Improved Interior Point Methods for Some Structured Combinatorial Problems

By

Tarun Kathuria

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Prasad Raghavendra, Chair
Professor Nikhil Srivastava
Professor Jelani Nelson

Fall 2023

Improved Interior Point Methods for Some Structured Combinatorial Problems

Abstract

Improved Interior Point Methods for Some Structured Combinatorial Problems

by

Tarun Kathuria

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Prasad Raghavendra, Chair

Over the last few decades, the design of algorithms for combinatorial problems has benefited greatly from a continous optimization perspective. In this dissertation, we will focus on some specific ideas from continous optimization, particularly so-called interior point methods (IPMs), for the design of efficient algorithms for some combinatorial problems in theoretical computer science. We will present algorithms for three different applications using these ideas:

- We first present an improved interior point method algorithm for exact s-t maximum flow on directed graphs that runs in time $O(m^{4/3+o(1)}U^{1/3})$ where U is the ratio of the maximum to minimum capacities on the edges.
- We then design a faster interior point method for the class of semi-definite programs (SDPs), a class of convex programs that have led to polynomial time algorithms for a variety of problems in combinatorial optimization, statistics and machine learning. Our algorithm runs in $O(\sqrt{n}(mn^2 + m^\omega + n^\omega)\log(1/\varepsilon))$ time for an SDP with m constraints and n sized variable matrix, which improves over the previous results in a wide variety of regimes.
- We present a constructive proof of the Spencer problem by designing a Stieltjes transform based barrier function and running a random walk analyzed using this barrier function which gives a signing with discrepancy of $O(\sqrt{n})$ with high probability. We then show that unlike previous such approaches, our approach may generalize to resolve matrix discrepancy problems.

To Dadu, Dadi, Mama and Papa

Contents

Contents **ii**

List of Figures **iii**

List of Tables **iv**

1 Introduction **1**
1.1 Interior Point Methods 3
1.2 Our Results 4

2 Maximum Flow in $O(m^{4/3+o(1)}U^{1/3})$ Time **5**
2.1 Introduction 5
2.2 Preliminaries 8
2.3 Warm up : $\sqrt{m}$ Iteration Algorithm 11
2.4 Improved $m^{4/3+o(1)}U^{1/3}$ Time Algorithm 16
2.5 Conclusion 26

3 A Faster IPM for Semi-definite Programming **27**
3.1 Introduction 27
3.2 Preliminaries 38
3.3 Matrix Multiplication 39
3.4 Main Theorem 45
3.5 Approximate Central Path via Approximate Hessian 45
3.6 Low-rank Update 50
3.7 Runtime Analysis 55
3.8 Comparison with Cutting Plane Method 60
3.9 Initialization 61
3.10 Matrix Multiplication: A Tensor Approach 63

4 An IPM Inspired Approach to Discrepancy **69**
4.1 Yet Another Algorithm for Spencer's Theorem 71

Bibliography **76**

List of Figures

List of Tables

3.1.1 Summary of key SDP algorithms. CPM stands for cutting plane method, and IPM, interior point method. n is the size of the variable matrix, and $m \leq n^2$ is the number of constraints. Runtimes hide $n^{o(1)}$, $m^{o(1)}$ and $\operatorname{poly}\log(1/\epsilon)$ factors, where ϵ is the accuracy parameter. [6] simplifies the proofs in [95, Section 5.5]. Neither [6] nor [95] explicitly analyzed their runtimes, and their runtimes shown here are our best estimates. 29

3.1.2 Total runtimes for the algorithms in Table 3.1.1 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 $\operatorname{poly}\log(1/\epsilon)$ factors, where ϵ is the accuracy parameter and assume ω to equal its currently best known upper bound of 2.373. 30

3.5.1 Summary of parameters in approximate central path for SDP. 46

Acknowledgments

Graduate school most certainly has been significantly more challenging than I could've envisioned and I owe the completion of my Ph.D. to the assistance of many amazing people. I'd first like to thank Prasad Raghavendra for being a patient and cheerful advisor. During graduate school, my grandfather and father passed away and that made it much harder for me to focus on research consistently. Prasad was very understanding and patient during this time period and I am extremely grateful to him for that. I'm also extremely grateful for his encouragement and support when I decided to pursue research areas different than his own. I would also like to thank Nikhil Srivastava for many discussions around problems involving spectral graph theory, random matrix theory and matrix discrepancy. I owe both him and Adam W. Marcus for discussions around free probability and their encouragement for my random ideas about trying to understand free probability and stochastic calculus tools in order to solve problems in Matrix Discrepancy. I'd especially like to thank Ramon van Handel for hosting me at Princeton University in Summer 2023. Ramon's professionalism and his way of approaching research problems, that helped us eliminate a lot of ideas, is really quite spectacular and I'm also grateful for his mentorship towards thinking like a mathematician during this time period. I'd also like to thank Dan Spielman for a very enjoyable summer at Yale University in Summer 2019. Yin Tat Lee deserves many, many thanks for answering many of my questions regarding Interior Point Methods. Watching Yin Tat work motivated me to not shy away from technically involved calculations which ultimately was very beneficial for me. In this same vein, Jelena Diakonikolas also answered many of my questions on acceleration and optimization in general. I thank her for that and for her overall optimism in all our discussions.

I also thank Sam Hopkins for serving as a source of great inspiration and energy. Even though my research interests lie outside sums-of-squares, I appreciate the time he spent trying to explain things to me. I thank Makrand Sinha for many discussions to make me appreciate the beauty of research and learn to enjoy the process more than getting the result. Thanks also to John Wright and Fermi Ma for some informative discussions about grad school in general. I thank