Hölder inequality, we have

$$\begin{aligned} \frac{\delta}{L} \cdot \langle C, \overline{X}_{[n]\times[n]} \rangle &\leq \frac{\delta}{L} \cdot \|C\|_{\mathrm{op}} \cdot \operatorname{tr}\left[\overline{X}_{[n]\times[n]}\right] \\ &\leq \frac{\delta}{L} \cdot \|C\|_{\mathrm{op}} \cdot \langle \overline{A}_{m+1}, \overline{X} \rangle \\ &\leq (n+1)\delta, \end{aligned}$$

where in the last step follows from $\|C\|_{\mathrm{op}} \leq L$ and $b_{m+1} = n+1$. We can thus upper bound θ as

$$\theta \leq \frac{\delta}{L} \cdot \langle C, \overline{X}_{[n]\times[n]} \rangle + \delta^2 - \frac{\delta}{LR} \cdot \mathsf{OPT} \leq (2n+1)\delta + \delta^2 \leq 4n\delta, \tag{B.8.2}$$

where the first step follows from Equation (B.8.1), the second step follows from $\mathsf{OPT} \geq -\|C\|_{\mathrm{op}} \cdot \|X^*\|_1 \geq -nLR$ where $\|\cdot\|_1$ is the Schatten 1-norm, and the last step follows from $\delta \leq 1 \leq n$. Notice

that by the feasiblity of $\overline{X}$ for the modified SDP, we have

$$\langle A_i, \overline{X}_{[n]\times[n]} \rangle + (\frac{1}{R} \cdot b_i - \mathrm{tr}[A_i])\theta = \frac{1}{R} \cdot b_i.$$

This implies that

$$\left| \langle A_i, \widehat{X} \rangle - b_i \right| = |(b_i - R \cdot \mathrm{tr}[A_i])\theta| \leq 4n\delta \cdot (R\|A_i\|_1 + |b_i|),$$

where the final step follows from the upper bound of θ in Equation (B.8.2). Summing the above inequality up over all $i \in [m]$ finishes the proof of the lemma. □

Appendix C

Appendix for Chapter 4

This chapter contains the appendix of Chapter 4.

C.1 Decomposable submodular function minimization

C.1.1 Preliminaries

Throughout, V denotes the ground set of elements. A set function $f : 2^V \to \mathbb{R}$ is *submodular* if it satisfies the following *diminishing marginal differences* property:

Definition C.1.1 (Submodularity)**.** *A function $f : 2^V \to \mathbb{R}$ is submodular if $f(T \cup \{i\}) - f(T) \leq f(S \cup \{i\}) - f(S)$, for any subsets $S \subseteq T \subseteq V$ and $i \in V \setminus T$.*

We may assume without loss of generality that $f(\emptyset) = 0$ by replacing $f(S)$ by $f(S) - f(\emptyset)$. We assume that f is accessed by an *evaluation oracle* and use EO to denote the time to compute $f(S)$ for a subset S. Our algorithm for decomposable SFM is based on the Lovász extension [GLS88], a standard convex extension of a submodular function.

Definition C.1.2 (Lovász extension [GLS88])**.** *The Lovász extension $\hat{f} : [0,1]^V \to \mathbb{R}$ of a submodular function f is defined as*

$$\hat{f}(x) = \mathbb{E}_{t\sim[0,1]}[f(\{i \in V : x_i \geq t\})],$$

where $t \sim [0,1]$ is drawn uniformly at random from $[0,1]$.

The Lovász extension $\hat{f}$ of a submodular function f has many desirable properties. In particular, $\hat{f}$ is a convex relaxation of f and it can be evaluated efficiently.

Theorem C.1.3 (Properties of Lovász extension [GLS88])**.** *Let $f : 2^V \to \mathbb{R}$ be a submodular function and $\hat{f}$ be its Lovász extension. Then,*

(a) $\hat{f}$ is convex and $\min_{x\in[0,1]^V} \hat{f}(x) = \min_{S\subseteq V} f(S)$;

(b) $f(S) = \hat{f}(I_S)$ for any subset $S \subseteq V$, where I_S is the indicator vector for S;

(c) Suppose $x \in [0,1]^V$ satisfies $x_1 \geq \cdots \geq x_{|V|}$, then $\hat{f}(x) = \sum_{i=1}^{|V|} (f([i]) - f([i-1]))x_i$.

Property (c) in Theorem C.1.3 allows us to implement a sub-gradient oracle for $\hat{f}$ by evaluating f.

Theorem C.1.4 (Sub-gradient oracle implementation for Lovász extension, Theorem 61 of [LSW15])**.** *Let $f : 2^V \to \mathbb{R}$ be a submodular function and $\hat{f}$ be its Lovász extension. Then a sub-gradient for $\hat{f}$ can be implemented in time $O(|V| \cdot \mathsf{EO} + |V|^2)$.*

C.1.2 Decomposable submodular function minimization proofs

In this subsection, we prove the following more general version of Theorem 4.1.3.

Theorem C.1.5 (Decomposable SFM). *Let $F : V \to [-1, 1]$ be given by $F(S) = \sum_{i=1}^{n} F_i(S \cap V_i)$, where each $F_i : 2^{V_i} \to \mathbb{R}$ is a submodular function on $V_i \subseteq V$ with $|V_i| = d_i$. Let $m = \sum_{i=1}^{n} d_i$ and $d_{\max} := \max_{i \in [n]} d_i$. Then we can find an ϵ-approximate minimizer of f using at most $O(d_{\max} m \log(m/\epsilon))$ evaluation oracle calls.*

Proof. Let $\hat{f}_i$ be the Lovász extension of each f_i, then $\hat{f} = \sum_{i=1}^{n} \hat{f}_i$ is the Lovász extension of f. Note that $\hat{f}$ is 2-Lipschitz since the range of f is $[-1, 1]$. Also, the diameter of the range $[0, 1]^{V_i}$ for each Lovász extension $\hat{f}_i$ is at most $\sqrt{|V_i|} \leq \sqrt{d_{\max}}$. Thus using Theorem 4.1.1, we can find a vector $x \in [0, 1]^V$ such that $\hat{f}(x) \leq \min_{x^* \in [0,1]^V} \hat{f}(x^*) + \epsilon$ in $\mathrm{poly}(m \log(1/\epsilon))$ time and $O(m \log\left(m \sqrt{d_{\max}}/\epsilon\right)) = O(m \log(m/\epsilon))$ subgradients of the $\hat{f}_i$'s. By Theorem C.1.4, each sub-gradient of $\hat{f}_i$ can be computed by making at most $d_i \leq d_{\max}$ queries to the evaluation oracle for f_i. Thus the total number of evaluation oracle calls we make in finding an ϵ-additive approximate minimizer $x \in [0, 1]^V$ of $\hat{f}$ is at most $O(d_{\max} m \log(m/\epsilon))$.

Next we turn the ϵ-additive approximate minimizer x of $\hat{f}$ into an ϵ-additive approximate minimizer $S \subseteq V$ for f. Without loss of generality, assume that $x_1 \geq \cdots \geq x_{|V|}$. Then by property (c) in Theorem C.1.3, we have

$$\hat{f}(x) = \sum_{i=1}^{|V|} (f([i]) - f([i-1]))x_i = f(V) \cdot x_{|V|} + \sum_{i=1}^{|V|-1} f([i]) \cdot (x_i - x_{i+1}).$$

Since $x_i - x_{i+1} \geq 0$, the above implies that $\min_{i \in \{1,\ldots,|V|\}} f([i]) \leq \hat{f}(x)$. Thus we can find a subset $S \subseteq V$ among $f([i])$ for all $i \in \{1, \cdots, |V|\}$ such that $f(S) \leq \hat{f}(x)$. Then by property (a) in Theorem C.1.3, the set S is an ϵ-additive approximate minimizer of f. This proves the theorem. □

Appendix D

Appendix for Chapter 5

This chapter contains details and proofs from Chapter 5.

D.1 Missing Proofs

Theorem 5.2.5. *Let $\{g_k\}$ be generated by* `MinNorm`(x, δ, ϵ). *Fix an index $k \geq 0$, and define the stopping time $\tau \stackrel{def}{=} \inf\{k\colon f(x - \delta\widehat{g_k}) < f(x) - \delta\|g_k\|_2/4$ or $\|g_k\|_2 \leq \epsilon\}$. Then, we have*

$$\mathbb{E}\left[\|g_k\|_2^2 1_{\tau > k}\right] \leq \frac{16L^2}{16+k}.$$

Proof. Fix an index k, and let $\mathbb{E}_k[\cdot]$ denote the conditional expectation on g_k. Suppose we are in the event $\{\tau > k\}$. Taking into account the Lipschitz continuity of f and Lemma 5.2.4, we deduce that almost surely, conditioned on g_k, the following estimate holds:

$$\begin{aligned}
\frac{1}{4}\|g_k\|_2 \geq \frac{f(x) - f\left(x - \delta\hat{g}_k\right)}{\delta} &\geq \frac{f(x) - f(x - \delta\cdot\hat{\zeta}_k)}{\delta} - L\|\hat{g}_k - \hat{\zeta}_k\|_2 \\
&= \frac{1}{\delta}\int_0^\delta \langle \nabla f(x - s\hat{\zeta}_k), \hat{\zeta}_k\rangle\, ds - L\|\hat{g}_k - \hat{\zeta}_k\|_2 \\
&\geq \frac{1}{\delta}\int_0^\delta \langle \nabla f(x - s\hat{\zeta}_k), \hat{g}_k\rangle\, ds - 2L\|\hat{g}_k - \hat{\zeta}_k\|_2 \\
&= \mathbb{E}_k\langle \nabla f(y_k), \hat{g}_k\rangle - 2L\|\hat{g}_k - \hat{\zeta}_k\|_2.
\end{aligned}$$

Rearranging yields $\mathbb{E}_k\langle \nabla f(y_k), \hat{g}_k\rangle \leq \frac{1}{4}\|g_k\|_2 + 2L\|\hat{g}_k - \hat{\zeta}_k\|$. Simple algebra shows $\|\hat{g}_k - \hat{\zeta}_k\|_2^2 \leq 2(1 - \sqrt{1 - r^2/\|g_k\|_2^2}) \leq \frac{\|g_k\|_2^2}{64L^2}$. Therefore, we infer that $\mathbb{E}_k\langle \nabla f(y_k), \hat{g}_k\rangle < \frac{1}{2}\|g_k\|_2$. Lemma 5.2.3 then guarantees that

$$\mathbb{E}_k[\|g_{k+1}\|_2^2 1_{\tau > k}] \leq \left(\|g_k\|_2^2 - \frac{\|g_k\|_2^4}{16L^2}\right) 1_{\tau > k}.$$

Define $b_k := \|g_k\|_2^2 1_{\tau > k}$ for all $k \geq 0$. Then the tower rule for expectations yields

$$\mathbb{E}b_{k+1} \leq \mathbb{E}[\|g_{k+1}\|_2^2 1_{\tau > k}] \leq \mathbb{E}\left[\left(1 - \frac{b_k}{16L^2}\right) b_k\right] \leq \left(1 - \frac{\mathbb{E}b_k}{16L^2}\right)\mathbb{E}b_k,$$

by Jensen's inequality applied to the concave function $t \mapsto (1 - t/16L^2)t$. Setting $a_k = \mathbb{E}b_k/L^2$, this inequality becomes $a_{k+1} \leq a_k - a_k^2/16$, which, upon rearranging, yields $\frac{1}{a_{k+1}} \geq \frac{1}{a_k(1 - a_k/16)} \geq \frac{1}{a_k} + \frac{1}{16}$.

Iterating the recursion and taking into account $a_0 \le 1$ completes the proof. □

Corollary 12. `MinNorm`(x, δ, ϵ) *terminates in at most* $\left\lceil \frac{64L^2}{\epsilon^2} \right\rceil \cdot \lceil 2\log(1/\gamma) \rceil$ *iterations with probability at least* $1-\gamma$*, where we define the stopping time* $\tau \overset{def}{=} \inf\{k\colon f(x - \delta\widehat{g_k}) < f(x) - \delta\|g_k\|_2/4 \text{ or } \|g_k\|_2 \le \epsilon\}$.

Proof. Notice that when $k \ge \frac{64L^2}{\epsilon^2}$, we have, by Theorem 5.2.5, that

$$\Pr(\tau > k) \le \Pr(\|g_k\|_2 1_{\tau > k} \ge \epsilon) \le \frac{16L^2}{(16+k)\epsilon^2} \le \frac{1}{4}.$$

Similarly, for all $i \in \mathbb{N}$, we have $\Pr(\tau > ik \mid \tau > (i-1)k) \le 1/4$. Therefore,

$$\Pr(\tau > ik) = \Pr(\tau > ik \mid \tau > (i-1)k)\Pr(\tau > (i-1)k) \le \frac{1}{4}\Pr(\tau > (i-1)k) \le \frac{1}{4^i}.$$

Consequently, we have $\Pr(\tau > ik) \le \frac{1}{4^i} \le \gamma$ whenever $i \ge \log(1/\gamma)/\log(4)$, as desired. □

D.2 Implementation of The Oracles

In this section, we show how to convert (5.3.2) into a deterministic guarantee.

Lemma 16. *Fix a unit vector* $\hat{g} \in \mathbb{R}^d$ *and let* $z \in \mathbb{R}^d$ *be a random vector satisfying* $\mathbb{E}\langle \nabla f(z), \hat{g}\rangle \le \frac{\epsilon}{3}$*. Let* $z_1, \dots, z_k$ *be i.i.d realizations of* z *with* $k = \left\lceil \frac{36L}{\epsilon} \right\rceil \cdot \left\lceil \frac{\log(1/\gamma)}{\log(4)} \right\rceil$*. Then with probability at least* $1-\gamma$*, one of the samples* z_i *satisfies* $\langle \nabla f(z_i), \hat{g}\rangle \le \frac{\epsilon}{2}$.

Proof. Define the random variable $Y \overset{\text{def}}{=} \langle \nabla f(z), \hat{g}\rangle$, and use $p \overset{\text{def}}{=} \Pr\left[Y \le \frac{\epsilon}{2}\right]$. We note that

$$\mathbb{E}[Y] = p \cdot \mathbb{E}[Y \mid Y \le \frac{\epsilon}{2}] + (1-p) \cdot \mathbb{E}[Y \mid Y > \frac{\epsilon}{2}].$$

Rearranging the terms and using $\mathbb{E}[Y] \le \epsilon/3$ gives

$$p \cdot \left(\mathbb{E}[Y \mid Y > \frac{\epsilon}{2}] - \mathbb{E}[Y \mid Y \le \frac{\epsilon}{2}]\right) \ge \frac{\epsilon}{6}.$$

Finally, taking into account that f is L-Lipschitz, we deduce $|Y| \le L$, which further implies $p \ge \frac{\epsilon}{12L}$. The result follows immediately. □

Lemma 17. *Let* $f\colon \mathbb{R}^d \to \mathbb{R}$ *be an* L*-Lipschitz continuous and* ρ*-weakly convex function. Fix a point* x *and a unit vector* $\hat{g} \in \mathbb{R}^d$ *such that* f *is differentiable almost everywhere on the line segment* $[x, y]$*, where* $y \overset{def}{=} x - \delta\hat{g}$*. Suppose that a random vector* z *sampled uniformly from* $[x, y]$ *satisfies* $\mathbb{E}_z\langle \nabla f(z), \hat{g}\rangle \le \frac{\epsilon}{3}$*. Then, Algorithm D.2.1 finds* $\bar{z} \in \mathbb{R}^d$ *such that* $\langle \nabla f(\bar{z}), \hat{g}\rangle \le \frac{\epsilon}{2}$ *using* $3\log(12\delta\rho/\epsilon)$ *function evaluations of* f.

Proof. Define the new function $h : [0,1] \to \mathbb{R}$ by $h(t) = \langle \nabla f(x + t(y-x)), \hat{g}\rangle$. Clearly, we have

$$\frac{\epsilon}{3} \ge \mathbb{E}[h(t)] = \frac{1}{2}\underbrace{\mathbb{E}[h(t) \mid t \le 0.5]}_{P_{\le}} + \frac{1}{2}\underbrace{\mathbb{E}[h(t) \mid t > 0.5]}_{P_{>}}.$$

Algorithm D.2.1 Binary Search for $\bar{z}$

Input. Line Segment $[x, y = x - \delta\hat{g}]$
Let $[a, b] = [0, 1]$
while $b - a > \frac{\epsilon}{6\delta\rho}$ **do**
  if $f(x - a\delta\hat{g}) - f(x - \frac{a+b}{2}\delta\hat{g}) \leq f(x - \frac{a+b}{2}\delta\hat{g}) - f(x - b\delta\hat{g})$ **then**
    Let $[a, b] \leftarrow [a, \frac{a+b}{2}]$
  else
    Let $[a, b] \leftarrow [\frac{a+b}{2}, b]$
  end if
end while
Return $x - a\delta\hat{g}$

Therefore, it cannot be the case that both $P_{\leq}$ and $P_{>}$ are larger than $\epsilon/3$, and at least one of them must be at most $\epsilon/3$. The fundamental theorem of calculus directly implies that

$$P_{\leq} = \mathbb{E}[h(t)|t \leq 0.5] = \mathbb{E}_{t\sim[0,1/2]}\langle \nabla f(x + t(y - x)), \hat{g}\rangle = \frac{2(f(x) - f(x - \frac{\delta}{2}\hat{g}))}{\delta}$$

and $P_{>} = \frac{2(f(x-\frac{\delta}{2}\hat{g})-f(y))}{\delta}$. Therefore, with three function evaluations we may determine which of the two alternatives holds. Repeating this procedure $\log(12\delta\rho/\epsilon)$ times, each time shrinking the interval by half, we can find an interval $[a, b] \subset [0, 1]$ such that $b - a \leq \frac{\epsilon}{6\delta\rho}$ and $\mathbb{E}_{t\in[a,b]}h(t) \leq \frac{\epsilon}{3}$. We then have

$$h(\bar{t}) - \mathbb{E}h(t) = \frac{1}{\delta}\mathbb{E}_{t\in[a,b]}\langle \nabla f(x + \bar{t}(y - x)) - \nabla f(x + t(y - x)), x - y\rangle \leq \mathbb{E}_{t\in[a,b]}\frac{\bar{t} - t}{\delta}\rho\|y - x\|^2 \leq \frac{\epsilon}{6},$$

where the first step is by plugging in the definition of h and $\widehat{g}$, the second step is by ρ-weak convexity of f (which in turn implies ρ-weak monotonicity of ∇f), and the final step is by plugging in the bound $b - a \leq \frac{\epsilon}{\delta\epsilon}$. We thus conclude $h(\bar{t}) \leq \frac{\epsilon}{3} + \frac{\epsilon}{6} = \frac{\epsilon}{2}$ as claimed. □

Appendix E

Appendix for Chapter 6

E.1 Appendix: Omitted Technical Details

In this section, we provide proofs for all the claims made in the main body of the chapter, additional supporting propositions and lemmas, and also omitted details of the implementation of our algorithm. The proofs are provided in the order in which the corresponding statement appears in the main body. In Section E.1.1, we prove the properties of our problem. We again emphasize that these properties are crucial to achieving our goal of a scale-invariant algorithm for (P). Section E.1.2 contains the proofs of all our results pertaining to the convergence analysis of Algorithm 6.3.1, with proofs of the growth rate of the scalar sequences $\{a_i\}$, $\{a_i^k\}$, and A_k grouped separately in Section E.1.3, owing to their more technical nature.

E.1.1 Omitted Proof from Section 6.2: Properties of Our Objective

Proposition 6.2.1. *Given $f : \mathbb{R}^n_+ \to \mathbb{R}$ as defined in (6.2.2) and $\mathbf{x}^\star \in \operatorname{argmin}_{\mathbf{x}\in\mathbb{R}^n_+} f(\mathbf{x})$, the following statements all hold.*

a) $\nabla f(\mathbf{x}^\star) \geq \mathbf{0}$.
b) $f(\mathbf{x}^\star) = -\frac{1}{2}\|\mathbf{A}\mathbf{x}^\star\|_2^2 = -\frac{1}{2}\mathbf{1}^\top\mathbf{x}^\star$.
c) for all $j \in [n]$, we have $x_j^\star \in \left[0, \frac{1}{\|\mathbf{A}_{:j}\|_2^2}\right]$.
d) $-\frac{1}{2}\sum_{j\in[n]} \frac{1}{\|\mathbf{A}_{:j}\|_2^2} \leq f(\mathbf{x}^\star) \leq -\frac{1}{2\min_{j\in[n]}\|\mathbf{A}_{:j}\|_2^2}$.

Proof. We recall the first-order optimality condition stated in Inequality 6.2.1: for all $\mathbf{x} \geq \mathbf{0}$, we have $\langle \nabla f(\mathbf{x}^\star), \mathbf{x} - \mathbf{x}^\star\rangle \geq 0$; we repeatedly invoke this inequality in the proof below.

1. Suppose there exists a coordinate j at which Proposition 6.2.1 (a) does not hold and instead, we have $\nabla_j f(\mathbf{x}^\star) < 0$. Consider $\mathbf{x} \geq \mathbf{0}$ such that $x_i = x_i^\star$ for all $i \neq j$ and let $x_j = x_j^\star + \epsilon$ for some $\epsilon > 0$. Then, Inequality 6.2.1 becomes $\nabla_j f(\mathbf{x}^\star)\cdot\epsilon \geq 0$. Under the assumption $\nabla_j f(\mathbf{x}^\star) < 0$, this is an invalid inequality, thus contradicting our assumption.

2. From Proposition 6.2.1 (a), we know that $\nabla f(\mathbf{x}^\star) \geq \mathbf{0}$. If $\nabla_i f(\mathbf{x}^\star) > 0$, and if $x_i^\star > 0$, then by picking a vector $\mathbf{x}$ such that $x_j = x_j^\star$ for $j \neq i$ and $x_i = x_i^\star - \gamma$ for any $\gamma \in (0, x_i^\star)$, we violate Inequality 6.2.1. Therefore it must be the case that if $\nabla_i f(\mathbf{x}^\star) > 0$, then $x_i^\star = 0$. Thus we have

$$0 = \langle \mathbf{x}^\star, \nabla f(\mathbf{x}^\star)\rangle = \langle \mathbf{x}^\star, \mathbf{A}^\top\mathbf{A}\mathbf{x}^\star - \mathbf{1}\rangle.$$

Therefore, $f(\mathbf{x}^\star) = \frac{1}{2}\langle \mathbf{x}^\star, \mathbf{A}^\top\mathbf{A}\mathbf{x}^\star\rangle - \mathbf{1}^\top\mathbf{x}^\star = -\frac{1}{2}\langle \mathbf{x}^\star, \mathbf{A}^\top\mathbf{A}\mathbf{x}^\star\rangle = -\frac{1}{2}\mathbf{1}^\top\mathbf{x}^\star$.

3. From the proof of Proposition 6.2.1 (b), we have $\langle \mathbf{x}^\star, \nabla f(\mathbf{x}^\star)\rangle = 0$. We also have $\mathbf{x}^\star \geq \mathbf{0}$ and, from Proposition 6.2.1 (a), that $\nabla f(\mathbf{x}^\star) \geq \mathbf{0}$. Therefore, if $\mathbf{x}_i^\star > 0$ for some coordinate i then it must be that $\nabla_i f(\mathbf{x}^\star) = 0$. That is, $1 = \langle \mathbf{A}_{:i}, \mathbf{A}\mathbf{x}^\star\rangle$. Combining this equality with the fact that $\mathbf{A}$ and $\mathbf{x}^\star$ are both coordinate-wise non-negative gives

$$1 = \langle \mathbf{A}_{:i}, \mathbf{A}\mathbf{x}^\star\rangle \geq \langle \mathbf{A}_{:i}, \mathbf{A}_{:i}x_i^\star\rangle,$$

which implies $x_i^\star \leq \frac{1}{\|\mathbf{A}_{:i}\|_2^2}$ for all coordinates i.

4. The lower bound follows immediately by combining Proposition 6.2.1 (b) and Proposition 6.2.1 (c). For the upper bound, we need to find *a* feasible point $\widehat{\mathbf{x}}$ and compute the function value at $\widehat{\mathbf{x}}$, since $f(\mathbf{x}^\star) = \min_{\mathbf{y}\geq\mathbf{0}} f(\mathbf{y}) \leq f(\widehat{\mathbf{x}})$. Let $\widehat{\mathbf{x}} = \gamma\mathbf{e}_k$ for some $\gamma > 0$. Then,

$$f(\widehat{\mathbf{x}}) = \frac{1}{2}\gamma^2\|\mathbf{A}_{:k}\|_2^2 - \gamma.$$

Let $\gamma = \frac{1}{\|\mathbf{A}_{:k}\|_2^2}$. Then, $f(\widehat{\mathbf{x}}) = -\frac{1}{2\|\mathbf{A}_{:k}\|_2^2}$. We pick $k = \arg\min_{i\in[n]} \|\mathbf{A}_{:i}\|_2$, therefore $f(\mathbf{x}^\star) \leq -\frac{1}{2\min_{i\in[n]}\|\mathbf{A}_{:i}\|_2^2}$ as claimed.

□

E.1.2 Omitted Proofs from Section 6.3: Analysis of Algorithm

Proofs from Section 6.3.1: Results on Upper and Lower Estimates

We first show the results stating U_k and L_k are indeed valid upper and lower (respectively) estimates of the Lagrangian.

Lemma 6.3.1. *For U_k as defined in Equation (6.3.5), Lagrangian defined in Equation (6.2.3) and $\widetilde{\mathbf{x}}_k \in \mathbb{R}_+^n$ in Equation (6.3.3), we have, for all $\mathbf{y} \in \mathbb{R}^m$, the upper bound $\mathrm{U}_k(\mathbf{y}) \geq \mathcal{L}(\widetilde{\mathbf{x}}_k, \mathbf{y})$.*

Proof. By evaluating the Lagrangian described by Equation (6.2.3) at $\mathbf{x} = \widetilde{\mathbf{x}}_k$, and by definition of ψ_k from Equation (6.3.4), we obtain the following upper bound on the Lagrangian at $(\widetilde{\mathbf{x}}_k, \mathbf{y})$.

$$\begin{aligned}
\mathcal{L}(\widetilde{\mathbf{x}}_k, \mathbf{y}) &= \langle \mathbf{A}\widetilde{\mathbf{x}}_k, \mathbf{y}\rangle - \frac{1}{2}\|\mathbf{y}\|_2^2 - \mathbf{1}^\top\widetilde{\mathbf{x}}_k \\
&= \frac{1}{A_k}\sum_{i\in[k]} a_i^k\left[\langle \mathbf{A}\mathbf{x}_i, \mathbf{y}\rangle - \frac{1}{2}\|\mathbf{y}\|^2 - \mathbf{1}^\top\mathbf{x}_i\right] = \frac{1}{A_k}\psi_k(\mathbf{y}) \\
&\leq \frac{1}{A_k}\psi_k(\mathbf{y}_k) - \frac{1}{2}\|\mathbf{y} - \mathbf{y}_k\|_2^2 = \mathrm{U}_k(\mathbf{y}),
\end{aligned}$$

where the final steps are by strong convexity of ψ_k and by Equation (6.3.5). □

We emphasize that $\mathbf{y}$ here can be random since this is a deterministic statement.

Lemma 6.3.2. *For L_k defined in Equation (6.3.12), for the Lagrangian in Equation (6.2.3) and $\widetilde{\mathbf{y}}_k$ in Equation (6.3.3), we have, for a fixed $\mathbf{u} \in \mathcal{X}$, the lower bound $\mathbb{E}\mathcal{L}(\mathbf{u}, \widetilde{\mathbf{y}}_k) \geq \mathbb{E}\mathrm{L}_k(\mathbf{u})$, where the expectation is with respect to all the random choices of coordinates in Algorithm 6.3.1.*

Proof. First, evaluating Equation (6.2.3) at $\widetilde{\mathbf{y}}_k$ gives

$$\mathcal{L}(\mathbf{u}, \widetilde{\mathbf{y}}_k) = \langle \mathbf{A}\mathbf{u}, \widetilde{\mathbf{y}}_k \rangle - \mathbf{1}^\top \mathbf{u} - \frac{1}{2}\|\widetilde{\mathbf{y}}_k\|_2^2.$$

Taking the expectation on both sides, applying the definition of $\widetilde{\mathbf{y}}_k$, convexity of $\frac{1}{2}\|\cdot\|^2$, and Jensen's inequality, and adding and subtracting $\frac{1}{A_k}\mathbb{E}\sum_{i\in[k]} a_i\langle \mathbf{A}\mathbf{x}, \overline{\mathbf{y}}_{i-1}\rangle + \frac{1}{A_k}\phi_0(\mathbf{u})$ gives

$$\begin{aligned}
\mathbb{E}\mathcal{L}(\mathbf{u}, \widetilde{\mathbf{y}}_k) &\geq \frac{1}{A_k}\mathbb{E}\left[\sum_{i\in[k]} a_i\left[\langle \mathbf{A}\mathbf{u}, \mathbf{y}_i\rangle - \mathbf{1}^\top\mathbf{u} - \frac{1}{2}\|\mathbf{y}_i\|_2^2\right]\right] \\
&= \frac{1}{A_k}\mathbb{E}\left[\sum_{i\in[k]} a_i\left[\langle \mathbf{A}\mathbf{u}, \overline{\mathbf{y}}_{i-1}\rangle - \mathbf{1}^\top\mathbf{u} - \frac{1}{2}\|\mathbf{y}_i\|_2^2\right]\right] + \frac{1}{A_k}\mathbb{E}\left[\phi_0(\mathbf{u})\right] - \frac{1}{A_k}\mathbb{E}\left[\phi_0(\mathbf{u})\right] \\
&\quad + \frac{1}{A_k}\mathbb{E}\left[\sum_{i\in[k]} a_i\langle \mathbf{A}\mathbf{u}, \mathbf{y}_i - \overline{\mathbf{y}}_{i-1}\rangle\right].
\end{aligned}$$

We continue the analysis as

$$\begin{aligned}
\mathbb{E}\mathcal{L}(\mathbf{u}, \widetilde{\mathbf{y}}_k) &\geq \frac{1}{A_k}\mathbb{E}\left[\phi_k(\mathbf{u})\right] - \frac{1}{A_k}\mathbb{E}\left[\phi_0(\mathbf{u})\right] + \frac{1}{A_k}\sum_{i\in[k]} a_i\mathbb{E}\langle \mathbf{A}\mathbf{u}, \mathbf{y}_i - \overline{\mathbf{y}}_{i-1}\rangle - \frac{1}{2A_k}\mathbb{E}\sum_{i\in[k]} a_i\|\mathbf{y}_i\|_2^2 \\
&\geq \frac{1}{A_k}\mathbb{E}\left[\phi_k(\mathbf{x}_k)\right] + \mathbb{E}\left[\frac{1}{2A_k}\|\mathbf{u} - \mathbf{x}_k\|_{\mathbf{\Lambda}}^2\right] - \frac{1}{A_k}\mathbb{E}[\phi_0(\mathbf{u})] + \frac{1}{A_k}\mathbb{E}\left[\sum_{i\in[k]} a_i\langle \mathbf{A}\mathbf{u}, \mathbf{y}_i - \overline{\mathbf{y}}_{i-1}\rangle\right] \\
&\quad - \frac{1}{2A_k}\mathbb{E}\sum_{i\in[k]} a_i\|\mathbf{y}_i\|_2^2 \\
&= \mathbb{E}\mathrm{L}_k(\mathbf{u}),
\end{aligned}$$

the first step comes from Equation (6.3.8), the second step comes from Equation (6.3.11), and the final step comes from Equation (6.3.12). □

Proofs from Section 6.3.2

We now describe three technical propositions that bound terms that show up in the proof of our result on bounding the scaled gap estimate.

Proposition E.1.1. *For ψ_k defined in Equation (6.3.4), with $\{a_i^k\}$ defined in Equation (6.3.2), we have for all $k \geq 1$,*

$$\begin{aligned}
\psi_k(\mathbf{y}_k) - \psi_{k-1}(\mathbf{y}_{k-1}) \leq a_k^k\left\{\langle \mathbf{y}_k, \mathbf{A}\mathbf{x}_k\rangle - \frac{1}{2}\|\mathbf{y}_k\|_2^2 - \mathbf{1}^\top\mathbf{x}_k\right\} + \sum_{i=1}^{k-1}(a_i^k - a_i^{k-1})\left[\langle \mathbf{y}_k, \mathbf{A}\mathbf{x}_i\rangle - \frac{1}{2}\|\mathbf{y}_k\|_2^2 - \mathbf{1}^\top\mathbf{x}_i\right] \\
- \frac{A_{k-1}}{2}\|\mathbf{y}_k - \mathbf{y}_{k-1}\|_2^2.
\end{aligned}$$

Proof. Evaluating ψ_k and ψ_{k-1} as defined in Equation (6.3.4) at $\mathbf{y}_k$ and subtracting, we have

$$\psi_k(\mathbf{y}_k) - \psi_{k-1}(\mathbf{y}_k) = a_k\langle \mathbf{A}^\top\mathbf{y}_k - \mathbf{1}, n\mathbf{x}_k - (n-1)\mathbf{x}_{k-1}\rangle - \frac{a_k}{2}\|\mathbf{y}_k\|_2^2. \tag{E.1.1}$$

Next, applying strong convexity of ψ_{k-1} at $\mathbf{y}_k$ and $\mathbf{y}_{k-1}$ while using the fact that $\mathbf{y}_{k-1}$ minimizes ψ_{k-1} gives

$$\psi_{k-1}(\mathbf{y}_k) - \psi_{k-1}(\mathbf{y}_{k-1}) \leq -\frac{1}{2}A_{k-1}\|\mathbf{y}_k - \mathbf{y}_{k-1}\|_2^2. \tag{E.1.2}$$

To complete the proof, it remains to add Equation (E.1.1) and Inequality E.1.2. □

Proposition E.1.2. *The random function $\phi_k : \mathcal{X} \to \mathbb{R}$, $k \geq 2$, defined in Equation (6.3.10) satisfies the following properties, with $\mathbf{x}_k$, $\mathbf{y}_k$, and $\overline{\mathbf{y}}_k$ evolving as per Algorithm 6.3.1.*

a) It is separable in its coordinates: $\phi_k(\mathbf{x}) = \sum_{j\in[n]} \phi_{k,j}(x_j)$, where, for each $j \in [n]$, we define $\phi_{0,j}(x_j) = \frac{\|\mathbf{A}_{:j}\|_2^2}{2}(x_j - [\mathbf{x}_0]_j)^2$, $\phi_{1,j}(x_j) = a_1 x_j(\mathbf{A}^\top \overline{\mathbf{y}}_0 - \mathbf{1})_j + \phi_{0,j}(x_j)$, and for $k \geq 2$,

$$\phi_{k,j}(x_j) = \phi_{k-1,j}(x_j) + na_k \mathbf{1}_{j=j_k}\langle \mathbf{A}^\top \overline{\mathbf{y}}_{k-1} - \mathbf{1}, x_{j_k}\mathbf{e}_{j_k}\rangle. \tag{E.1.3}$$

b) The primal variable $\mathbf{x}_k$ is updated only on the j_k^{th} coordinate in each iteration: $\mathbf{x}_k = \mathbf{x}_{k-1} + \gamma \mathbf{e}_{j_k}$ for some γ, and $[\mathbf{x}_k]_j = [\mathbf{x}_{k-1}]_j$ for $j \neq j_k$.

c) For a fixed $\mathbf{x} \in \mathcal{X}$ and for $k \geq 1$, we have, over all the randomness in the algorithm,

$$\mathbb{E}\left[\phi_k(\mathbf{x})\right] = \mathbb{E}\left[\phi_0(\mathbf{x})\right] + \sum_{i\in[k]} a_i \mathbb{E}\left[\langle \mathbf{A}^\top \overline{\mathbf{y}}_{i-1} - \mathbf{1}, \mathbf{x}\rangle\right]. \tag{E.1.4}$$

Proof. In the statement of Proposition E.1.2 (a), the claim about separability of ϕ_0 and ϕ_1 can be checked just from the definitions of $\phi_{0,j}$ and $\phi_{1,j}$. We prove the claim of coordinate-wise separability for $k \geq 2$ by summing over $j \in [n]$ both sides of Equation (E.1.3). We can compute this sum via following observation, which concludes the proof of Proposition E.1.2 (a).

$$\sum_{j\in[n]} a_k \mathbf{1}_{j=j_k}\langle \mathbf{A}^\top \overline{\mathbf{y}}_{k-1} - \mathbf{1}, x_{j_k}\mathbf{e}_{j_k}\rangle = a_k\langle \mathbf{A}^\top \overline{\mathbf{y}}_{k-1} - \mathbf{1}, x_{j_k}\mathbf{e}_{j_k}\rangle.$$

From Proposition E.1.2 (a), we may therefore define, for $j \neq j_k$,

$$[\mathbf{x}_k]_j = \arg\min_{u\in\mathbb{R}_+} \phi_{k,j}(u) = \arg\min_{u\in\mathbb{R}_+} \phi_{k-1,j}(u) = [\mathbf{x}_{k-1}]_j.$$

Therefore, $[\mathbf{x}_k]_j = [\mathbf{x}_{k-1}]_j$ for all $j \neq j_k$, thus proving Proposition E.1.2 (b). To prove Proposition E.1.2 (c), we use induction on k. The base case holds for $k = 1$ by the definition of $\phi_1(\mathbf{x})$. Let Proposition E.1.2 (c) hold for $k \geq 1$. Then, by the definition of ϕ_k as in Equation (6.3.10), we have

$$\phi_k(\mathbf{x}) = \phi_{k-1}(\mathbf{x}) + na_k\langle \mathbf{A}^\top \overline{\mathbf{y}}_{k-1} - \mathbf{1}, x_{j_k}\mathbf{e}_{j_k}\rangle, \text{ for all } k \geq 2.$$

Let $\mathcal{F}_{k-1}$ be the natural filtration, containing all the randomness in the algorithm up to and including iteration $k-1$. Taking expectations with respect to all the randomness until iteration k and invoking linearity of expectation, the inductive hypothesis, and the tower rule $\mathbb{E}[\,\cdot\,] = \mathbb{E}[\mathbb{E}[\,\cdot\,|\mathcal{F}_{k-1}]]$, we have

$$\begin{aligned}
\mathbb{E}[\phi_k(\mathbf{x})] &= \mathbb{E}[\phi_{k-1}(\mathbf{x})] + na_k\mathbb{E}\left[\left[\mathbb{E}\langle \mathbf{A}^\top \overline{\mathbf{y}}_{k-1} - \mathbf{1}, x_{j_k}\mathbf{e}_{j_k}\rangle|\mathcal{F}_{k-1}\right]\right] \\
&= \mathbb{E}\left[\phi_0(\mathbf{x})\right] + \sum_{i\in[k-1]} a_i\mathbb{E}\left[\langle \mathbf{A}^\top \overline{\mathbf{y}}_{i-1} - \mathbf{1}, \mathbf{x}\rangle\right] + a_k\mathbb{E}\langle \mathbf{A}^\top \overline{\mathbf{y}}_{k-1} - \mathbf{1}, \mathbf{x}\rangle \\
&= \mathbb{E}\left[\phi_0(\mathbf{x})\right] + \sum_{i\in[k]} a_i\mathbb{E}\left[\langle \mathbf{A}^\top \overline{\mathbf{y}}_{i-1} - \mathbf{1}, \mathbf{x}\rangle\right],
\end{aligned}$$

which finishes the proof of Proposition E.1.2 (c). □

Proposition E.1.3. *For all $k \geq 2$, the random function $\phi_k : \mathcal{X} \to \mathbb{R}$, $k \geq 2$, defined in Equation (6.3.10) satisfies the following inequality, where $\mathbf{x}_k$ and $\overline{\mathbf{y}}_k$ evolve according to Algorithm 6.3.1.*

$$\phi_k(\mathbf{x}_k) - \phi_{k-1}(\mathbf{x}_{k-1}) \geq a_k \left(n\langle \mathbf{A}^\top \overline{\mathbf{y}}_{k-1} - \mathbf{1}, [\mathbf{x}_k]_{j_k} \mathbf{e}_{j_k} \rangle \right) + \frac{1}{2} \|\mathbf{x}_k - \mathbf{x}_{k-1}\|_{\mathbf{\Lambda}}^2.$$

Proof. We have, using $\mathbf{x} = \mathbf{x}_k$ in Equation (6.3.10), that

$$\phi_k(\mathbf{x}_k) - \phi_{k-1}(\mathbf{x}_k) = n a_k \langle \mathbf{A}^\top \overline{\mathbf{y}}_{k-1} - \mathbf{1}, [\mathbf{x}_k]_{j_k} \mathbf{e}_{j_k} \rangle.$$

Applying Equation (6.3.11) to ϕ_{k-1} at $\mathbf{x}_k$ gives

$$\phi_{k-1}(\mathbf{x}_k) - \phi_{k-1}(\mathbf{x}_{k-1}) \geq \frac{1}{2} \|\mathbf{x}_k - \mathbf{x}_{k-1}\|_{\mathbf{\Lambda}}^2.$$

Adding these inequalities completes the proof of the claim. □

We now use the preceding technical results to bound the change in scaled gap.

Lemma 6.3.3. *Consider the iterates $\{\mathbf{x}_k\}$ and $\{\mathbf{y}_k\}$ evolving according to Algorithm 6.3.1. Let $n \geq 2$ and assume that $a_1 = \frac{1}{\sqrt{2} n^{1.5}}$ and $a_1 \geq (n-1) a_2$, while for $k \geq 3$,*

$$a_k \leq \min\left(\frac{n a_{k-1}}{n-1}, \frac{\sqrt{A_{k-1}}}{2n} \right). \tag{6.3.13}$$

Then, for fixed $\mathbf{u} \in \mathcal{X}$, any $\mathbf{v} \in \mathbb{R}^m$, and all $k \geq 2$, the gap estimate $\mathrm{G}_k = \mathrm{U}_k - \mathrm{L}_k$ satisfies

$$\begin{aligned}
&\mathbb{E}(A_k G_k(\mathbf{x}, \mathbf{y}) - A_{k-1} G_{k-1}(\mathbf{x}, \mathbf{y})) \\
&\leq -\mathbb{E}\left(\frac{A_k}{2} \|\mathbf{y} - \mathbf{y}_k\|_2^2 - \frac{A_{k-1}}{2} \|\mathbf{y} - \mathbf{y}_{k-1}\|_2^2 \right) - \frac{1}{2} \mathbb{E} \|\mathbf{x} - \mathbf{x}_k\|_{\mathbf{\Lambda}}^2 + \frac{1}{2} \mathbb{E} \|\mathbf{x} - \mathbf{x}_{k-1}\|_{\mathbf{\Lambda}}^2 \\
&\quad - a_k \mathbb{E} \langle \mathbf{A}(\mathbf{x} - \mathbf{x}_k), \mathbf{y}_k - \mathbf{y}_{k-1} \rangle + a_{k-1} \mathbb{E} \langle \mathbf{A}(\mathbf{x} - \mathbf{x}_{k-1}), \mathbf{y}_{k-1} - \mathbf{y}_{k-2} \rangle \\
&\quad - \frac{1}{4} A_{k-1} \mathbb{E} \|\mathbf{y}_k - \mathbf{y}_{k-1}\|_2^2 + \frac{1}{4} A_{k-2} \mathbb{E} \|\mathbf{y}_{k-1} - \mathbf{y}_{k-2}\|_2^2.
\end{aligned}$$

Proof. Using $\mathrm{G}_k = \mathrm{U}_k - \mathrm{L}_k$, U_k from Equation (6.3.5), and L_k from Equation (6.3.12), we have

$$\begin{aligned}
A_k \mathrm{G}_k(\mathbf{u}, \mathbf{v}) = \psi_k(\mathbf{y}_k) - \phi_k(\mathbf{x}_k) + \phi_0(\mathbf{u}) \\
- \frac{A_k}{2} \|\mathbf{v} - \mathbf{y}_k\|_2^2 - \frac{1}{2} \|\mathbf{u} - \mathbf{x}_k\|_{\mathbf{\Lambda}}^2 - \sum_{i \in [k]} a_i \langle \mathbf{A}\mathbf{u}, \mathbf{y}_i - \overline{\mathbf{y}}_{i-1} \rangle + \sum_{i \in [k]} \frac{a_i}{2} \|\mathbf{y}_i\|_2^2.
\end{aligned}$$

Therefore, the difference in scaled gap between successive iterations is

$$\begin{aligned}
A_k \mathrm{G}_k(\mathbf{u}, \mathbf{v}) - A_{k-1} \mathrm{G}_{k-1}(\mathbf{u}, \mathbf{v}) = \; & [\psi_k(\mathbf{y}_k) - \psi_{k-1}(\mathbf{y}_{k-1})] - \left[\phi_k(\mathbf{x}_k) - \phi_{k-1}(\mathbf{x}_{k-1}) \right] \\
& - \frac{A_k}{2} \|\mathbf{v} - \mathbf{y}_k\|_2^2 + \frac{A_{k-1}}{2} \|\mathbf{v} - \mathbf{y}_{k-1}\|_2^2 \\
& - \frac{1}{2} \|\mathbf{u} - \mathbf{x}_k\|_{\mathbf{\Lambda}}^2 + \frac{1}{2} \|\mathbf{u} - \mathbf{x}_{k-1}\|_{\mathbf{\Lambda}}^2 \\
& - a_k \langle \mathbf{A}\mathbf{u}, \mathbf{y}_k - \overline{\mathbf{y}}_{k-1} \rangle + \frac{a_k}{2} \|\mathbf{y}_k\|_2^2.
\end{aligned} \tag{E.1.5}$$

Based on the above expression, to prove the lemma, it suffices to bound the expectation of

$$T_k(\mathbf{u}) \stackrel{\text{def}}{=} [\psi_k(\mathbf{y}_k) - \psi_{k-1}(\mathbf{y}_{k-1})] - \left[\phi_k(\mathbf{x}_k) - \phi_{k-1}(\mathbf{x}_{k-1})\right] - a_k\langle \mathbf{A}\mathbf{u}, \mathbf{y}_k - \overline{\mathbf{y}}_{k-1}\rangle + \frac{a_k}{2}\|\mathbf{y}_k\|_2^2. \tag{E.1.6}$$

First, we take expectations on both sides of the inequality in Proposition E.1.3 by invoking $\mathbb{E}[\,\cdot\,] = \mathbb{E}[\mathbb{E}[\,\cdot\,|\mathcal{F}_{k-1}]]$ as before, where $\mathcal{F}_{k-1}$ denotes the natural filtration. By using the fact that $\mathbf{x}_{k-1}$ is updated only at coordinate j_k (as stated in Proposition E.1.2 (b)), we observe the following for the term from the right hand side of Proposition E.1.3.

$$\begin{aligned}\mathbb{E}\left[\langle \mathbf{A}^\top\overline{\mathbf{y}}_{k-1} - \mathbf{1}, [\mathbf{x}_k]_{j_k}\mathbf{e}_{j_k}\rangle\right] =& \mathbb{E}\left[\langle \mathbf{A}^\top\overline{\mathbf{y}}_{k-1} - \mathbf{1}, \mathbf{x}_k - \mathbf{x}_{k-1}\rangle\right] + \mathbb{E}\left[\mathbb{E}\left[\langle \mathbf{A}^\top\overline{\mathbf{y}}_{k-1} - \mathbf{1}, [\mathbf{x}_{k-1}]_{j_k}\mathbf{e}_{j_k}\rangle|\mathcal{F}_{k-1}\right]\right] \\ =& \mathbb{E}\left[\langle \mathbf{A}^\top\overline{\mathbf{y}}_{k-1} - \mathbf{1}, \mathbf{x}_k - \mathbf{x}_{k-1}\rangle\right] + \frac{1}{n}\mathbb{E}\left[\langle \mathbf{A}^\top\overline{\mathbf{y}}_{k-1} - \mathbf{1}, \mathbf{x}_{k-1}\rangle\right] \\ =& \mathbb{E}\left[\langle \mathbf{A}^\top\overline{\mathbf{y}}_{k-1} - \mathbf{1}, \mathbf{x}_k - (1-1/n)\,\mathbf{x}_{k-1}\rangle\right].\end{aligned} \tag{E.1.7}$$

Therefore, we have from Proposition E.1.3 and scaling Equation (E.1.7) by $-na_k$ that

$$\begin{aligned}-\mathbb{E}\left[\phi_k(\mathbf{x}_k) - \phi_{k-1}(\mathbf{x}_{k-1})\right] \leq & -\frac{1}{2}\mathbb{E}\left[\|\mathbf{x}_k - \mathbf{x}_{k-1}\|_{\mathbf{\Lambda}}^2\right] \\ & - a_k\mathbb{E}\left[\langle \mathbf{A}^\top\overline{\mathbf{y}}_{k-1} - \mathbf{1}, n\mathbf{x}_k - (n-1)\mathbf{x}_{k-1}\rangle\right].\end{aligned} \tag{E.1.8}$$

We now bound the expectation of the term involving differences of ψ_k by taking expectations of both sides of Proposition E.1.1.

$$\begin{aligned}\mathbb{E}\left[\psi_k(\mathbf{y}_k) - \psi_{k-1}(\mathbf{y}_{k-1})\right] \leq & -\frac{A_{k-1}}{2}\mathbb{E}\left[\|\mathbf{y}_k - \mathbf{y}_{k-1}\|_2^2\right] - \frac{a_k}{2}\mathbb{E}\left[\|\mathbf{y}_k\|_2^2\right] \\ & + a_k\mathbb{E}\left[\langle \mathbf{A}^\top\mathbf{y}_k - \mathbf{1}, n\mathbf{x}_k - (n-1)\mathbf{x}_{k-1}\rangle\right].\end{aligned} \tag{E.1.9}$$

By taking expectations on both sides of Equation (E.1.6), we have

$$\mathbb{E}[T_k(\mathbf{u})] = \mathbb{E}\left[\psi_k(\mathbf{y}_k) - \psi_{k-1}(\mathbf{y}_{k-1})\right] - \mathbb{E}\left[\phi_k(\mathbf{x}_k) - \phi_{k-1}(\mathbf{x}_{k-1})\right] - a_k\mathbb{E}\left[\langle \mathbf{A}\mathbf{u}, \mathbf{y}_k - \overline{\mathbf{y}}_{k-1}\rangle\right] + \frac{a_k}{2}\mathbb{E}\left[\|\mathbf{y}_k\|_2^2\right]. \tag{E.1.10}$$

Combining Inequality E.1.8, Inequality E.1.9, and Equation (E.1.10) then gives

$$\mathbb{E}[T_k(\mathbf{u})] \leq -\frac{A_{k-1}}{2}\mathbb{E}\left[\|\mathbf{y}_k - \mathbf{y}_{k-1}\|_2^2\right] - \frac{1}{2}\mathbb{E}\left[\|\mathbf{x}_k - \mathbf{x}_{k-1}\|_{\mathbf{\Lambda}}^2\right] + a_k\mathbb{E}\left[\langle \mathbf{A}^\top(\mathbf{y}_k - \overline{\mathbf{y}}_{k-1}), n\mathbf{x}_k - (n-1)\mathbf{x}_{k-1} - \mathbf{u}\rangle\right]. \tag{E.1.11}$$

Recall that by the assumption in the statement of the lemma,

$$\overline{\mathbf{y}}_{k-1} = \mathbf{y}_{k-1} + \frac{a_{k-1}}{a_k}(\mathbf{y}_{k-1} - \mathbf{y}_{k-2}).$$

Plugging into Equation (E.1.11) and rearranging, we have

$$\begin{aligned}\mathbb{E}[T_k(\mathbf{u})] \leq & -\frac{A_{k-1}}{2}\mathbb{E}\left[\|\mathbf{y}_k - \mathbf{y}_{k-1}\|_2^2\right] - \frac{1}{2}\mathbb{E}\left[\|\mathbf{x}_k - \mathbf{x}_{k-1}\|_{\mathbf{\Lambda}}^2\right] \\ & + (n-1)a_k\mathbb{E}\left[\langle \mathbf{A}^\top(\mathbf{y}_k - \mathbf{y}_{k-1}), \mathbf{x}_k - \mathbf{x}_{k-1}\rangle\right] - na_{k-1}\mathbb{E}\left[\langle \mathbf{A}^\top(\mathbf{y}_{k-1} - \mathbf{y}_{k-2}), \mathbf{x}_k - \mathbf{x}_{k-1}\rangle\right] \\ & + a_k\mathbb{E}\left[\langle \mathbf{A}^\top(\mathbf{y}_k - \mathbf{y}_{k-1}), \mathbf{x}_k - \mathbf{u}\rangle\right] - a_{k-1}\mathbb{E}\left[\langle \mathbf{A}^\top(\mathbf{y}_{k-1} - \mathbf{y}_{k-2}), \mathbf{x}_{k-1} - \mathbf{u}\rangle\right].\end{aligned} \tag{E.1.12}$$

To complete the proof, we need to bound the terms from the first two lines on the right-hand side of Equation (E.1.12). First, observe that, by the coordinate update of $\mathbf{x}_k$ and Young's inequality, $\forall \beta > 0$,

$$\begin{aligned}\langle \mathbf{A}^\top(\mathbf{y}_k - \mathbf{y}_{k-1}), \mathbf{x}_k - \mathbf{x}_{k-1}\rangle &= \langle \mathbf{y}_k - \mathbf{y}_{k-1}, \mathbf{A}(\mathbf{x}_k - \mathbf{x}_{k-1})\rangle \\ &= \langle \mathbf{y}_k - \mathbf{y}_{k-1}, \mathbf{A}_{:j_k}([\mathbf{x}_k]_{j_k} - [\mathbf{x}_{k-1}]_{j_k})\rangle \\ &\leq \frac{\beta}{2}\|\mathbf{y}_k - \mathbf{y}_{k-1}\|_2^2 + \frac{1}{2\beta}\|\mathbf{A}_{:j_k}\|_2^2|[\mathbf{x}_k]_{j_k} - [\mathbf{x}_{k-1}]_{j_k}|^2 \\ &= \frac{\beta}{2}\|\mathbf{y}_k - \mathbf{y}_{k-1}\|_2^2 + \frac{1}{2\beta}\|\mathbf{x}_k - \mathbf{x}_{k-1}\|_{\mathbf{\Lambda}}^2. \end{aligned} \tag{E.1.13}$$

By the same token, $\forall \gamma > 0$,

$$-\langle \mathbf{A}^\top(\mathbf{y}_{k-1} - \mathbf{y}_{k-2}), \mathbf{x}_k - \mathbf{x}_{k-1}\rangle \leq \frac{\gamma}{2}\|\mathbf{y}_{k-1} - \mathbf{y}_{k-2}\|_2^2 + \frac{1}{2\gamma}\|\mathbf{x}_k - \mathbf{x}_{k-1}\|_{\mathbf{\Lambda}}^2. \tag{E.1.14}$$

Recalling that, by the choice of step sizes, $(n-1)a_k \leq na_{k-1}$ and $a_k \leq \frac{\sqrt{A_{k-1}}}{2n}$, we can verify that for $\beta = 2(n-1)a_k$ and $\gamma = 2na_{k-1}$, the following inequalities hold:

$$\begin{aligned}(n-1)a_k\beta - A_{k-1} &\leq -\frac{A_{k-1}}{2}, \\ \frac{(n-1)a_k}{\beta} + \frac{na_{k-1}}{\gamma} &\leq 1. \end{aligned} \tag{E.1.15}$$

Combining Equations (E.1.12)–(E.1.15),

$$\begin{aligned}\mathbb{E}[T_k(\mathbf{u})] \leq\ & -\frac{A_{k-1}}{4}\mathbb{E}[\|\mathbf{y}_k - \mathbf{y}_{k-1}\|_2^2] + n^2{a_{k-1}}^2\mathbb{E}[\|\mathbf{y}_{k-1} - \mathbf{y}_{k-2}\|_2^2] \\ & + a_k\mathbb{E}\left[\langle \mathbf{A}^\top(\mathbf{y}_k - \mathbf{y}_{k-1}), \mathbf{x}_k - \mathbf{u}\rangle\right] - a_{k-1}\mathbb{E}\left[\langle \mathbf{A}^\top(\mathbf{y}_{k-1} - \mathbf{y}_{k-2}), \mathbf{x}_{k-1} - \mathbf{u}\rangle\right]. \end{aligned} \tag{E.1.16}$$

It remains to combine Equation (E.1.5), Equation (E.1.6), and Equation (E.1.16). □

Lemma 6.3.4. *Given a fixed* $\mathbf{u} \in \mathcal{X}$, *any* $\mathbf{v} \in \mathbb{R}^m$, $\bar{\mathbf{y}}_0 = \mathbf{y}_0$, *and* $\mathbf{x}_1$ *and* $\mathbf{y}_1$ *from Algorithm 6.3.1, we have*

$$A_1\mathrm{G}_1(\mathbf{u}, \mathbf{v}) = a_1\langle \mathbf{A}^\top(\mathbf{y}_1 - \mathbf{y}_0), \mathbf{x}_1 - \mathbf{u}\rangle + \phi_0(\mathbf{u}) - \phi_0(\mathbf{x}_1) - \frac{1}{2}\|\mathbf{u} - \mathbf{x}_1\|_{\mathbf{\Lambda}}^2 - \frac{A_1}{2}\|\mathbf{v} - \mathbf{y}_1\|_2^2.$$

Proof. Evaluating Equation (6.3.5) and Equation (6.3.12) at $k = 1$ gives

$$A_1U_1(\mathbf{v}) = \psi_1(\mathbf{y}_1) - \frac{A_1}{2}\|\mathbf{v} - \mathbf{y}_1\|_2^2, \tag{E.1.17}$$

$$A_1L_1(\mathbf{u}) = \phi_1(\mathbf{x}_1) + \frac{1}{2}\|\mathbf{u} - \mathbf{x}_1\|_{\mathbf{\Lambda}}^2 - \phi_0(\mathbf{u}) + a_1\langle \mathbf{A}\mathbf{u}, \mathbf{y}_1 - \bar{\mathbf{y}}_0\rangle - \frac{a_1}{2}\|\mathbf{y}_1\|_2^2. \tag{E.1.18}$$

By definition of ψ_1 from Equation (6.3.4), ϕ_1 from Equation (6.3.7), and the assignment $a_1^1 = a_1$, we have

$$\begin{aligned}\psi_1(\mathbf{y}_1) - \phi_1(\mathbf{x}_1) &= -\frac{a_1}{2}\|\mathbf{y}_1\|_2^2 + a_1\langle \mathbf{y}_1, \mathbf{A}\mathbf{x}_1\rangle - a_1\mathbf{1}^\top\mathbf{x}_1 - \phi_0(\mathbf{x}_1) - a_1\langle \mathbf{A}^\top\bar{\mathbf{y}}_0 - \mathbf{1}, \mathbf{x}_1\rangle \\ &= -\frac{a_1}{2}\|\mathbf{y}_1\|_2^2 + a_1\langle \mathbf{A}(\mathbf{y}_1 - \mathbf{y}_0), \mathbf{x}_1\rangle - \phi_0(\mathbf{x}_1), \end{aligned}$$

where we have used that $\bar{\mathbf{y}}_0 = \mathbf{y}_0$, which holds by assumption. To complete the proof, it remains to subtract Equation (E.1.18) from Equation (E.1.17) and combine with the last equality. □

Theorem 6.3.5. *[Main Result] Assume that $n \geq 4$. Given a matrix $\mathbf{A} \in \mathbb{R}_+^{m\times n}$, $\epsilon > 0$, an arbitrary $\mathbf{x}_0 \in \mathcal{X}$ and $\bar{\mathbf{y}}_0 = \mathbf{y}_0 = \mathbf{A}\mathbf{x}_0$, let $\mathbf{x}_k$ and A_k evolve according to SI-NNLS+ (Algorithm 6.3.1) for $k \geq 1$. For f defined in (6.2.2), define $\mathbf{x}^\star \in \operatorname{argmin}_{\mathbf{x}\geq\mathbf{0}} f(\mathbf{x})$. Then, for all $K \geq 2$, we have*

$$\mathbb{E}\Big[\langle \nabla f(\widetilde{\mathbf{x}}_K), \widetilde{\mathbf{x}}_K - \mathbf{x}^\star\rangle + \frac{1}{2}\|\mathbf{A}(\widetilde{\mathbf{x}}_K - \mathbf{x}^\star)\|^2\Big] \leq \frac{2\phi_0(\mathbf{x}^\star)}{A_K} = \frac{\|\mathbf{x}_0 - \mathbf{x}^\star\|_{\mathbf{\Lambda}}^2}{A_K}.$$

When $K \geq \frac{5}{2}n\log n$, we have $A_K \geq \frac{(K-\frac{5}{2}n\log n)^2}{36n^2}$. If $\phi_0(\mathbf{x}^\star) \leq |f(\mathbf{x}^\star)|$, then for $K \geq \frac{5}{2}n\log n + \frac{6n}{\sqrt{\epsilon}}$, we have $\mathbb{E}[f(\widetilde{\mathbf{x}}_K) - f(\mathbf{x}^\star)] \leq \varepsilon|f(\mathbf{x}^\star)|$. The total cost is $O\Big(nnz(\mathbf{A})\Big(\log n + \frac{1}{\sqrt{\epsilon}}\Big)\Big)$.

Proof. Observe that, by the choice of step sizes, $n^2 a_{k-1}^2 \leq \frac{A_{k-2}}{4}$, $\forall k \geq 3$. Thus, telescoping the bound in Lemma 6.3.3 and combining with Lemma 6.3.4, we have

$$\begin{aligned}\mathbb{E}[A_K G_K(\mathbf{u},\mathbf{v})] \leq& \phi_0(\mathbf{u}) - \frac{A_K}{2}\mathbb{E}[\|\mathbf{v}-\mathbf{y}_K\|_2^2] - \frac{1}{2}\mathbb{E}[\|\mathbf{u}-\mathbf{x}_K\|_{\mathbf{\Lambda}}^2] - a_K\mathbb{E}[\langle \mathbf{A}(\mathbf{u}-\mathbf{x}_K), \mathbf{y}_K - \mathbf{y}_{K-1}\rangle] \\ &- \frac{A_{K-1}}{4}\mathbb{E}[\|\mathbf{y}_K - \mathbf{y}_{K-1}\|_2^2] + n^2 {a_1}^2\mathbb{E}[\|\mathbf{y}_1 - \mathbf{y}_0\|_2^2] - \mathbb{E}[\phi_0(\mathbf{x}_1)].\end{aligned} \tag{E.1.19}$$

We first show how to cancel out the inner product term with the negative quadratic terms. Observe that, $\forall \beta > 0$,

$$\begin{aligned}-a_K\langle \mathbf{A}(\mathbf{u}-\mathbf{x}_K), \mathbf{y}_K - \mathbf{y}_{K-1}\rangle =& -a_K\sum_{j=1}^n (\mathbf{y}_K - \mathbf{y}_{K-1})^\top \mathbf{A}_{:j}(u_j - [\mathbf{x}_K]_j) \\ \leq& a_K\Big(\frac{n\beta}{2}\|\mathbf{y}_K - \mathbf{y}_{K-1}\|_2^2 + \frac{1}{2\beta}\|\mathbf{u}-\mathbf{x}_K\|_{\mathbf{\Lambda}}^2\Big),\end{aligned}$$

where the last line is by Young's inequality. In particular, choosing $\beta = 2a_K$, we have $\frac{1}{2}a_K n\beta = na_K^2$, which is at most $\frac{A_{K-1}}{4}$, by the choice of step sizes in SI-NNLS+. Thus, since $\phi_0(\mathbf{x}_1) = \frac{1}{2}\|\mathbf{x}_1 - \mathbf{x}_0\|_{\mathbf{\Lambda}}^2$, Equation (E.1.19) simplifies to

$$\mathbb{E}[A_K G_K(\mathbf{u},\mathbf{v})] \leq \phi_0(\mathbf{u}) - \frac{A_K}{2}\mathbb{E}[\|\mathbf{v}-\mathbf{y}_K\|_2^2] - \frac{1}{4}\mathbb{E}[\|\mathbf{u}-\mathbf{x}_K\|_{\mathbf{\Lambda}}^2] + n^2{a_1}^2\mathbb{E}[\|\mathbf{y}_1-\mathbf{y}_0\|_2^2] - \mathbb{E}[\phi_0(\mathbf{x}_1)]. \tag{E.1.20}$$

Since $-\frac{1}{4}\mathbb{E}[\|\mathbf{u}-\mathbf{x}_K\|_{\mathbf{\Lambda}}^2] \leq 0$, we can ignore it. Let us now bound $n^2{a_1}^2\|\mathbf{y}_1-\mathbf{y}_0\|_2^2 - \phi_0(\mathbf{x}_1)$. By definition of $\mathbf{x}_1$ and ϕ_1, we have $\mathbf{x}_1 = \mathbf{x}_0 - a_1\mathbf{\Lambda}^{-1}(\mathbf{A}^\top\mathbf{y}_0 - \mathbf{1})$. Further, from Equation (6.3.4) and Equation (6.3.1), as we have $\mathbf{y}_1 = \mathbf{A}\mathbf{x}_1$ and $\mathbf{y}_0 = \mathbf{A}\mathbf{x}_0$, we can simplify the terms to bound as follows.

$$\begin{aligned}n^2a_1^2\|\mathbf{y}_1-\mathbf{y}_0\|_2^2 - \phi_0(\mathbf{x}_1) =& {a_1}^2\Big(n^2{a_1}^2\|\mathbf{A}\mathbf{\Lambda}^{-1}(\mathbf{A}^\top\mathbf{y}_0-\mathbf{1})\|_2^2 - \frac{1}{2}\|\mathbf{\Lambda}^{-1}(\mathbf{A}^\top\mathbf{y}_0-\mathbf{1})\|_{\mathbf{\Lambda}}^2\Big) \\ \leq& {a_1}^2\Big(n^2{a_1}^2\|\mathbf{\Lambda}^{-1/2}\mathbf{A}^\top\mathbf{A}\mathbf{\Lambda}^{-1/2}\|_2\|\mathbf{\Lambda}^{-\frac{1}{2}}(\mathbf{A}^\top\mathbf{y}_0-\mathbf{1})\|_2^2 - \frac{1}{2}\|\mathbf{\Lambda}^{-\frac{1}{2}}(\mathbf{A}^\top\mathbf{y}_0-\mathbf{1})\|_2^2\Big) \\ \leq& {a_1}^2\|\mathbf{\Lambda}^{-\frac{1}{2}}(\mathbf{A}^\top\mathbf{y}_0-\mathbf{1})\|_2^2\Big(n^3{a_1}^2 - \frac{1}{2}\Big),\end{aligned}$$

where the reasoning behind the first inequality follows from the definition of spectral norm and

that $\|\mathbf{A}\mathbf{\Lambda}^{-1/2}\|_2^2 = \lambda_{\max}(\mathbf{\Lambda}^{-1/2}\mathbf{A}^\top\mathbf{A}\mathbf{\Lambda}^{-1/2})$; the last inequality follows as the matrix $\mathbf{\Lambda}^{-1/2}\mathbf{A}^\top\mathbf{A}\mathbf{\Lambda}^{-1/2}$ has all ones on the main diagonal, and thus its trace is at most n, and since it is positive semidefinite, its spectral norm is at most its trace. As $a_1 = \frac{1}{\sqrt{2}n^{1.5}}$, we conclude that $n^2{a_1}^2\|\mathbf{y}_1 - \mathbf{y}_0\|_2^2 - \phi_0(\mathbf{x}_1) \le 0$. Thus, Equation (E.1.20) simplifies to

$$\mathbb{E}[A_K \mathrm{G}_K(\mathbf{u},\mathbf{v})] \le \mathbb{E}[\phi_0(\mathbf{u})] - \frac{A_K}{2}\mathbb{E}[\|\mathbf{v} - \mathbf{y}_K\|_2^2]. \tag{E.1.21}$$

By construction, $\mathrm{Gap}_{\mathcal{L}}^{\mathbf{u},\mathbf{v}}(\widetilde{\mathbf{x}}_K, \widetilde{\mathbf{y}}_K) \le \mathrm{G}_K(\mathbf{u},\mathbf{v})$. Further, by Inequality 6.2.5, $f(\widetilde{\mathbf{x}}_K) - f(\mathbf{x}^\star) + \frac{1}{2}\|\mathbf{A}(\widetilde{\mathbf{x}}_K - \mathbf{x}^\star)\|^2 \le \mathrm{Gap}_{\mathcal{L}}^{\mathbf{u},\mathbf{v}}(\widetilde{\mathbf{x}}_K, \widetilde{\mathbf{y}}_K)$. Hence, we can conclude from Equation (E.1.21) that

$$\mathbb{E}\Big[f(\widetilde{\mathbf{x}}_K) - f(\mathbf{x}^\star) + \frac{1}{2}\|\mathbf{A}(\widetilde{\mathbf{x}}_K - \mathbf{x}^\star)\|^2\Big] \le \frac{\phi_0(\mathbf{x}^\star)}{A_K} = \frac{\|\mathbf{x}_0 - \mathbf{x}^\star\|_{\mathbf{\Lambda}}^2}{2A_K}. \tag{E.1.22}$$

On the other hand, for $\mathbf{u} = \mathbf{x}^\star$ and $\mathbf{v} = \mathbf{y}^\star = \mathbf{A}\mathbf{x}^\star$, $\mathrm{Gap}_{\mathcal{L}}^{\mathbf{u},\mathbf{v}}(\widetilde{\mathbf{x}}_K, \widetilde{\mathbf{y}}_K) \ge 0$, and, recalling from Equation (6.3.1), Equation (6.3.3), and Equation (6.3.4) that $\mathbf{y}_K = \mathbf{A}\widetilde{\mathbf{x}}_K$, we can also conclude from Equation (E.1.21) that

$$\mathbb{E}\Big[\frac{1}{2}\|\mathbf{A}(\widetilde{\mathbf{x}}_K - \mathbf{x}^\star)\|_2^2\Big] \le \frac{\phi_0(\mathbf{x}^\star)}{A_K} = \frac{\|\mathbf{x}_0 - \mathbf{x}^\star\|_{\mathbf{\Lambda}}^2}{2A_K}. \tag{E.1.23}$$

By Proposition 6.2.1 (b), $f(\mathbf{x}^\star) = -\frac{1}{2}\mathbf{1}^\top\mathbf{x}^\star = -\frac{1}{2}\|\mathbf{A}\mathbf{x}^\star\|_2^2$. Using this identity, one can verify that, $\forall \mathbf{x}$,

$$f(\mathbf{x}) - f(\mathbf{x}^\star) + \frac{1}{2}\|\mathbf{A}(\mathbf{x} - \mathbf{x}^\star)\|_2^2 = \langle \mathbf{A}^T\mathbf{A}\mathbf{x} - \mathbf{1}, \mathbf{x} - \mathbf{x}^\star\rangle = \langle \nabla f(\mathbf{x}), \mathbf{x} - \mathbf{x}^\star\rangle.$$

Hence, summing Equation (E.1.22) and Equation (E.1.23), we also have

$$\mathbb{E}\Big[\langle \nabla f(\widetilde{\mathbf{x}}_K), \widetilde{\mathbf{x}}_K - \mathbf{x}^\star\rangle + \frac{1}{2}\|\mathbf{A}(\widetilde{\mathbf{x}}_K - \mathbf{x}^\star)\|^2\Big] \le \frac{2\phi_0(\mathbf{x}^\star)}{A_K} = \frac{\|\mathbf{x}_0 - \mathbf{x}^\star\|_{\mathbf{\Lambda}}^2}{A_K}.$$

Finally, the bound on the rate of growth of A_k is provided in Section E.1.3. □

The reason $\mathbf{A}$ does not show up in the final bounds (thereby rendering our algorithm "scale-invariant") is because Proposition 6.2.1 allows bounding $\|\mathbf{x}_0 - \mathbf{x}^\star\|_{\mathbf{\Lambda}}^2$ by $|f(\mathbf{x}^\star)|$ where we crucially use the non-negativity of $\mathbf{A}$ and $\mathbf{x}$. This does not seem possible for general $\mathbf{A}$. However, an additive (as opposed to multiplicative) error bound can be obtained even with the more general $\mathbf{A}$ with only small updates to the analysis. This bound would necessarily depend on the scale of $\mathbf{A}$. The choice of the regularizer $\frac{1}{2}\|\cdot - \mathbf{x}_0\|_{\mathbf{\Lambda}}^2$ is also crucial here.

E.1.3 Omitted Proofs from Section 6.3: Growth of Scalar Sequences

In this section, we use the properties of $\{a_i\}$ and $\{a_i^k\}$ to obtain our claimed rate of growth of A_k. Note that in any iteration $k \ge 2$ of Algorithm 6.3.1, there are two possible updates to a_k, which we name as follows.

$$\text{Type I update: } a_{k+1} = \frac{na_k}{n-1} \tag{E.1.24}$$

$$\text{Type II update: } a_{k+1} = \frac{\sqrt{A_k}}{2n} \tag{E.1.25}$$

Obtaining a handle on the growth rate of A_k requires controlling the number of updates of both types specified above. At a high level, the idea behind obtaining such a bound is that if the algorithm had only Type II updates, we would have $A_k \geq \Omega(\frac{k^2}{n^2})$; we then go on to show that we cannot have more than $\frac{5}{2}n \log n$ Type I updates since those make a_k grow too fast. We formalize this intuition in the following lemmas.

Lemma E.1.4. *In Algorithm 6.3.1, we have, for $k \geq 2$, that $a_{k+1} \geq a_k$ and $A_{k+1} > A_k$.*

Proof. Notice that for all k, we have $a_k > 0$, which implies that $A_{k+1} \overset{\text{def}}{=} A_k + a_{k+1}$ satisfies $A_{k+1} > A_k$. To check the non-decreasing nature of a_k, we recall that $a_{k+1} = \min\left(\frac{na_k}{n-1}, \frac{\sqrt{A_k}}{2n}\right)$. In the case that $\frac{\sqrt{A_k}}{2n} \geq \frac{na_k}{n-1}$, we have $a_{k+1} = \frac{na_k}{n-1} > a_k$, as claimed. Consider the other case with $a_{k+1} = \frac{\sqrt{A_k}}{2n}$, and suppose, for the sake of contradiction, that $a_{k+1} < a_k$. Chaining this inequality with the assumed expression for a_{k+1}, scaling appropriately, and squaring both sides gives $A_k < 4n^2 a_k^2$. Plugging this into $A_k = A_{k-1} + a_k$ and solving for a_k from this quadratic inequality (and further invoking the nonnegativity of a_k), yields $a_k > \frac{1+\sqrt{1+16n^2 A_{k-1}}}{8n^2} > \frac{\sqrt{A_{k-1}}}{2n}$. However, this contradicts $a_k = \min\left(\frac{na_{k-1}}{n-1}, \frac{\sqrt{A_{k-1}}}{2n}\right) \leq \frac{\sqrt{A_{k-1}}}{2n}$. □

Lemma E.1.5. *Consider the iterations $\{s_k\}$ in which Algorithm 6.3.1 performs a Type II update $a_{s_k+1} = \frac{\sqrt{A_{s_k}}}{2n}$. Then we have $A_{s_k} \geq \frac{k^2}{c_1 n^2}$ and $a_{s_k} \geq \frac{k-1}{2\sqrt{c_1}n^2}$ for $c_1 = 36$.*

Proof. We prove this claim by induction. First, notice that $s_k \geq k+1$ for any k. Recall our initialization $a_1 = A_1 = \frac{1}{\sqrt{2}n^{1.5}}$. By combining this with the monotonicity property stated in Lemma E.1.4, we have $a_{s_1} \geq a_2 = \frac{1}{\sqrt{2}n^{2.5}} \geq 0$. By using Lemma E.1.4 again, we have, in a similar fashion, that $A_{s_1} \geq A_2 = \frac{1}{\sqrt{2}n^{2.5}} + \frac{1}{\sqrt{2}n^{1.5}} \geq \frac{1}{c_1 n^2}$, which proves the base case for induction. Assume that for some $k > 1$, we have the induction hypothesis $A_{s_k} \geq \frac{k^2}{c_1 n^2}$ and $a_{s_k} \geq \frac{k-1}{2\sqrt{c_1}n^2}$. Then, combining the monotonicity of A_k from Lemma E.1.4 with the fact that the algorithm performs a Type II update on a_{s_k}, we have $a_{s_{k+1}} = \frac{\sqrt{A_{s_{k+1}-1}}}{2n} \geq \frac{\sqrt{A_{s_k}}}{2n} \geq \frac{k}{2\sqrt{c_1}n^2}$. By again applying monotonicity of A_k and the induction hypothesis about a_k, we have $A_{s_{k+1}} = A_{s_{k+1}-1} + a_{s_{k+1}} \geq A_{s_k} + a_{s_{k+1}} \geq \frac{2k^2+\sqrt{c_1}k}{2c_1 n^2} > \frac{(k+1)^2}{c_1 n^2}$. □

Lemma E.1.6. *If at some k_0^{th} iteration of Algorithm 6.3.1, we have that*

$$a_{k_0} > \frac{n-1}{2\sqrt{c_1}n^2}; \quad A_{k_0} \geq \frac{1}{c_1} \tag{E.1.26}$$

then for all $k \geq k_0$, we have that

$$a_k \geq \frac{k-1-k_0+n}{2\sqrt{c_1}n^2}; \quad A_k \geq \frac{(k-k_0+n)^2}{c_1 n^2} \tag{E.1.27}$$

for $c_1 = 36$.

Proof. We prove the claim by induction. First, the base case is true for $k = k_0$ by our assumption on a_{k_0} and A_{k_0}. Assume the induction hypothesis $a_k \geq \frac{k-1-k_0+n}{2\sqrt{c_1}n^2}$ and $A_k \geq \frac{(k-k_0+n)^2}{c_1 n^2}$ for $k \geq k_0$. We now discuss how a_k changes with the two types of updates.

If the algorithm performs a Type I update on a_k, then, by definition, $a_{k+1} = \frac{na_k}{n-1}$. Now applying the assumed lower bound on a_k, we have, when $k > k_0$, that

$$a_{k+1} = \frac{na_k}{n-1} \geq \frac{k-1-k_0+n}{2\sqrt{c_1}n(n-1)} \geq \frac{k-k_0+n}{2\sqrt{c_1}n^2}.$$

Similarly, given that $A_k \geq \frac{(k-k_0+n)^2}{c_1 n^2}$, we have,

$$A_{k+1} = A_k + a_{k+1} \geq \frac{(k-k_0+n)^2}{c_1 n^2} + \frac{k-k_0+n}{2\sqrt{c_1}n^2} \geq \frac{(k+1-k_0+n)^2}{c_1 n^2}.$$

If, on the other hand, the algorithm performs a Type II update on a_k, then we have

$$a_{k+1} = \frac{\sqrt{A_k}}{2n} \geq \frac{k-k_0+n}{2\sqrt{c_1}n^2}.$$

This completes the induction. □

As we saw in Lemma E.1.6, after the k_0^{th} iteration - starting at which Inequality E.1.26 holds - A_k grows fast. We therefore need to estimate the number of Type I updates *before* the k_0^{th} iteration.

Lemma E.1.7. *There are at most $\frac{3}{2}n\log n$ Type I updates (Equation (E.1.24)) performed before the k_0^{th} iteration (the first iteration at which Inequality E.1.26 holds).*

Proof. Suppose there are n_1 Type I updates performed by Algorithm 6.3.1 before the k_0^{th} iteration, when Inequality E.1.26 starts to hold. Further, by Lemma E.1.4, a_k is monotonically increasing (for both types of updates). Then, when considering Type I updates (Equation (E.1.24)), we have $a_{k_0} \geq \left(\frac{n}{n-1}\right)^{n_1} a_2 = \left(\frac{n}{n-1}\right)^{n_1} \cdot \frac{1}{\sqrt{2}n^{2.5}}$. In order for $a_{k_0} > \frac{n-1}{12n^2}$, we only need to have $n_1 > \log_{\frac{n}{n-1}}\left(\frac{\sqrt{n}(n-1)}{6\sqrt{2}}\right)$. In a similar fashion, combining the monotonicity of A_k from Lemma E.1.4 with the Type I update rule, we have

$$A_{k_0} \geq a_2\left(1 + \frac{n}{n-1} + \left(\frac{n}{n-1}\right)^2 + \cdots + \left(\frac{n}{n-1}\right)^{n_1}\right) > \frac{(\frac{n}{n-1})^{n_1}}{\sqrt{2}n^{2.5}}.$$

So, in order to have $A_{k_0} > \frac{1}{36}$ per Inequality E.1.26, we only need to have $n_1 \geq \log_{\frac{n}{n-1}}(\frac{n^{2.5}}{18\sqrt{2}})$. By using the approximation $1 + x \leq e^x$ and combining the above two bounds, we get as soon as $n_1 \geq \frac{3}{2}n\log n$, the inequality (E.1.26) holds. □

Proposition E.1.8. *[Rate of change of A_k] When $k \geq \frac{5}{2}n\log n$, we have $A_k \geq \frac{(k-\frac{5}{2}n\log n)^2}{36n^2}$.*

Proof. Let there be t_1 Type I updates and t_2 Type II updates before the first iteration at which Inequality E.1.26 holds, and let us call this iteration k_0. By the result of Lemma E.1.7, we have $t_1 \leq \frac{3}{2}n\log n$. By the result of Lemma E.1.5, we must have $A_{k_0} \geq \frac{t_2^2}{c_1 n^2}$ and $a_{k_0} \geq \frac{t_2-1}{2\sqrt{c_1}n^2}$. To meet the requirement in Inequality E.1.26 then, we can see that $t_2 \leq n$. Therefore, $k_0 = t_1 + t_2 \leq \frac{3}{2}n\log n + n \leq \frac{5}{2}n\log n$. Having reached the k_0^{th} iteration, the result of Lemma E.1.6 applies, and we have $A_k \geq \frac{(k-k_0)^2}{c_1 n^2}$. □

E.1.4 Omitted Proofs from Section 6.4: Restart Strategy

To establish local error bounds, we start with the observation that (P) is equivalent to a linear complementarity problem.

Proposition E.1.9. *Problem* (P) *is equivalent to the following linear complementarity problem, denoted by* LCP($\mathbf{M}, \mathbf{q}$).

$$\mathbf{M}\mathbf{x} + \mathbf{q} \geq \mathbf{0},\ \mathbf{x} \geq \mathbf{0},\ \langle \mathbf{x}, \mathbf{M}\mathbf{x} + \mathbf{q} \rangle = 0, \tag{E.1.28}$$

where $\mathbf{\Lambda}^{-1}\mathbf{M} = \mathbf{A}^\top \mathbf{A}$ *and* $\mathbf{q} = -\mathbf{\Lambda}^{-1}\mathbf{1}$.

Proof. Observe first that, as $\mathbf{\Lambda}^{-1}$ is a diagonal matrix with positive elements on the diagonal, the stated linear complementarity problem is equivalent to

$$\nabla f(\mathbf{x}) \geq \mathbf{0},\ \mathbf{x} \geq \mathbf{0},\ \langle \nabla f(\mathbf{x}), \mathbf{x} \rangle = 0. \tag{E.1.29}$$

By Proposition 6.2.1, these conditions hold for any solution of (P). In the opposite direction, suppose that the conditions from Equation (E.1.29) hold for some $\mathbf{x}$. Then applying these conditions for any $\mathbf{u} \geq \mathbf{0}$ gives

$$\langle \nabla f(\mathbf{x}), \mathbf{u} - \mathbf{x} \rangle = \langle \nabla f(\mathbf{x}), \mathbf{u} \rangle \geq 0.$$

But $\langle \nabla f(\mathbf{x}), \mathbf{u} - \mathbf{x} \rangle \geq 0$ is the first-order optimality condition for (P), and so $\mathbf{x}$ solves (P). □

For $r(\mathbf{x}) = \|\mathbf{R}(\mathbf{x})\|_{\mathbf{\Lambda}}$, a quantity termed *natural residual* [MR94], local error bound is obtained as a corollary of the following theorem.

Theorem E.1.10 ([MR94], Theorem 2.1)**.** *Let* $\mathbf{M} \in \mathbb{R}^{n \times n}$ *be such that* LCP($\mathbf{M}, \mathbf{0}$) *has* $\mathbf{0}$ *as its unique solution. Then there exists* $\mu > 0$ *such that for each* $\mathbf{x} \in \mathbb{R}^n$*, we have* $r(\mathbf{x}) \geq \mu \|\mathbf{x} - \mathbf{x}^\star\|$*, where* $\mathbf{x}^\star$ *is a solution to* LCP($\mathbf{M}, \mathbf{q}$) *that is closest to* $\mathbf{x}$ *under the norm* $\|\cdot\|$.

Theorem E.1.10 applies to our problem due to the nonnegativity (and nondegeneracy) of $\mathbf{A}$ and choosing $\|\cdot\| = \|\cdot\|_{\mathbf{\Lambda}}$. By arguing that Theorem 6.3.5 provides an upper bound on $r(\widetilde{\mathbf{x}}_K)$, in expectation, we then obtain our final result below.

Proposition E.1.11. *For any* $\mathbf{x} \in \mathbb{R}^n_+$, $r(\mathbf{x}) \leq \sqrt{2n(f(\mathbf{x}) - f(\mathbf{x}^\star))}$, *where* $\mathbf{x}^\star \in \operatorname{argmin}_{\mathbf{u} \in \mathbb{R}^n_+} f(\mathbf{u})$.

Proof. Given $\mathbf{x} \in \mathbb{R}^n_+$, consider $\hat{\mathbf{x}}$ defined as $\hat{x}_{j^\star} = x_{j^\star} - \mathbf{R}_{j^\star}(\mathbf{x})$, where $j^\star = \operatorname{argmax}_{1 \leq j \leq n} |\mathbf{R}_j(\mathbf{x})| \cdot \|\mathbf{A}_{:j}\|_2$, and $\hat{x}_j = x_j$ for $j \neq j^\star$. Then observing that

$$\begin{aligned} f(\hat{\mathbf{x}}) - f(\mathbf{x}) &= \nabla_{j^\star} f(\mathbf{x})([\hat{\mathbf{x}}]_{j^\star} - [\mathbf{x}]_{j^\star}) + \frac{\|\mathbf{A}_{:j^\star}\|_2^2}{2} |[\hat{\mathbf{x}}]_{j^\star} - [\mathbf{x}]_{j^\star}|^2 \\ &\leq -\frac{1}{2} |\mathbf{R}_{j^\star}(\mathbf{x})|^2 \|\mathbf{A}_{:j^\star}\|_2^2 \leq -\frac{1}{2n} \|\mathbf{R}(\mathbf{x})\|_{\mathbf{\Lambda}}^2, \end{aligned}$$

and combining with $f(\hat{\mathbf{x}}) \geq f(\mathbf{x}^\star)$, $r(\mathbf{x}) = \|\mathbf{R}(\mathbf{x})\|_{\mathbf{\Lambda}}$, the claimed bound follows after a simple rearrangement. □

Theorem 6.4.1. *Given an error parameter* $\varepsilon > 0$ *and* $\mathbf{x}_0 = \mathbf{0}$*, consider the following algorithm* $\mathcal{A}$:

$\mathcal{A}$: ***SI-NNLS+ with Restarts***
Initialize: $k = 1$.
Initialize Lazy SI-NNLS+ at $\mathbf{x}_{k-1}$.
Run Lazy SI-NNLS+ until the output $\widetilde{\mathbf{x}}_K^k$ *satisfies* $r(\widetilde{\mathbf{x}}_K^k) \leq \frac{1}{2} r(\mathbf{x}_{k-1})$.
Restart Lazy SI-NNLS+ initializing at $\mathbf{x}_k = \widetilde{\mathbf{x}}_K^k$.
Increment k.
Repeat until $r(\widetilde{\mathbf{x}}_K^k) \leq \epsilon$.

Then, the expected number of arithmetic operations of $\mathcal{A}$ *is* $O\Big(\mathrm{nnz}(\mathbf{A})\Big(\log n + \frac{\sqrt{n}}{\mu}\Big)\log\Big(\frac{r(\mathbf{x}_0)}{\epsilon}\Big)\Big)$. *As a consequence, given* $\bar{\epsilon} > 0$, *the total expected number of arithmetic operations until a point with* $f(\mathbf{x}) - f(\mathbf{x}^\star) \leq \bar{\epsilon}|f(\mathbf{x}^\star)|$ *can be constructed by* $\mathcal{A}$ *is* $O\Big(\mathrm{nnz}(\mathbf{A})\Big(\log n + \frac{\sqrt{n}}{\mu}\Big)\log\Big(\frac{n}{\mu\bar{\epsilon}}\Big)\Big)$.

Proof. Because each restart halves the natural residual $r(\mathbf{x})$, it is immediate that the total number of restarts until $r(\widetilde{\mathbf{x}}_K^k) \leq \epsilon$ is bounded by $\log\Big(\frac{r(\mathbf{x}_0)}{\epsilon}\Big)$. Thus, to prove the first (and main) part of the theorem, we only need to bound the number of iterations (and the overall number of arithmetic operations) of (Lazy) SI-NNLS+ in expectation. Hence, in the following, we only consider one run of SI-NNLS+ until the natural residual is halved. To keep the notation simple, we let $\mathbf{x}_0$ denote the initial point of SI-NNLS+ and $\widetilde{\mathbf{x}}_k$ denote the output of SI-NNLS+ at iteration k. If $r(\mathbf{x}_0) = 0$, $\mathcal{A}$ halts immediately and the bound on the number of iterations holds trivially, so assume $r(\mathbf{x}_0) > 0$. Using Theorem 6.3.5, we have that $\forall k \geq 2$,

$$\mathbb{E}[A_k r^2(\widetilde{\mathbf{x}}_k)] \leq n\|\mathbf{x}_0 - \mathbf{x}^\star\|_{\mathbf{\Lambda}}^2 \leq \frac{n}{\mu^2} r^2(\mathbf{x}_0). \tag{E.1.30}$$

As $r^2(\cdot)$ is nonnegative, we can use Markov's inequality to bound the total number of iterations K until $r(\widetilde{\mathbf{x}}_K) \leq \frac{r(\mathbf{x}_0)}{2}$. In particular, using Equation (E.1.30), we get by Markov's inequality that $\Pr[K > k] \leq \Pr\Big[r^2(\widetilde{\mathbf{x}}_k) \geq \frac{r^2(\mathbf{x}_0)}{4}\Big] \leq \frac{4n}{\mu^2 A_k}$. As K is nonnegative, we can estimate its expectation using

$$\begin{aligned}
\mathbb{E}[K] &= \sum_{i=0}^{\infty} \Pr[K > i] \leq \sum_{i=0}^{\infty} \min\Big\{1, \frac{4n}{\mu^2 A_i}\Big\} \\
&\leq \sum_{i=0}^{\lceil 12n\sqrt{n}/\mu + \frac{5}{2}n\log n\rceil} 1 + \sum_{\lceil 12n\sqrt{n}/\mu + \frac{5}{2}n\log n\rceil + 1}^{\infty} \frac{4n}{\mu^2 A_i} \\
&\leq 24n\sqrt{n}/\mu + \frac{5}{2}n\log n + 2,
\end{aligned}$$

where in the last inequality we use the rate of A_k from Proposition E.1.8.

In the lazy implementation of SI-NNLS+, as argued in Section E.2, the expected cost of an iteration is $\frac{\mathrm{nnz}(\mathbf{A})}{n}$, which leads to the claimed bound on the number of arithmetic operations until $r(\mathbf{x}) \leq \epsilon$.

By using that $r(\mathbf{x}_0) \leq \sqrt{2n(f(\mathbf{x}_0) - f(\mathbf{x}^\star))} = \sqrt{2n|f(\mathbf{x}^\star)|}$, $f\Big(\widetilde{\mathbf{x}}^{K_1} - \mathbf{R}(\widetilde{\mathbf{x}}^{K_1})\Big) - f(\mathbf{x}^\star) \leq \Big((n-1) + \frac{n+1}{\mu}\Big) r^2(\widetilde{\mathbf{x}}^{K_1})$

(argued below), the bound on the number of iterations until $f\left(\widetilde{\mathbf{x}}^{K_1} - \mathbf{R}(\widetilde{\mathbf{x}}^{K_1})\right) - f(\mathbf{x}^\star) \le \bar{\epsilon}|f(\mathbf{x}^\star)|$ have

$$\begin{aligned} f\left(\widetilde{\mathbf{x}}^{K_1} - \mathbf{R}(\widetilde{\mathbf{x}}^{K_1})\right) - f(\mathbf{x}^\star) &\le \left((n-1) + \frac{n+1}{\mu}\right) r^2(\widetilde{\mathbf{x}}^{K_1}) \\ &\le \left((n-1) + \frac{n+1}{\mu}\right) \frac{1}{2^{2K_1}} r^2(\mathbf{x}_0) \\ &\le \left((n-1) + \frac{n+1}{\mu}\right) \frac{1}{2^{2K_1}} 2n|f(\mathbf{x}^\star)| \end{aligned}$$

and by setting $K_1 = \frac{1}{2}\log_2 \frac{2n\left((n-1)+\frac{n+1}{\mu}\right)}{\bar{\epsilon}}$, we have this bound.

Finally, it remains to argue that $f\left(\widetilde{\mathbf{x}}^{K_1} - \mathbf{R}(\widetilde{\mathbf{x}}^{K_1})\right) - f(\mathbf{x}^\star) \le \left((n-1) + \frac{n+1}{\mu}\right) r^2(\widetilde{\mathbf{x}}^{K_1})$. Observe that the definition of $\mathbf{R}(\mathbf{x})$ is equivalent to $\mathbf{x} - \bar{\mathbf{x}}$, where

$$\bar{\mathbf{x}} = \operatorname{argmin}_{\mathbf{u}\in\mathbb{R}^n_+} \left\{ \langle \nabla f(\mathbf{x}), \mathbf{u} - \mathbf{x}\rangle + \frac{1}{2}\|\mathbf{u} - \mathbf{x}\|^2_{\mathbf{\Lambda}} \right\}.$$

By the first-order optimality of $\bar{\mathbf{x}}$ based on the equivalent definition of $\mathbf{R}(\mathbf{x})$ above, we have $\langle \nabla f(\mathbf{x}) + \mathbf{\Lambda}(\bar{\mathbf{x}} - \mathbf{x}), \mathbf{x}^\star - \bar{\mathbf{x}}\rangle \ge 0$. Rearranging, and using the definition of convexity of f, we have

$$\begin{aligned} f(\bar{\mathbf{x}}) - f(\mathbf{x}^\star) &\le \langle \nabla f(\bar{\mathbf{x}}), \bar{\mathbf{x}} - \mathbf{x}^\star\rangle \\ &\le \langle \nabla f(\mathbf{x}) - \nabla f(\bar{\mathbf{x}}) + \mathbf{\Lambda}(\bar{\mathbf{x}} - \mathbf{x}), \mathbf{x}^\star - \bar{\mathbf{x}}\rangle \\ &= \langle (\mathbf{A}^\top\mathbf{A} - \mathbf{\Lambda})(\mathbf{x} - \bar{\mathbf{x}}), \mathbf{x}^\star - \bar{\mathbf{x}}\rangle \\ &= \langle (\mathbf{A}^\top\mathbf{A} - \mathbf{\Lambda})\mathbf{R}(\mathbf{x}), \mathbf{R}(\mathbf{x})\rangle + \langle (\mathbf{A}^\top\mathbf{A} - \mathbf{\Lambda})\mathbf{R}(\mathbf{x}), \mathbf{x}^\star - \mathbf{x}\rangle \\ &= \langle (\mathbf{A}^\top\mathbf{A} - \mathbf{\Lambda})\mathbf{R}(\mathbf{x}), \mathbf{R}(\mathbf{x})\rangle + \langle \mathbf{A}^\top\mathbf{A}\mathbf{R}(\mathbf{x}), \mathbf{x}^\star - \mathbf{x}\rangle - \langle \mathbf{\Lambda}\mathbf{R}(\mathbf{x}), \mathbf{x}^\star - \mathbf{x}\rangle \\ &\le (n-1)\|\mathbf{R}(\mathbf{x})\|^2_{\mathbf{\Lambda}} + (n+1)\|\mathbf{R}(\mathbf{x})\|_{\mathbf{\Lambda}}\|\mathbf{x} - \mathbf{x}^\star\|_{\mathbf{\Lambda}} \\ &\le \left((n-1) + \frac{n+1}{\mu}\right) r^2(\mathbf{x}), \end{aligned}$$

where in the last inequality we have used the error bound from Theorem E.1.10. □

E.2 Implementation Version of SI-NNLS+

Since Algorithm 6.3.1 explicitly updates $\widetilde{\mathbf{x}}_k$ and $\widetilde{\mathbf{y}}_k$ (of lengths n and m respectively), the per iteration cost is $O(m+n)$, which is unnecessarily high when the matrix $\mathbf{A}$ is sparse. In this section, we show that by using a *lazy* update strategy, we can efficiently implement Algorithm 6.3.1 with overall complexity independent of the ambient dimension. To attain this result, we maintain implicit representations for $\widetilde{\mathbf{x}}_k$, $\mathbf{y}_k$, and $\bar{\mathbf{y}}_k$ by introducing two auxiliary variables that are amenable to efficient updates.

Efficiently Updating the Primal Variable. In Lemma E.2.1, we show that we can work with an implicit representation of $\widetilde{\mathbf{x}}_k$ by introducing $\mathbf{r}_k$.

Lemma E.2.1. *For $\{\widetilde{\mathbf{x}}_k\}$ defined in Equation (6.3.3) (and simplified in Algorithm 6.3.1), we have, for $k \ge 1$,*

$$\widetilde{\mathbf{x}}_k = \mathbf{x}_k + \frac{1}{A_k}\mathbf{r}_k, \tag{E.2.1}$$

where $\mathbf{x}_k$ evolves as per Algorithm 6.3.1, $\mathbf{r}_1 = \mathbf{0}$ and, when $k \ge 1$, $\mathbf{r}_k = \mathbf{r}_{k-1} + ((n-1)a_k - A_{k-1})(\mathbf{x}_k - \mathbf{x}_{k-1})$.

Proof. We prove the lemma by induction. Using the facts that $\mathbf{x}_0 = \mathbf{0}$, $\mathbf{x}_1 = \widetilde{\mathbf{x}}_1$, $\mathbf{s}_1 = \mathbf{0}$, $a_1 = A_1$, and $A_0 = 0$, we have

$$\begin{aligned}\widetilde{\mathbf{x}}_1 &= \frac{1}{A_1}\left(A_0\widetilde{\mathbf{x}}_0 + a_1\Big(n\mathbf{x}_1 - (n-1)\mathbf{x}_0\Big)\right)\\ &= \frac{1}{A_1}\left(a_1\mathbf{x}_1 + (n-1)a_1(\mathbf{x}_1 - \mathbf{x}_0)\right)\\ &= \mathbf{x}_1 + ((n-1)a_1 - A_0)(\mathbf{x}_1 - \mathbf{x}_0). \end{aligned} \tag{E.2.2}$$

Assume for certain $k \geq 2$, that Eq. (E.2.1) holds for $k-1$. Then, using the recursion of $\tilde{\mathbf{x}}_k$ in Algorithm 6.3.1, we have that for $k \geq 3$,

$$\begin{aligned}A_k\widetilde{\mathbf{x}}_k &= A_{k-1}\widetilde{\mathbf{x}}_{k-1} + a_k\mathbf{x}_k + (n-1)a_k(\mathbf{x}_k - \mathbf{x}_{k-1})\\ &= A_{k-1}\mathbf{x}_{k-1} + \mathbf{r}_{k-1} + a_k\mathbf{x}_k + (n-1)a_k(\mathbf{x}_k - \mathbf{x}_{k-1})\\ &= A_{k-1}(\mathbf{x}_{k-1} - \mathbf{x}_k + \mathbf{x}_k) + \mathbf{r}_{k-1} + a_k\mathbf{x}_k + (n-1)a_k(\mathbf{x}_k - \mathbf{x}_{k-1})\\ &= A_{k-1}(\mathbf{x}_{k-1} - \mathbf{x}_k) + A_{k-1}\mathbf{x}_k + \mathbf{r}_{k-1} + a_k\mathbf{x}_k + (n-1)a_k(\mathbf{x}_k - \mathbf{x}_{k-1})\\ &= A_k\mathbf{x}_k + \mathbf{r}_{k-1} + ((n-1)a_k - A_{k-1})(\mathbf{x}_k - \mathbf{x}_{k-1})\\ &= A_k\mathbf{x}_k + \mathbf{r}_k,\end{aligned}$$

as required. □

The expression for $\mathbf{r}_k$ in Lemma E.2.1 shows that it can be updated at cost $O(1)$ as $\mathbf{x}_k$ differs from $\mathbf{x}_{k-1}$ only at one coordinate. Therefore, by Equation (E.2.1) we need not compute $\tilde{\mathbf{x}}_k$ in all iterations and can instead maintain $\mathbf{r}_k$. Along the same lines, we give an efficient implementation strategy for $\mathbf{y}_k$ and $\bar{\mathbf{y}}_k$ in the following discussion.

Efficiently Updating the Dual Variable. We now show how to update the dual variable efficiently.

Lemma E.2.2. *Consider* $\{\mathbf{y}_k\}$ *and* $\{\mathbf{x}_k\}$ *evolving as per Algorithm 6.3.1. Then, for* $k = 1$, *we have* $\mathbf{y}_1 = \mathbf{A}\mathbf{x}_1$; *for* $k \geq 2$, *we have*

$$\mathbf{y}_k = \frac{A_{k-1}}{A_k}\mathbf{y}_{k-1} + \frac{a_k}{A_k}\mathbf{A}\mathbf{x}_k + \frac{(n-1)a_k}{A_k}\mathbf{A}(\mathbf{x}_k - \mathbf{x}_{k-1}), \tag{E.2.3}$$

Proof. The proof is directly from the definition of $\mathbf{y}_k$ in Algorithm 6.3.1. □

Lemma E.2.3. *Consider* $\{\mathbf{y}_k\}$ *and* $\{\mathbf{x}_k\}$ *evolving as per Algorithm 6.3.1. Then for all* $k \geq 1$, *we have*

$$\mathbf{y}_k = \mathbf{A}\mathbf{x}_k + \frac{1}{A_k}\mathbf{s}_k, \tag{E.2.4}$$

where $\mathbf{s}_1 = 0$ *and* $\mathbf{s}_k = \mathbf{s}_{k-1} + ((n-1)a_k - A_{k-1})\mathbf{A}(\mathbf{x}_k - \mathbf{x}_{k-1})$ *when* $k \geq 2$.

Proof. We prove the lemma by induction. For the base case of $k = 1$, we have, by the choice of $\mathbf{s}_1 = \mathbf{0}$,

that $\mathbf{y}_1 = \mathbf{A}\mathbf{x}_1 = \mathbf{A}\mathbf{x}_1 + \frac{1}{A_1}\mathbf{s}_1$. Then for some $k \geq 2$, assume Eq. (E.2.4) holds for $k-1$, then we have,

$$\begin{aligned} A_k\mathbf{y}_k &= A_{k-1}\mathbf{y}_{k-1} + a_k\mathbf{A}\mathbf{x}_k + (n-1)a_k\mathbf{A}(\mathbf{x}_k - \mathbf{x}_{k-1}) \\ &= A_{k-1}\mathbf{A}\mathbf{x}_{k-1} + \mathbf{s}_{k-1} + a_k\mathbf{A}\mathbf{x}_k + (n-1)a_k\mathbf{A}(\mathbf{x}_k - \mathbf{x}_{k-1}) \\ &= A_{k-1}\mathbf{A}(\mathbf{x}_{k-1} - \mathbf{x}_k + \mathbf{x}_k) + \mathbf{s}_{k-1} + a_k\mathbf{A}\mathbf{x}_k + (n-1)a_k\mathbf{A}(\mathbf{x}_k - \mathbf{x}_{k-1}) \\ &= A_k\mathbf{A}\mathbf{x}_k + \mathbf{s}_{k-1} + ((n-1)a_k - A_{k-1})\mathbf{A}(\mathbf{x}_k - \mathbf{x}_{k-1}) \\ &= A_k\mathbf{A}\mathbf{x}_k + \mathbf{s}_k, \end{aligned} \tag{E.2.5}$$

where the first step is by Lemma E.2.2, second by the induction hypothesis, third by adding and subtracting $A_{k-1}\mathbf{A}\mathbf{x}_k$, fourth by rearranging terms appropriately, and the final step uses the recursive definition of $\mathbf{s}_k$ stated in the lemma. Dividing throughout by A_k then finishes the proof. □

Algorithm E.2.1 SI-NNLS+ (Implementation)

Input: Matrix $\mathbf{A} \in \mathbb{R}_+^{m\times n}$ with $n \geq 4$, accuracy ϵ
Output: Vector $\widetilde{\mathbf{x}}_K \in \mathbb{R}_+^n$ such that $f(\widetilde{\mathbf{x}}_K) \leq (1-\epsilon)f(\mathbf{x}^\star)$.
Initialize: $a_1 = \frac{1}{n-1}$, $a_2 = \frac{n}{n-1}$, $A_1 = a_1$, $\phi_0(\mathbf{x}) = \frac{1}{2}\|\mathbf{x} - \mathbf{x}_0\|_{\mathbf{\Lambda}}^2$, $\bar{\mathbf{y}}_0 = \mathbf{y}_0 = \mathbf{A}\mathbf{x}_0$, $\mathbf{p}_0 = \mathbf{0}$, $\mathbf{q}_0 = \mathbf{A}\mathbf{x}_0$, $\mathbf{t}_0 = \mathbf{0}, \mathbf{s}_1 = \mathbf{0}, \mathbf{r}_1 = \mathbf{0}$.
for $k = 1, 2, \ldots, K$ **do**
  Sample j_k uniformly at random from $\{1, 2, \ldots, n\}$
  if $k = 1$ **then**
    $\bar{\mathbf{y}}_0 = \mathbf{q}_0$
  else if $k = 2$ **then**
    $\bar{\mathbf{y}}_1 = \mathbf{q}_1 + \frac{a_1}{a_2}\mathbf{t}_1$
  else if $k \geq 3$ **then**
    $\bar{\mathbf{y}}_{k-1} = \mathbf{q}_{k-1} + \frac{1}{A_{k-1}}\left(1 - \frac{a_{k-1}^2}{a_k A_{k-2}}\right)\mathbf{s}_{k-1} + \frac{(n-1)a_{k-1}^2}{a_k A_{k-2}}\mathbf{t}_{k-1}$
  end if
  $p_{k,i} = \begin{cases} p_{k-1,i}, & i \neq j_k \\ p_{k-1,i} + na_k\left(\mathbf{A}_{:i}^T\bar{\mathbf{y}}_{k-1} - 1\right), & i = j_k. \end{cases}$
  $x_{k,i} = \begin{cases} x_{k-1,i}, & i \neq j_k \\ \max\left\{0, \min\left\{x_{0,i} - \frac{1}{\|\mathbf{A}_{:i}\|^2}\cdot p_{k,i}, \frac{1}{\|\mathbf{A}_{:i}\|^2}\right\}\right\}, & i = j_k \end{cases}$
  $\mathbf{t}_k = \mathbf{A}(\mathbf{x}_k - \mathbf{x}_{k-1})$
  if $k \geq 2$ **then**
    $\mathbf{r}_k = \mathbf{r}_{k-1} + ((n-1)a_k - A_{k-1})(\mathbf{x}_k - \mathbf{x}_{k-1})$
    $\mathbf{s}_k = \mathbf{s}_{k-1} + ((n-1)a_k - A_{k-1})\mathbf{t}_k$
  end if
  $\mathbf{q}_k = \mathbf{q}_{k-1} + \mathbf{t}_k$
  $A_{k+1} = A_k + a_{k+1}$
  $a_{k+2} = \min\{\frac{na_{k+1}}{n-1}, \frac{\sqrt{A_{k+1}}}{2n}\}$
end for
return $\mathbf{x}_K + \frac{1}{A_K}\mathbf{r}_K$

Lemma E.2.4. *Consider* $\{\mathbf{x}_k\}$, $\{\mathbf{y}_k\}$, *and* $\{\bar{\mathbf{y}}_k\}$ *evolving as per Algorithm 6.3.1. Then we have that* $\bar{\mathbf{y}}_1 =$

$\mathbf{A}\mathbf{x}_1 + \frac{a_1}{a_2}\mathbf{A}(\mathbf{x}_1 - \mathbf{x}_0)$ *and*

$$\bar{\mathbf{y}}_k = \mathbf{A}\mathbf{x}_k + \frac{1}{A_k}\Big(1 - \frac{a_k^2}{a_{k+1}A_{k-1}}\Big)\mathbf{s}_k + \frac{(n-1)a_k^2}{a_{k+1}A_{k-1}}\mathbf{A}(\mathbf{x}_k - \mathbf{x}_{k-1}). \tag{E.2.6}$$

Proof. From the definition of $\bar{\mathbf{y}}_k$, the initializations for $\mathbf{x}_0, \mathbf{y}_0$, and $\overline{\mathbf{y}}_0$, and Lemma E.2.2, we have $\bar{\mathbf{y}}_1 = \mathbf{y}_1 + \frac{a_1}{a_2}(\mathbf{y}_1 - \mathbf{y}_0) = \mathbf{A}\mathbf{x}_1 + \frac{a_1}{a_2}\mathbf{A}(\mathbf{x}_1 - \mathbf{x}_0)$. For $k \geq 2$, by Lemma E.2.2, we have $A_k\mathbf{y}_k - A_{k-1}\mathbf{y}_{k-1} = a_k\mathbf{A}\mathbf{x}_k + (n-1)a_k\mathbf{A}(\mathbf{x}_k - \mathbf{x}_{k-1})$. As a result,

$$A_{k-1}(\mathbf{y}_k - \mathbf{y}_{k-1}) = a_k\mathbf{A}\mathbf{x}_k + (n-1)a_k\mathbf{A}(\mathbf{x}_k - \mathbf{x}_{k-1}) - a_k\mathbf{y}_k. \tag{E.2.7}$$

So for $k \geq 2$, it follows that

$$\begin{aligned}
\bar{\mathbf{y}}_k &= \mathbf{y}_k + \frac{a_k}{a_{k+1}}(\mathbf{y}_k - \mathbf{y}_{k-1}) = \mathbf{y}_k + \frac{a_k}{a_{k+1}}\Big(\frac{a_k}{A_{k-1}}\mathbf{A}\mathbf{x}_k + \frac{(n-1)a_k}{A_{k-1}}\mathbf{A}(\mathbf{x}_k - \mathbf{x}_{k-1}) - \frac{a_k}{A_{k-1}}\mathbf{y}_k\Big) \\
&= \Big(1 - \frac{a_k^2}{a_{k+1}A_{k-1}}\Big)\mathbf{y}_k + \frac{a_k^2}{a_{k+1}A_{k-1}}\mathbf{A}\mathbf{x}_k + \frac{(n-1)a_k^2}{a_{k+1}A_{k-1}}\mathbf{A}(\mathbf{x}_k - \mathbf{x}_{k-1}) \\
&= \mathbf{A}\mathbf{x}_k + \frac{1}{A_k}\Big(1 - \frac{a_k^2}{a_{k+1}A_{k-1}}\Big)\mathbf{s}_k + \frac{(n-1)a_k^2}{a_{k+1}A_{k-1}}\mathbf{A}(\mathbf{x}_k - \mathbf{x}_{k-1}),
\end{aligned}$$

where the first step is by the definition of $\overline{\mathbf{y}}_k$ in Algorithm 6.3.1, the second step is by Equation (E.2.7), the third step is by rearranging, and the final step is by Lemma E.2.3. □

Based on the above lemmas, we give our efficient lazy implementation version of Algorithm 6.3.1 in Algorithm E.2.1. In Algorithm E.2.1, we also introduce other auxiliary variables $\mathbf{p}_k, \mathbf{q}_k$ and $\mathbf{t}_k$. Based on Lemma E.2.1-Lemma E.2.4, it is easy to verify the equivalence between Algorithm 6.3.1 and Algorithm E.2.1. With this implementation, by updating only the dual coordinates corresponding to the nonzero coordinates of the selected column of $\mathbf{A}$, the per-iteration cost is proportional to the number of nonzero elements of the selected row in the iteration. As a result, the overall complexity result will depend only on the number of nonzero elements of $\mathbf{A}$.

Appendix F

Appendix for Chapter 7

This chapter contains details and proofs from Chapter 7.

We start with a piece of notation we frequently use in the appendix. For a given vector $x \in \mathbb{R}^m$, we use $\mathbf{Diag}(x)$ to describe the diagonal matrix with x on its diagonal. For a matrix $\mathbf{X}$, we use $\mathbf{diag}(\mathbf{X})$ to denote the vector made up of the diagonal entries of $\mathbf{X}$. Further, recall as stated in Section 7.1.4, that given any vector x, we use its uppercase boldface name $\mathbf{X} \stackrel{\text{def}}{=} \mathbf{Diag}(x)$.

F.1 Technical Proofs: Gradient, Hessian, Initial Error, Minimum Progress

Lemma 7.2.3 (Gradient and Hessian)**.** *For any $w \in \mathbb{R}^m_{>0}$, the objective in (7.1.4), $\mathcal{F}(w) = -\log\det(\mathbf{A}^\top\mathbf{W}\mathbf{A}) + \frac{1}{1+\alpha}\mathbf{1}^\top w^{1+\alpha}$, has gradient and Hessian given by the following expressions.*

$$[\nabla\mathcal{F}(w)]_i = w_i^{-1}\cdot(w_i^{1+\alpha} - \sigma_i(w)) \text{ and } \nabla^2\mathcal{F}(w) = \mathbf{W}^{-1}\mathbf{P}(w)^{(2)}\mathbf{W}^{-1} + \alpha\mathbf{W}^{\alpha-1}.$$

Proof. The proof essentially follows by combining Lemmas 48 and 49 of [LS14]. For completeness, we provide the full proof here. Applying chain rule to the log det function and then the definition of $\rho(w)$ from (7.2.1) gives the claim that

$$\nabla_i\mathcal{F}(w) = -(\mathbf{A}(\mathbf{A}^\top\mathbf{W}\mathbf{A})^{-1}\mathbf{A}^\top)_{ii} + w_i^\alpha = -a_i^\top(\mathbf{A}^\top\mathbf{W}\mathbf{A})^{-1}a_i + w_i^\alpha = \frac{-\sigma_i(w)}{w_i} + w_i^\alpha.$$

We now set some notation to compute the Hessian: let $\mathbf{M} \stackrel{\text{def}}{=} \mathbf{A}(\mathbf{A}^\top\mathbf{W}\mathbf{A})^{-1}\mathbf{A}^\top$, let $h \in \mathbb{R}^m$ be any arbitrary vector, and let $\mathbf{H} \stackrel{\text{def}}{=} \mathbf{Diag}(h)$. For $f : \mathbb{R}^n \to \mathbb{R}$ and for $x, h \in \mathbb{R}^n$ we let $\mathcal{D}_x f(x)[h]$ denote the directional derivative of f at x in the direction h, i.e., $\mathcal{D}_x f(x)[h] = \lim_{t\to 0}(f(x+th) - f(x))/t$. Then we have,

$$\begin{aligned}
\mathcal{D}_w\langle h, -\mathbf{Diag}(\mathbf{A}(\mathbf{A}^\top\mathbf{W}\mathbf{A})^{-1}\mathbf{A}^\top)\rangle[h] &= \langle h, -\mathbf{Diag}(\mathbf{A}\mathcal{D}_w(\mathbf{A}^\top\mathbf{W}\mathbf{A})^{-1}[h]\mathbf{A}^\top)\rangle \\
&= \langle h, \mathbf{Diag}(\mathbf{A}(\mathbf{A}^\top\mathbf{W}\mathbf{A})^{-1}\mathcal{D}_w(\mathbf{A}^\top\mathbf{W}\mathbf{A})[h]\mathbf{A}^\top\mathbf{W}\mathbf{A})^{-1}\mathbf{A}^\top)\rangle \\
&= \langle h, \mathbf{Diag}(\mathbf{M}\mathbf{H}\mathbf{M})\rangle \\
&= \sum_{i,j} h_i h_j \mathbf{M}_{ij}\mathbf{M}_{ji} = \sum_{i,j} h_i h_j \mathbf{M}_{ij}^2,
\end{aligned}$$

where the last step follows by symmetry of $\mathbf{M}$. This implies

$$\nabla^2_{ij}\mathcal{F}(w) = \begin{cases} (a_i^\top(\mathbf{A}^\top\mathbf{W}\mathbf{A})^{-1}a_j)^2 & \text{if } i \neq j \\ (a_i^\top(\mathbf{A}^\top\mathbf{W}\mathbf{A})^{-1}a_j)^2 + \alpha w_i^{\alpha-1} & \text{otherwise} \end{cases},$$

which, in shorthand, is $\nabla^2 \mathcal{F}(w) = \mathbf{M} \circ \mathbf{M} + \alpha \mathbf{W}^{\alpha-1}$. We may express this Hessian as in the statement of the lemma by writing $\mathbf{M}$ in terms of $\mathbf{P}(w)$. □

Lemma 7.2.4 (Initial Sub-Optimality)**.** *At the start of Algorithm 7.2.1, the value of the objective of (7.1.4) differs from the optimum objective value as* $\mathcal{F}(w^{(0)}) \leq \mathcal{F}(\overline{w}) + n \log(m/n)$.

Proof. We study the two terms constituting the objective in (7.1.4). First, by choice of $w^{(0)} = \frac{n}{m}\mathbf{1}$, we have

$$-\log\det\Big(\mathbf{A}^\top \mathbf{W}^{(0)} \mathbf{A}\Big) = -\log\det\Big((n/m)\mathbf{A}^\top \mathbf{A}\Big). \tag{F.1.1}$$

Next, since leverage scores always lie between zero and one, the optimality condition for (7.1.4), $\sigma(\overline{w}) = (\overline{w})^{1+\alpha}$, implies $\overline{w} \leq 1$, which in turn gives $\overline{\mathbf{W}} \leq I$. This implies $\mathbf{A}^\top \overline{\mathbf{W}} \mathbf{A} \leq \mathbf{A}^\top \mathbf{A}$. Therefore,

$$-\log\det\Big(\mathbf{A}^\top \mathbf{A}\Big) \leq -\log\det\Big(\mathbf{A}^\top \overline{\mathbf{W}} \mathbf{A}\Big). \tag{F.1.2}$$

Combining (F.1.1) and (F.1.2) gives

$$-\log\det\Big(\mathbf{A}^\top \mathbf{W}^{(0)} \mathbf{A}\Big) \leq -\log\det\Big(\mathbf{A}^\top \overline{\mathbf{W}} \mathbf{A}\Big) + n \log(m/n). \tag{F.1.3}$$

Next, observe that $\mathbf{1}^\top (w^{(0)})^{1+\alpha} = m \cdot (n/m)^{1+\alpha}$, and $\mathbf{1}^\top (\overline{w})^{1+\alpha} = \sum_{i=1}^m \sigma_i(\overline{w}) = n$, where we invoked Fact 7.1.6. By now applying $m \geq n$, we get

$$\mathbf{1}^\top (w^{(0)})^{1+\alpha} \leq \mathbf{1}^\top (\overline{w})^{1+\alpha}. \tag{F.1.4}$$

Combining (F.1.3), (F.1.4), and the definition of the objective (7.1.4) finishes the claim. □

Lemma 7.2.6 (Function Decrease in **Descent**$(\cdot)$)**.** *Let* $w, \eta \in \mathbb{R}^m_{>0}$ *with* $\eta_i \in [0, \frac{1}{3\widetilde{\alpha}}]$ *for all* $i \in [m]$. *Further, let* $w^+ = \textbf{\textit{Descent}}(w, [m], \eta)$, *where* ***Descent*** *is defined in Equation (7.2.2). Then,* $w^+ \in \mathbb{R}^m_{>0}$ *with the following decrease in function objective.*

$$\mathcal{F}(w^+) \leq \mathcal{F}(w) - \sum_{i \in [m]} \frac{\eta_i}{2} \cdot w_i^{1+\alpha} \cdot \frac{(\rho_i(w) - 1)^2}{\rho_i(w) + 1}.$$

Proof. By the remainder form of Taylor's theorem, for some $t \in [0, 1]$ and $\widetilde{w} = tw + (1-t)w^+$

$$\mathcal{F}(w^+) = \mathcal{F}(w) + \langle \nabla \mathcal{F}(w), w^+ - w \rangle + \frac{1}{2}(w^+ - w)^\top \nabla^2 \mathcal{F}(\widetilde{w})(w^+ - w). \tag{F.1.5}$$

We prove the result by bounding the quadratic form of $\nabla^2 \mathcal{F}(\widetilde{w})$ from above and leveraging the structure of w^+ and $\nabla \mathcal{F}(w)$. Lemma 7.2.3 and Fact 7.1.6 imply that

$$\nabla^2 \mathcal{F}(\widetilde{w}) = \widetilde{\mathbf{W}}^{-1} \mathbf{P}(\widetilde{w})^{(2)} \widetilde{\mathbf{W}}^{-1} + \alpha \widetilde{\mathbf{W}}^{\alpha-1} \leq \widetilde{\mathbf{W}}^{-1} \Sigma(\widetilde{w}) \widetilde{\mathbf{W}}^{-1} + \alpha \widetilde{\mathbf{W}}^{\alpha-1}. \tag{F.1.6}$$

Further, the positivity of w_i and $\sigma_i(w)$ and the non-negativity of η and ρ imply that $(1 - \|\eta\|_\infty) w_i \leq w_i^+ \leq (1 + \|\eta\|_\infty) w_i$ for all $i \in [m]$. Since $\|\eta\|_\infty \leq \frac{1}{3\widetilde{\alpha}}$, this implies that

$$(1 - \tfrac{1}{3\widetilde{\alpha}}) w_i \leq \widetilde{w}_i \leq (1 + \tfrac{1}{3\widetilde{\alpha}}) w_i \quad \text{for all} \quad i \in [m].$$

Consequently, for all $i \in [m]$, we bound the first term of (F.1.6) as

$$\begin{aligned}\left[\widetilde{\mathbf{W}}^{-1}\Sigma(\widetilde{w})\widetilde{\mathbf{W}}^{-1}\right]_{ii} &= e_i^\top \widetilde{\mathbf{W}}^{-1/2}\mathbf{A}(\mathbf{A}^\top\widetilde{\mathbf{W}}\mathbf{A})^{-1}\mathbf{A}^\top\widetilde{\mathbf{W}}^{-1/2}e_i = \frac{1}{\widetilde{w}_i}a_i^\top(\mathbf{A}^\top\widetilde{\mathbf{W}}\mathbf{A})^{-1}a_i \\ &\le (1-\tfrac{1}{3\bar{\alpha}})^{-1}\frac{1}{w_i}a_i^\top(\mathbf{A}^\top\widetilde{\mathbf{W}}\mathbf{A})^{-1}a_i \le (1-\tfrac{1}{3\bar{\alpha}})^{-2}\frac{1}{w_i}a_i^\top(\mathbf{A}^\top\mathbf{W}\mathbf{A})^{-1}a_i \\ &= (1-\tfrac{1}{3\bar{\alpha}})^{-2}\left[\mathbf{W}^{-1}\Sigma(w)\mathbf{W}^{-1}\right]_{ii} \le 3\left[\mathbf{W}^{-1}\Sigma(w)\mathbf{W}^{-1}\right]_{ii} \end{aligned} \tag{F.1.7}$$

Further, when $\alpha \in (0,1]$, we bound the second term of (F.1.6) as

$$\widetilde{\mathbf{W}}^{\alpha-1} \le (1-\tfrac{1}{3\bar{\alpha}})^{\alpha-1}\mathbf{W}^{\alpha-1} \le (1-\tfrac{1}{3\bar{\alpha}})^{-1}\mathbf{W}^{\alpha-1} \le 3\mathbf{W}^{\alpha-1}, \tag{F.1.8}$$

and when $\alpha \ge 1$, we have

$$\widetilde{\mathbf{W}}^{\alpha-1} \le (1+\tfrac{1}{3\bar{\alpha}})^{\alpha-1}\mathbf{W}^{\alpha-1} \le \exp\left(\frac{\alpha-1}{3\bar{\alpha}}\right)\mathbf{W}^{\alpha-1} = \exp\left(\frac{\alpha-1}{3\alpha}\right)\mathbf{W}^{\alpha-1} \le 3\mathbf{W}^{\alpha-1}. \tag{F.1.9}$$

Using (F.1.7), (F.1.8), and (F.1.9) in (F.1.6), we have that in all cases

$$\nabla^2\mathcal{F}(\widetilde{w}) \preceq 3\left[\mathbf{W}^{-1}\Sigma(w)\mathbf{W}^{-1} + \alpha\mathbf{W}^{\alpha-1}\right] \preceq 3\bar{\alpha}\mathbf{W}^{-1}\left[\Sigma(w) + \mathbf{W}^{1+\alpha}\right]\mathbf{W}^{-1}.$$

Applying to the above Loewner inequality the definition of w^+ gives

$$\begin{aligned}(w^+ - w)^\top\nabla^2\mathcal{F}(\widetilde{w})(w^+ - w) &\le \sum_{i\in[m]} 3\bar{\alpha}\cdot(w_i^{1+\alpha} + \sigma_i(w))\cdot\left(\eta_i\cdot\frac{\rho_i(w)-1}{\rho_i(w)+1}\right)^2 \\ &= \sum_{i\in[m]} 3\bar{\alpha}\cdot\eta_i^2\cdot w_i^{1+\alpha}\cdot\frac{(\rho_i(w)-1)^2}{\rho_i(w)+1}.\end{aligned} \tag{F.1.10}$$

Next, recall that by Lemma 7.2.3, $[\nabla\mathcal{F}(w)]_i = w_i^{-1}\cdot(w_i^{1+\alpha} - \sigma_i(w))$ for all $i \in [m]$. Consequently,

$$\langle\nabla\mathcal{F}(w), w^+ - w\rangle = \sum_{i\in[m]}(w_i^{1+\alpha} - \sigma_i(w))\cdot\left(\eta_i\cdot\frac{\rho_i(w)-1}{\rho_i(w)+1}\right) = -\sum_{i\in[m]}\eta_i\cdot w_i^{1+\alpha}\cdot\frac{(\rho_i(w)-1)^2}{\rho_i(w)+1}. \tag{F.1.11}$$

Combining (F.1.5), (F.1.10), and (F.1.11) yields that

$$\mathcal{F}(w^+) \le \mathcal{F}(w) + \sum_{i\in[m]}\left(-\eta_i + \frac{3\bar{\alpha}\eta_i^2}{2}\right)\cdot w_i^{1+\alpha}\cdot\frac{(\rho_i(w)-1)^2}{\rho_i(w)+1}.$$

The result follows by plugging in $\eta_i \in [0, (3\bar{\alpha})^{-1}]$, as assumed. □

F.2 From Optimization Problem to Lewis Weights

The goal of this section is to prove how to obtain ϵ-approximate Lewis weights from an $\widetilde{\varepsilon}$-approximate solution to the problem in (7.1.4). Our proof strategy is to first utilize the fact that the vector w_{R} obtained after the rounding step following the `for` loop of Algorithm 7.2.1 satisfies the properties of being $\widetilde{\varepsilon}$-suboptimal (additively) and also the rounding condition (7.2.3). In Lemma 7.2.1, the $\widetilde{\varepsilon}$-suboptimality is used to show a bound on $\|\sigma(w_{\mathrm{R}}) - w_{\mathrm{R}}^{1+\alpha}\|_\infty$. Coupled with the rounding condition, we then show in Lemma F.2.1 that $\widehat{w_{\mathrm{R}}}$ constructed as per the last line of Algorithm 7.2.1 then satisfies

approximate optimality, $\sigma(\widehat{w}) \approx_\delta \widehat{w}^{1+\alpha}$, for some small $\delta > 0$. In Lemma F.2.3, we finally relate this approximate optimality to coordinate-wise multiplicative closeness between $\widehat{w}$ and the vector of true Lewis weights. Finally, in Lemma 7.2.1, we pick the appropriate approximation factors for each of the lemmas invoked and prove the desired approximation. Since the vector $w^{\mathcal{T}_{\text{total}}}$ obtained at the end of the `for` loop of Algorithm 7.5.1 also satisfies the aforementioned properties of w_{R}, the same set of lemmas apply to Algorithm 7.5.1 as well. We begin with some technical lemmas.

F.2.1 From Approximate Closeness to Approximate Optimality

Lemma F.2.1. *Let $w \in \mathbb{R}^m_{>0}$ such that $\|\sigma(w) - w^{1+\alpha}\|_\infty \leq \overline{\varepsilon}$ for some parameter $0 < \overline{\varepsilon} \leq \frac{1}{100m^2(\alpha+\alpha^{-1})^2}$ and also let $\rho_{\max}(w) \leq 1 + \alpha$. Define $\widehat{w}_i = (a_i^\top (\mathbf{A}^\top \mathbf{W}\mathbf{A})^{-1} a_i)^{1/\alpha}$. Then, for the parameter $\delta = 20\sqrt{\overline{\varepsilon}}m(\alpha + \alpha^{-1})$, we have that $\sigma(\widehat{w}) \approx_\delta \widehat{w}^{1+\alpha}$.*

Proof. Our strategy to prove $\sigma(\widehat{w}) \approx_\delta \widehat{w}^{1+\alpha}$ involves first noting that this is the same as proving $\widehat{w}^{-1} \cdot \sigma(\widehat{w}) \approx_\delta \widehat{w}^{\alpha}$ and, from the definition of $\widehat{w}$ in the statement of the lemma, to instead prove $\mathbf{A}^\top \widehat{\mathbf{W}} \mathbf{A} \approx_\delta \mathbf{A}^\top \mathbf{W}\mathbf{A}$.

To this end, we split $\mathbf{W}$ into two matrices based on the size of its coordinates, setting the following notation. Define $\mathbf{W}_{w \leq \eta}$ to be the diagonal matrix $\mathbf{W}$ with zeroes at indices corresponding to $w > \eta$, and $\widehat{\mathbf{W}}_{w \leq \eta}$ to be the diagonal matrix $\widehat{\mathbf{W}}$ with zeroes at indices corresponding to $w > \eta$. We first show that $\mathbf{A}^\top \widehat{\mathbf{W}}_{w \leq \eta} \mathbf{A}$ and $\mathbf{A}^\top \mathbf{W}_{w \leq \eta} \mathbf{A}$ are small compared to $\mathbf{A}^\top \mathbf{W}\mathbf{A}$ and can therefore be ignored in the preceding desired approximation. We then prove that for $w > \eta$, we have $w \approx_\delta \widehat{w}$. This proof technique is inspired by Lemma 4 of [Vai89a].

First, we prove that $\mathbf{A}^\top \widehat{\mathbf{W}}_{w \leq \eta} \mathbf{A}$ is small as compared to $\mathbf{A}^\top \mathbf{W}_{w > \eta} \mathbf{A}$. Since (7.2.3) is satisfied, it means

$$a_i^\top (\mathbf{A}^\top \mathbf{W}\mathbf{A})^{-1} a_i = \sigma_i(w) \cdot w_i^{-1} \leq (1 + \alpha) w_i^\alpha.$$

Combining this with the definition of $\widehat{w}_i$ as in the statement of the lemma, we may use non-negativity of α to derive

$$\widehat{w}_i \leq (1 + \alpha)^{1/\alpha} w_i \leq 3w_i. \tag{F.2.1}$$

We apply this inequality in the following expression to obtain

$$\begin{aligned}
\text{Tr}\Big((\mathbf{A}^\top \widehat{\mathbf{W}}_{w \leq \eta} \mathbf{A})(\mathbf{A}^\top \mathbf{W}\mathbf{A})^{-1}\Big) &= \sum_{w_i \leq \eta} \widehat{w}_i (a_i^\top (\mathbf{A}^\top \mathbf{W}\mathbf{A})^{-1} a_i) \\
&= \sum_{w_i \leq \eta} (a_i^\top (\mathbf{A}^\top \mathbf{W}\mathbf{A})^{-1} a_i)^{1+1/\alpha} \\
&\leq (1 + \alpha)^{1+1/\alpha} \sum_{w_i \leq \eta} w_i^{1+\alpha} \\
&\leq 3(1 + \alpha) m \eta^{1+\alpha}.
\end{aligned} \tag{F.2.2}$$

This implies that[1]

$$\mathbf{A}^\top \widehat{\mathbf{W}}_{w \leq \eta} \mathbf{A} \leq 3(1 + \alpha) m \eta^{1+\alpha} \mathbf{A}^\top \mathbf{W}\mathbf{A}. \tag{F.2.3}$$

Our next goal is to bound $\mathbf{A}^\top \widehat{\mathbf{W}}_{w > \eta} \mathbf{A}$ in terms of $\mathbf{A}^\top \mathbf{W}\mathbf{A}$, which we do by first bounding it in terms

[1] Given $X, Y \succeq 0$, we have $Y^{1/2} X Y^{1/2} \succeq 0$. Then, if $\text{Tr}(XY) \leq 1$, we have $\text{Tr}\big(Y^{1/2} X Y^{1/2}\big) \leq 1$, and combining these with the previous matrix inequality, we conclude that $Y^{1/2} X Y^{1/2} \preceq I$, which implies that $X \preceq Y^{-1}$.

of $\mathbf{A}^\top \mathbf{W}_{w>\eta}\mathbf{A}$ and then bounding $\mathbf{A}^\top \mathbf{W}_{w>\eta}\mathbf{A}$ in terms of $\mathbf{A}^\top \mathbf{W}\mathbf{A}$. By definition, $\widehat{w}_i^\alpha = \sigma_i(w) \cdot w_i^{-1}$. Further, by assumption, $\|\sigma(w) - w^{1+\alpha}\|_\infty \le \overline{\varepsilon}$. Therefore, for any $w_i \ge \eta$

$$\widehat{w}_i^\alpha \le (w_i^{1+\alpha} + \overline{\varepsilon}) \cdot w_i^{-1} \le (1 + \overline{\varepsilon}/\eta^{1+\alpha}) w_i^{1+\alpha} \cdot w_i^{-1} = (1 + \overline{\varepsilon}/\eta^{1+\alpha}) w_i^\alpha,$$

and

$$\widehat{w}_i^\alpha \ge (w_i^{1+\alpha} - \overline{\varepsilon}) \cdot w_i^{-1} \ge (1 - \overline{\varepsilon}/\eta^{1+\alpha}) w_i^{1+\alpha} \cdot w_i^{-1} = (1 - \overline{\varepsilon}/\eta^{1+\alpha}) w_i^\alpha.$$

By our choice of $\overline{\varepsilon}$, for $w_i \ge \eta$, we have

$$\left(1 - \frac{2\overline{\varepsilon}}{\alpha\eta^{1+\alpha}}\right) w_i \le \widehat{w}_i \le \left(1 + \frac{2\overline{\varepsilon}}{\alpha\eta^{1+\alpha}}\right) w_i. \tag{F.2.4}$$

Further, we have the following inequality:

$$\mathbf{A}^\top \mathbf{W}_{w>\eta}\mathbf{A} \preceq \mathbf{A}^\top \mathbf{W}\mathbf{A}. \tag{F.2.5}$$

Hence, we can combine Inequality F.2.5, Inequality F.2.4, and Inequality F.2.3 to see that

$$\begin{aligned}
\mathbf{A}^\top \widehat{\mathbf{W}}\mathbf{A} &= \mathbf{A}^\top \widehat{\mathbf{W}}_{w>\eta}\mathbf{A} + \mathbf{A}^\top \widehat{\mathbf{W}}_{w\le\eta}\mathbf{A} \\
&\preceq \left(1 + \frac{2\overline{\varepsilon}}{\alpha\eta^{1+\alpha}}\right) \mathbf{A}^\top \mathbf{W}_{w>\eta}\mathbf{A} + 3(1+\alpha) m\eta^{1+\alpha} \mathbf{A}^\top \mathbf{W}\mathbf{A} \\
&\preceq \mathbf{A}^\top \mathbf{W}\mathbf{A} \left(1 + \frac{2\overline{\varepsilon}}{\alpha\eta^{1+\alpha}} + 3(1+\alpha) m\eta^{1+\alpha}\right).
\end{aligned}$$

Set $\eta^{1+\alpha} = \sqrt{\overline{\varepsilon}}$ for the upper bound.

For the lower bound, we bound $\mathbf{A}^\top \mathbf{W}_{w\le\eta}\mathbf{A}$ and, therefore, also $\mathbf{A}^\top \mathbf{W}_{w>\eta}\mathbf{A}$. Observe that

$$\begin{aligned}
\mathrm{Tr}\Big((\mathbf{A}^\top \mathbf{W}_{w\le\eta}\mathbf{A})(\mathbf{A}^\top \mathbf{W}\mathbf{A})^{-1}\Big) &= \sum_{w_i\le\eta} w_i a_i^\top (\mathbf{A}^\top \mathbf{W}\mathbf{A})^{-1} a_i = \sum_{w_i\le\eta} \sigma_i(w) \\
&\le \sum_{w_i\le\eta} (w_i^{1+\alpha} + \overline{\varepsilon}) \le m(\eta^{1+\alpha} + \overline{\varepsilon}),
\end{aligned}$$

where the second step is by $\|\sigma(w) - w^{1+\alpha}\|_\infty \le \overline{\varepsilon}$, as assumed in the lemma. This implies that

$$\mathbf{A}^\top \mathbf{W}_{w\le\eta}\mathbf{A} \preceq m(\eta^{1+\alpha} + \overline{\varepsilon}) \mathbf{A}^\top \mathbf{W}\mathbf{A},$$

and therefore that

$$\mathbf{A}^\top \mathbf{W}_{w>\eta}\mathbf{A} \succeq (1 - m(\eta^{1+\alpha} + \overline{\varepsilon})) \mathbf{A}^\top \mathbf{W}\mathbf{A}.$$

Repeating the method for the upper bound then finishes the proof. □

F.2.2 From Approximate Optimality to Approximate Lewis Weights

In this section, we go from the previous notion of approximation to the one we finally seek in Equation (7.1.5). Specifically, we show that if $\sigma(w) \approx_\beta w^{1+\alpha}$, then $w \approx_{O((\beta/\alpha)\sqrt{n})} \overline{w}$. To prove this, we first give a technical result. We recall notation stated in Section 7.1.4: for any projection matrix $\mathbf{P}(w) \in \mathbb{R}^{m\times m}$, we have the vector of leverage scores $\sigma(w) = \mathbf{diag}(\mathbf{P}(w))$.

Claim F.2.2. *For any projection matrix $\mathbf{P}(w) \in \mathbb{R}^{m\times m}$, $\alpha \geq 0$, and vector $x \in \mathbb{R}^m$, we have that*

$$\|\left[\mathbf{P}(w)^{(2)} + \alpha\mathbf{\Sigma}(w)\right]^{-1}\mathbf{\Sigma}(w)x\|_\infty \leq \frac{1}{\alpha}\|x\|_\infty + \frac{1}{\alpha^2}\|x\|_{\mathbf{\Sigma}(w)} \leq \left(\frac{1+\sqrt{n}/\alpha}{\alpha}\right)\|x\|_\infty$$

Proof. Let $y \stackrel{\text{def}}{=} \left[\mathbf{P}(w)^{(2)} + \alpha\mathbf{\Sigma}(w)\right]^{-1}\mathbf{\Sigma}(w)x$. Since $\mathbf{0} \preceq \mathbf{P}(w)^{(2)} \preceq \mathbf{\Sigma}(w)$ (Fact 7.1.6), we have that $\mathbf{\Sigma}(w) \preceq \frac{1}{\alpha}\left[\mathbf{P}(w)^{(2)} + \alpha\mathbf{\Sigma}(w)\right]$ and $(\mathbf{P}(w)^{(2)} + \alpha\mathbf{\Sigma}(w))^{-1} \preceq \alpha^{-1}\mathbf{\Sigma}(w)^{-1}$. Consequently, taking norms in terms of these matrices gives

$$\|y\|_{\mathbf{\Sigma}(w)} = \|\left[\mathbf{P}(w)^{(2)} + \alpha\mathbf{\Sigma}(w)\right]^{-1}\mathbf{\Sigma}(w)x\|_{\mathbf{\Sigma}(w)} \leq \frac{1}{\sqrt{\alpha}}\|\mathbf{\Sigma}(w)x\|_{\left[\mathbf{P}(w)^{(2)}+\alpha\mathbf{\Sigma}(w)\right]^{-1}} \leq \frac{1}{\alpha}\|x\|_{\mathbf{\Sigma}(w)}\,. \tag{F.2.6}$$

Next, since by Lemma 47 of [LS14], $\|\mathbf{\Sigma}(w)^{-1}\mathbf{P}(w)^{(2)}z\|_\infty \leq \|z\|_{\mathbf{\Sigma}(w)}$ for all $z \in \mathbb{R}^m$, we see that $\left|[\mathbf{P}(w)^{(2)}y]_i\right| \leq \sigma_i(w)\|y\|_{\mathbf{\Sigma}(w)}$ for all $i \in [m]$, and since by definition of y, we have $[(\mathbf{P}(w)^{(2)} + \alpha\mathbf{\Sigma}(w))y]_i = \sigma_i(w)x_i$ for all $i \in [m]$, we have that

$$\|y\|_\infty = \max_{i\in[m]}|y_i| = \max_{i\in[m]}\left|\frac{1}{\alpha}x_i + \frac{1}{\alpha\sigma_i(w)}\left[\mathbf{P}(w)^{(2)}y\right]_i\right| \leq \frac{1}{\alpha}\|x\|_\infty + \frac{1}{\alpha}\|y\|_{\mathbf{\Sigma}(w)}\,. \tag{F.2.7}$$

Combining Inequality F.2.6 and Inequality F.2.7 and using that $\sum_{i\in[m]}\sigma_i(w) \leq n$ yields the claim. □

Lemma F.2.3. *Let $\widehat{w} \in \mathbb{R}^m_{>0}$ be a vector that satisfies approximate optimality of (7.1.4) in the following sense:*

$$\sigma(\widehat{w}) = \widehat{\mathbf{W}}^{1+\alpha}v, \text{ for } \exp(-\mu)\mathbf{1} \leq v \leq \exp(\mu)\mathbf{1}.$$

Then, $\widehat{w}$ is also coordinate-wise multiplicatively close to $\overline{w}$, the true vector of Lewis weights, as formalized below.

$$\exp\left(-\frac{1}{\alpha}(1+\sqrt{n}/\alpha)\mu\right)\overline{w} \leq \widehat{w} \leq \exp\left(\frac{1}{\alpha}(1+\sqrt{n}/\alpha)\mu\right)\overline{w}\,.$$

Proof. For all $t \in [0,1]$, let $[v_t]_i = [v_i^t]$ so that $v_1 = v$ and $v_0 = \mathbf{1}$. Further, for all $t \in [0,1]$, let w_t be the unique solution to

$$w_t = \text{argmin}_{w\in\mathbb{R}^m_{>0}} f_t(w) \stackrel{\text{def}}{=} -\log\det\left(\mathbf{A}^\top\mathbf{W}\mathbf{A}\right) + \frac{1}{1+\alpha}\sum_{i\in[m]}[v_t]_i w_i^{1+\alpha}. \tag{F.2.8}$$

Then we have the following gradients.

$$\begin{aligned}
\nabla_w f_t(w) &= -\mathbf{W}^{-1}\sigma(w) + \mathbf{W}^\alpha v_t\,, \\
\nabla_w(\frac{d}{dt}f_t)(w) &= \mathbf{W}^\alpha\frac{d}{dt}v_t = \mathbf{W}^\alpha v_t\ln(v) && \text{(F.2.9)} \\
\nabla^2_{ww} f_t(w) &= \mathbf{W}^{-1}\left[\mathbf{P}(w)^{(2)} + \alpha\mathbf{W}^{1+\alpha}\mathbf{V}\right]\mathbf{W}^{-1}\,. && \text{(F.2.10)}
\end{aligned}$$

Consequently, by optimality of w_t as defined in (F.2.8), we have $\mathbf{0} = \nabla_w f_t(w_t) = -\mathbf{W}_t^{-1}\sigma(w_t) + \mathbf{W}_t^\alpha v_t$. Rearranging the terms of this equation yields that

$$\sigma(w_t) = \mathbf{W}_t^{1+\alpha}v_t, \tag{F.2.11}$$

and therefore $w_1 = \widehat{w}$ and $w_0 = \overline{w}$. To prove the lemma, it therefore suffices to bound

$$\ln(\widehat{w}/\overline{w}) = \ln(w_1/w_0) = \int_{t=0}^{1} \left[\frac{d}{dt}\ln(w_t)\right] dt = \int_{t=0}^{1} \mathbf{W}_t^{-1}\left[\frac{d}{dt}w_t\right] dt\,. \tag{F.2.12}$$

To bound Equation (F.2.12), it remains to compute $\frac{d}{dt}w_t$ and apply Claim F.2.2. To do this, note that

$$\mathbf{0} = \frac{d}{dt}\boldsymbol{\nabla}_w\left[f_t(w_t)\right] = \nabla_w(\frac{d}{dt}f_t)(w_t) + \nabla^2_{ww}f_t(w_t)\cdot\frac{d}{dt}w_t\,.$$

Using that $\mathbf{P}(w_t)^{(2)} + \mathbf{W}_t^{1+\alpha}\mathbf{V}_t \succ \mathbf{0}$, we have, by rearranging the above equation and applying Equation (F.2.9) and Equation (F.2.10) that

$$\frac{d}{dt}w_t = -\left[\nabla^2_{ww}f_t(w_t)\right]^{-1}\cdot\left[\nabla_w(\frac{d}{dt}f_t)(w_t)\right] = -\mathbf{W}_t\left[\mathbf{P}(w_t)^{(2)} + \alpha\mathbf{W}_t^{1+\alpha}\mathbf{V}_t\right]^{-1}\mathbf{W}_t^{1+\alpha}v_t\ln(v)\,. \tag{F.2.13}$$

Applying Equation (F.2.11) to Equation (F.2.13), we have that

$$\mathbf{W}_t^{-1}\left[\frac{d}{dt}w_t\right] = -\left[\mathbf{P}(w_t)^{(2)} + \alpha\boldsymbol{\Sigma}(w_t)\right]^{-1}\boldsymbol{\Sigma}(w_t)\ln(v)\,.$$

Applying Claim F.2.2 to the above equality, substituting in Equation (F.2.12) and $\|\ln(v)\|_\infty \le \mu$ therefore yields

$$\|\ln(\widehat{w}/\overline{w})\|_\infty = \|\ln(w_1/w_0)\|_\infty \le \int_{t=0}^{1}\|\mathbf{W}_t^{-1}\left[\frac{d}{dt}w_t\right]\|_\infty dt \le \int_{t=0}^{1}\left(\frac{1+\sqrt{n}/\alpha}{\alpha}\right)\mu dt\,.$$

□

F.2.3 From Optimization Problem to Approximate Lewis Weights

Lemma 7.2.1 (Lewis Weights from Optimization Solution)**.** *Let $w \in \mathbb{R}^m_{>0}$ be a vector at which the objective (7.1.4) is $\widetilde{\varepsilon}$-suboptimal in the additive sense for $\widetilde{\varepsilon} = \frac{\alpha^8\epsilon^4}{(25m(\sqrt{n}+\alpha)(\alpha+\alpha^{-1}))^4}$, i.e., $\mathcal{F}(\overline{w}) \le \mathcal{F}(w) \le \mathcal{F}(\overline{w}) + \widetilde{\varepsilon}$. Further assume that w satisfies the rounding condition: $\rho_{\max}(w) \le 1+\alpha$. Then, the vector $\widehat{w}$ defined as $\widehat{w}_i = (a_i^\top(\mathbf{A}^\top\mathbf{W}\mathbf{A})^{-1}a_i)^{1/\alpha}$ satisfies $\widehat{w}_i \approx_\epsilon \overline{w}_i$ for all $i \in [m]$, thus achieving the goal spelt out in Equation (7.1.5).*

Proof. We are given a vector $w \in \mathbb{R}^m$ satisfying $\mathcal{F}(\overline{w}) \le \mathcal{F}(w) \le \mathcal{F}(\overline{w}) + \widetilde{\varepsilon}$. Then by Lemma 7.2.5, we have that $\frac{(\sigma_i(w)-w_i^{1+\alpha})^2}{\sigma_i(w)+w_i^{1+\alpha}} \le \widetilde{\varepsilon}$ for each $i \in [m]$. This bound implies that $w_i \le 3$ for all i because, if not, then because of $\sigma_i(w) \in [0,1]$ and the decreasing nature of $(x-a)^2/(x+a)$ over $x \in [0,1]$ for a fixed $a \ge 3$, we obtain $\frac{(\sigma_i(w)-w_i^{1+\alpha})^2}{\sigma_i(w)+w_i^{1+\alpha}} \ge \frac{(1-w_i^{1+\alpha})^2}{1+w_i^{1+\alpha}} \ge 1$, a contradiction. Therefore $\|\sigma(w) - w^{1+\alpha}\|_\infty \le 2\sqrt{\widetilde{\varepsilon}}$. Coupled with the provided guarantee $\rho_{\max}(w) \le 1+\alpha$, we see that the requirements of Lemma F.2.1 are met with $\overline{\varepsilon} = 2\sqrt{\widetilde{\varepsilon}}$, for $\widetilde{\varepsilon} \stackrel{\text{def}}{=} \frac{\widehat{\epsilon}^4}{(25m(\alpha+\alpha^{-1}))^4}$, and Algorithm 7.2.1 therefore guarantees a $\widehat{w}$ satisfying $\sigma(\widehat{w}) \approx_{\widehat{\epsilon}} \widehat{w}^{1+\alpha}$. Therefore, we can now apply Lemma F.2.3 with $\mu = \widehat{\epsilon}$, and choosing $\widehat{\epsilon} = \frac{\alpha^2}{\alpha+\sqrt{n}}\epsilon$ lets us conclude that $\widehat{w}_i \approx_\epsilon \overline{w}_i$, as claimed. □

F.3 A Geometric View of Rounding

At the end of Algorithm 7.2.2 and Algorithm 7.2.3, the iterate w satisfies the condition $\rho_{\max}(w) \leq 1+\alpha$. We now show the geometry implied by the preceding condition, thereby provide the reason behind the terminology "rounding."

Lemma F.3.1. *Given $w \in \mathbb{R}^m_{>0}$ such that $\rho_{\max}(w) \leq 1+\alpha$. Define the ellipsoid $\mathcal{E}(w) := \{x : x^\top \mathbf{A}^\top \mathbf{W}\mathbf{A}x \leq 1\}$. Then, we have that*

$$\mathcal{E}(w) \subset \{x \in \mathbb{R}^n \mid \|\mathbf{W}^{-\alpha/2}\mathbf{A}x\|_\infty \leq \sqrt{1+\alpha}\}.$$

Proof. Consider any point $x \in \mathcal{E}(w)$. Then, by Cauchy-Schwarz inequality and $\rho_{\max}(w) \leq 1+\alpha$,

$$\begin{aligned}
\|\mathbf{W}^{-\alpha/2}\mathbf{A}x\|_\infty &= \max_{i\in[m]} e_i^\top \mathbf{W}^{-\alpha/2}\mathbf{A}x = \max_{i\in[m]} e_i^\top \mathbf{W}^{-\alpha/2}\mathbf{A}(\mathbf{A}^\top\mathbf{W}\mathbf{A})^{-\frac{1}{2}}(\mathbf{A}^\top\mathbf{W}\mathbf{A})^{\frac{1}{2}}x \\
&\leq \max_{i\in[m]} \sqrt{e_i^\top \mathbf{W}^{-\alpha/2}\mathbf{A}(\mathbf{A}^\top\mathbf{W}\mathbf{A})^{-1}\mathbf{A}^\top\mathbf{W}^{-\alpha/2}e_i}\,\sqrt{x^\top\mathbf{A}^\top\mathbf{W}\mathbf{A}x} \\
&\leq \max_{i\in[m]} \sqrt{e_i^\top \mathbf{W}^{-\alpha/2}\mathbf{A}(\mathbf{A}^\top\mathbf{W}\mathbf{A})^{-1}\mathbf{A}^\top\mathbf{W}^{-\alpha/2}e_i} = \max_{i\in[m]} \sqrt{\frac{\sigma_i(w)}{w_i^{1+\alpha}}} \leq \sqrt{1+\alpha}.
\end{aligned}$$

□

F.4 Explanations of Runtimes in Prior Work

The convex program (7.1.3) formulated by [CP15] has a variable size of n^2. Therefore, by [LSW15], the number of iterations to solve it using the cutting plane method is $O(n^2 \log\left(n\epsilon^{-1}\right)$, each iteration computing $a_i^\top \mathbf{M} a_i$ for $i \in [m]$. This can be computed by multiplying an $n \times n$ matrix with an $n \times m$ matrix, which costs between $O(mn)$ (at least the size of the larger input matrix) and $O(mn^2)$ (each entry of the resulting $m \times n$ matrix obtained by an inner product of length n vectors). Further, there is at least a total of $O(n^6)$ additional work done by the cutting plane method. This gives us a cost of at least $n^2(mn + n^4)$. The runtime of [Lee16] follows from Theorem 5.3.4.

Appendix G

Appendix for Chapter 8

This chapter contains details and proofs from Chapter 8.

G.1 Proofs: Strict RoS Constraint

The goal of this section is to prove Proposition 8.4.2 and Proposition 8.4.3, for which we need the following definition and results.

Definition G.1.1. *We define the following dual variables.*

- *Let $f_t(b) := v_t \cdot x_t(b)$, recall g_t from Equation (8.2.3), and for some fixed $\lambda \geq 0$, define*

$$f^{\star}_{t,\mathsf{RoS}}(\lambda) := \max_{b \geq 0} \left[f_t(b) + \lambda \cdot g_t(b) \right]. \tag{G.1.1}$$

- *Let $f^{\star}$ be defined as in Equation (G.1.1). Then we define the following dual variable parametrized by the input distribution $\mathcal{P}$.*

$$\overline{\mathcal{D}}_{\mathsf{RoS}}(\lambda|\mathcal{P}) := \mathbb{E}_{(v,p)\sim\mathcal{P}} \left[f^{\star}_{\mathsf{RoS}}(\lambda) \right]. \tag{G.1.2}$$

Proposition G.1.2. *Recall $\overline{\mathcal{D}}_{\mathsf{RoS}}(\lambda|\mathcal{P})$ as defined in Equation (G.1.2). Then the optimum value $\mathsf{Reward}(\mathsf{Opt}, \overrightarrow{\gamma})$ for Problem 8.2.2 defined for a sequence $\overrightarrow{\gamma}_\ell \sim \mathcal{P}^\ell$ of ℓ requests satisfies the inequality*

$$\mathbb{E}_{\overrightarrow{\gamma}_\ell \sim \mathcal{P}^\ell} \left[\mathsf{Reward}(\mathsf{Opt}, \overrightarrow{\gamma}_\ell) \right] \leq \ell \cdot \min_{\lambda \geq 0} \overline{\mathcal{D}}_{\mathsf{RoS}}(\lambda|\mathcal{P}).$$

Proposition G.1.3. *For some fixed number r, let $\overline{\lambda}_r := \frac{1}{r}\sum_{t=1}^{r} \lambda_t$, where λ_t are the dual iterates in Algorithm 8.3.1. Then, the reward (see Equation (8.2.6)) of Algorithm 8.3.1 is lower bounded as*

$$\mathsf{Reward}(\textit{Algorithm 8.3.1}, \overrightarrow{\gamma}_r) \geq \mathbb{E}_{\overrightarrow{\gamma}_r \sim \mathcal{P}^r} \left[r \cdot \overline{\mathcal{D}}_{\mathsf{RoS}}(\overline{\lambda}_r|\mathcal{P}) \right] - \mathbb{E}_{\overrightarrow{\gamma}_r \sim \mathcal{P}^r} \left[\sum_{t=1}^{r} \lambda_t \cdot g_t(b_t) \right].$$

Proposition 8.4.2. *Under Assumption 8.4.1 for the distribution $\mathcal{P}$, let $K(\overrightarrow{\gamma})$ be the number of iterations in the first phase of Algorithm 8.4.1 for some input sequence $\overrightarrow{\gamma}$. Then, we have*

$$\mathbb{E}_{\overrightarrow{\gamma} \sim \mathcal{P}^T}[K(\overrightarrow{\gamma})] \leq O(\sqrt{T} \log T).$$

Proof. Let $z_t = \max(0, v_t \cdot x_t(b_t) - p_t(b_t))$ be the reward collected at iteration t (in the first phase). Let $z'_t := \frac{\beta}{2}\mathbf{1}_{z_t \geq \beta/2}$. By construction, $z'_t \leq z_t$ for all t. For any given sequence $\overrightarrow{\gamma}$, let $K(\overrightarrow{\gamma})$ and $K'(\overrightarrow{\gamma})$, respectively, be the first time such that $\sum_{t=1}^{K(\overrightarrow{\gamma})} z_t \geq R$ and $\sum_{t=1}^{K'(\overrightarrow{\gamma})} z'_t \geq R$ for some reward R. Then, for

every $\overrightarrow{\gamma}$, we have $K(\overrightarrow{\gamma}) \le K'(\overrightarrow{\gamma})$. By the boundedness assumption on z_t, we have

$$\textbf{Prob}(z'_t = \beta/2) = \textbf{Prob}(z_t \ge \beta/2) \ge \mathbb{E}[z_t] - \textbf{Prob}(z_t \le \beta/2) \cdot \mathbb{E}[z_t | z_t \le \beta/2] \ge \beta/2.$$

Then, by Hoeffding bound,

$$\textbf{Prob}(K(\overrightarrow{\gamma}) \ge q) \le \textbf{Prob}(K'(\overrightarrow{\gamma}) \ge q) \le e^{-O(q\beta^2)}. \tag{G.1.3}$$

Picking $q = O(R/\beta^2)$ for $R = 2\sqrt{T}\log T$ finishes the claim. □

Proposition 8.4.3. *Let $\overrightarrow{\gamma}_\ell \sim \mathcal{P}^\ell$ and $\overrightarrow{\gamma}_r \sim \mathcal{P}^r$ be sequences of lengths ℓ and r, respectively, with $\ell \le r$, of i.i.d. requests each from a distribution $\mathcal{P}$. Then the following inequality holds.*

$$\mathbb{E}_{\overrightarrow{\gamma}_\ell \sim \mathcal{P}^\ell}\left[\textsf{Reward}(\textit{Algorithm 8.3.1}, \overrightarrow{\gamma}_\ell)\right] \ge \frac{\ell}{r}\mathbb{E}_{\overrightarrow{\gamma}_r \sim \mathcal{P}^r}\left[\textsf{Reward}(\textsf{Opt}, \overrightarrow{\gamma}_r)\right] - O(\sqrt{r}).$$

Proof. By Proposition G.1.2, Proposition G.1.3, and Theorem 8.3.1,

$$\begin{aligned}
\textsf{Reward}(\textit{Algorithm 8.3.1}, \overrightarrow{\gamma}_\ell) &\ge \mathbb{E}_{\overrightarrow{\gamma}_\ell \sim \mathcal{P}^\ell}\left[\ell \cdot \overline{\mathcal{D}}_{\mathsf{RoS}}(\overline{\lambda}_\ell | \mathcal{P})\right] - \mathbb{E}_{\overrightarrow{\gamma}_\ell \sim \mathcal{P}^\ell}\left[\sum_{t=1}^{\ell} \lambda_t \cdot g_t(b_t)\right] \\
&\ge \ell \cdot \min_{\lambda \ge 0} \overline{\mathcal{D}}_{\mathsf{RoS}}(\lambda | \mathcal{P}) - \mathbb{E}_{\overrightarrow{\gamma}_\ell \sim \mathcal{P}^\ell}\left[\sum_{t=1}^{\ell} \lambda_t \cdot g_t(b_t)\right] \\
&\ge \frac{\ell}{r} \cdot \mathbb{E}_{\overrightarrow{\gamma}_r \sim \mathcal{P}^r}\left[\textsf{Reward}(\textsf{Opt}, \overrightarrow{\gamma}_r)\right] - O(\sqrt{r}).
\end{aligned}$$

□

G.2 Proofs: Both Strict Constraints

The goal of this section is proving Theorem 8.5.2.

Definition G.2.1. *We need the following definitions of dual variables.*

- *For some $\lambda \ge 0$ and $\mu \ge 0$, let $f_t(b) := v_t \cdot x_t(b)$, define g_t as in Equation (8.2.3), and define*

$$f^\star_{t,\textsf{combined}}(\mu, \lambda) := \max_b \left[f_t(b) + \lambda \cdot g_t(b)) - \mu \cdot p_t(b)\right]. \tag{G.2.1}$$

- *The following dual variable parametrized by ρ and $\mathcal{P}$; the quantity $f^\star$ is defined in the same way as in Equation (G.2.1).*

$$\overline{\mathcal{D}}_{\textsf{combined}}(\mu, \lambda | \mathcal{P}, \rho) := \mu \cdot \rho + \mathbb{E}_{(v,p)\sim\mathcal{P}}\left[f^\star_{\textsf{combined}}(\mu, \lambda)\right]. \tag{G.2.2}$$

Proposition G.2.2. *For some $\rho' \ge 0$, let $\overline{\mathcal{D}}_{\textsf{combined}}(\mu, \lambda | \mathcal{P}, \rho')$ be as defined in Equation (G.2.2). Then the optimum value $\textsf{Reward}(\textsf{Opt}, \overrightarrow{\gamma}_\ell, \rho)$ for Problem 8.2.5 with a total initial budget of $\rho\ell$ over a sequence $\overrightarrow{\gamma}_\ell \sim \mathcal{P}^\ell$ of ℓ requests satisfies the inequality*

$$\mathbb{E}_{\overrightarrow{\gamma}_\ell \sim \mathcal{P}^\ell}\left[\textsf{Reward}(\textsf{Opt}, \overrightarrow{\gamma}_\ell, \rho)\right] \le \ell \cdot \min_{\mu \ge 0, \lambda \ge 0}\left[\overline{\mathcal{D}}_{\textsf{combined}}(\mu, \lambda | \mathcal{P}, \rho') + (\rho - \rho') \cdot \mu\right].$$

Definition G.2.3. *The stopping time τ of Algorithm 8.5.1, with a total initial budget of B is the first time τ at which $\sum_{t=1}^{\tau} p_t(b_t) + 1 \geq B$. Intuitively, this is the first time step at which the total price paid almost exceeds the total budget.*

Proposition G.2.4. *Let τ be a stopping time as in Definition G.2.3 for some initial budget $\rho' k$. Let $\overline{\mu}_\tau = \frac{1}{\tau}\sum_{i=1}^{\tau} \mu_i$ and $\overline{\lambda}_\tau = \frac{1}{\tau}\sum_{i=1}^{\tau} \lambda_i$. Then the expected reward (see Equation (8.2.6)) of Algorithm 8.5.1 over a sequence of length k with i.i.d. input requests from distribution $\mathcal{P}^k$ is lower bounded as*

$$\mathbb{E}_{\overrightarrow{\gamma}_k \sim \mathcal{P}^k}\left[\textsf{Reward}(\textit{Algorithm 8.5.1}, \overrightarrow{\gamma}, \rho')\right] \geq \mathbb{E}_{\overrightarrow{\gamma}_k \sim \mathcal{P}^k}\left[\tau \cdot \overline{\mathcal{D}}_{\textsf{combined}}(\overline{\mu}_\tau, \overline{\lambda}_\tau | \mathcal{P}, \rho')\right] - \mathbb{E}_{\overrightarrow{\gamma}_k \sim \mathcal{P}^k}\left[\sum_{t=1}^{\tau} \mu_t \cdot (\rho' - p_t(b_t)) - \sum_{t=1}^{\tau} \lambda_t \cdot g_t(b_t)\right].$$

Proposition G.2.5. *Consider a run of Algorithm 8.5.1 with initial total budget $\rho\ell$ and the total time horizon ℓ. We define the corresponding stopping time (as defined in Definition G.2.3) as the time τ at which $\sum_{t=1}^{\tau} p_t(b_t) \geq \rho\ell - 1$. Then, the dual variable $\{\mu_t\}$ that evolves as per Line 8 in Algorithm 8.5.1 satisfies the inequality $\sum_{t=1}^{\tau} \mu_t \cdot (\rho - p_t(b_t)) \leq (\tau - \ell) + 1/\rho + O(\sqrt{\ell})$.*

Theorem 8.5.2. *With i.i.d. inputs from a distribution $\mathcal{P}$ over a time horizon T, the regret of Algorithm 8.5.2 on Problem 8.2.5 is, under Assumption 8.4.1, bounded by*

$$\textsf{Regret}(\textit{Algorithm 8.5.2}, \mathcal{P}^T) \leq O(\sqrt{T}\log T).$$

Further, Algorithm 8.5.2 suffers no constraint violation of either the RoS or budget constraint.

Proof. The RoS constraint is not violated because the first phase accumulates the buffer that is the guaranteed cap on violation in the second phase. The budget constraint is respected by design: the first phase pays at most ρT, and the second phase strictly respects the budget, as guaranteed by Algorithm 8.5.1. Next, we note that the total expected reward is at least that in the second phase

$$\mathbb{E}_{\overrightarrow{\gamma} \sim \mathcal{P}^T}\left[\textsf{Reward}(\textit{Algorithm 8.5.2}, \overrightarrow{\gamma}, \rho)\right] \geq \mathbb{E}_k\left[\mathbb{E}_{\overrightarrow{\gamma}_{k+1:T} \sim \mathcal{P}^{T-k}}\left(\textsf{Reward}(\textit{Algorithm 8.5.1}, \overrightarrow{\gamma}_{k+1:T}, \widehat{\rho})\right)\right], \tag{G.2.3}$$

where $\widehat{\rho} = \frac{\rho T - K(\overrightarrow{\gamma})}{T - K(\overrightarrow{\gamma})}$ and the right-hand side captures the reduced time horizon $T - K(\overrightarrow{\gamma})$ and reduced initial budget $\rho T - K(\overrightarrow{\gamma})$ for Algorithm 8.5.1. Conditioning on the high-probability event that $k \leq \rho T$ (by Inequality G.1.3 coupled with the assumption that ρ is a fixed constant) and letting $R = \textsf{Reward}(\textit{Algorithm 8.5.1}, \overrightarrow{\gamma}_{k+1:T}, \widehat{\rho})$:

$$\mathbb{E}_k\left[\mathbb{E}_{\overrightarrow{\gamma}_{k+1:T} \sim \mathcal{P}^{T-k}}(R)\right] \geq (1 - e^{-O(T)})\mathbb{E}_k\left[\mathbb{E}_{\overrightarrow{\gamma}_{k+1:T} \sim \mathcal{P}^{T-k}}(R) \mid k \leq \rho T\right]. \tag{G.2.4}$$

Applying Proposition G.2.4 with the reduced budget and time horizon:

$$\mathbb{E}_{\overrightarrow{\gamma}_{k+1:T} \sim \mathcal{P}^{T-k}}[R] \geq \mathbb{E}_{\overrightarrow{\gamma}_{k+1:T} \sim \mathcal{P}^{T-k}}\left[\tau \cdot \overline{\mathcal{D}}_{\textsf{combined}}(\overline{\mu}_\tau, \overline{\lambda}_\tau | \mathcal{P}, \widehat{\rho})\right] - \mathbb{E}_{\overrightarrow{\gamma}_{k+1:T} \sim \mathcal{P}^{T-k}}\left[\sum_{t=1}^{\tau} \mu_t \cdot (\widehat{\rho} - p_t(b_t))\right] - \mathbb{E}_{\overrightarrow{\gamma}_{k+1:T} \sim \mathcal{P}^{T-k}}\left[\sum_{t=1}^{\tau} \lambda_t \cdot g_t(b_t)\right]. \tag{G.2.5}$$

Next, by Proposition G.2.2, we have:

$$\overline{\mathcal{D}}_{\text{combined}}(\overline{\mu}_\tau, \overline{\lambda}_\tau | \mathcal{P}, \widehat{\rho}) + (\rho - \widehat{\rho}) \cdot \overline{\mu}_\tau \geq \frac{1}{T} \mathbb{E}_{\overrightarrow{\gamma} \sim \mathcal{P}^T} \left[\text{Reward}(\text{Opt}, \overrightarrow{\gamma}, \rho) \right]. \tag{G.2.6}$$

We can now repeat the trick in the proof of Theorem 8.5.1:

$$\mathbb{E}_{\overrightarrow{\gamma} \sim \mathcal{P}^T} \left[\text{Reward}(\text{Opt}, \overrightarrow{\gamma}, \rho) \right] \leq \frac{\tau}{T} \cdot \mathbb{E}_{\overrightarrow{\gamma} \sim \mathcal{P}^T} \left[\text{Reward}(\text{Opt}, \overrightarrow{\gamma}, \rho) \right] + (T - \tau), \tag{G.2.7}$$

Then, Inequality G.2.3, Inequality G.2.4, Inequality G.2.5, Inequality G.2.6, and Inequality G.2.7 give

$$\begin{aligned} \mathbb{E}_{\overrightarrow{\gamma} \sim \mathcal{P}^T} \left[\text{Regret}(\textit{Algorithm } 8.5.2, \overrightarrow{\gamma}) \right] \leq \mathbb{E}_{\overrightarrow{\gamma} \sim \mathcal{P}^T} (T - \tau) + \frac{T}{e^{O(T)}} \\ + \mathbb{E}_k \left[\mathbb{E}_{\overrightarrow{\gamma}_{k+1:T} \sim \mathcal{P}^{T-k}} \left[\sum_{t=1}^{\tau} \mu_t \cdot (\widehat{\rho} - p_t(b_t)) \right] \mid k \leq \rho T \right] \\ + \mathbb{E}_k \left[\mathbb{E}_{\overrightarrow{\gamma}_{k+1:T} \sim \mathcal{P}^{T-k}} \left[\sum_{t=1}^{\tau} \lambda_t \cdot g_t(b_t) \right] \mid k \leq \rho T \right] \\ + \mathbb{E}_k \left[\mathbb{E}_{\overrightarrow{\gamma}_{k+1:T} \sim \mathcal{P}^{T-k}} \left[\tau \overline{\mu}_\tau (\rho - \widehat{\rho}) \right] \mid k \leq \rho T \right]. \end{aligned} \tag{G.2.8}$$

By applying Proposition G.2.5 and Proposition 8.4.2, we have

$$\begin{aligned} \mathbb{E}_k \left[\mathbb{E}_{\overrightarrow{\gamma}_{k+1:T} \sim \mathcal{P}^{T-k}} \left[\sum_{t=1}^{\tau} \mu_t \cdot (\widehat{\rho} - p_t(b_t)) \right] \mid k \leq \rho T \right] &\leq \mathbb{E}_{\overrightarrow{\gamma} \sim \mathcal{P}^T} \left[(\tau - T) + K(\overrightarrow{\gamma}) \right] + O(\sqrt{T}) \\ &\quad + \mathbb{E}_k \left[\mathbb{E}_{\overrightarrow{\gamma}_{k+1:T}} (1/\widehat{\rho}) \mid k \leq \rho T \right] \\ &\leq \mathbb{E}_{\overrightarrow{\gamma} \sim \mathcal{P}^T} (\tau - T) + O(\sqrt{T} \log T). \end{aligned} \tag{G.2.9}$$

We invoke Lemma 14 to conclude $\sum_{t=1}^{R} \lambda_t g_t \leq O(\sqrt{R})$ for all $R \leq T$, which lets us conclude

$$\mathbb{E}_{\overrightarrow{\gamma} \sim \mathcal{P}^T} \left[\sum_{t=1}^{\tau} \lambda_t g_t \right] \leq O(\sqrt{T}). \tag{G.2.10}$$

To bound the final term in Inequality G.2.8, we observe that $\rho - \widehat{\rho} = \frac{(1-\rho)K(\overrightarrow{\gamma})}{T - K(\overrightarrow{\gamma})}$ by definition of $\widehat{\rho}$. Combining this with $\tau \leq T$, the bound on $\sum_{i=1}^{\tau} \mu_i$ from Algorithm 8.5.1, the result of Proposition 8.4.2, and the conditional expectation, we get

$$\mathbb{E}_k \left[\mathbb{E}_{\overrightarrow{\gamma}_{k+1:T} \sim \mathcal{P}^{T-k}} \left[\tau \overline{\mu}_\tau (\rho - \widehat{\rho}) \right] \mid k \leq \rho T \right] \leq O(\sqrt{T}). \tag{G.2.11}$$

Combining Inequality G.2.8, Inequality G.2.9, Inequality G.2.10, and Inequality G.2.11 finishes the proof. □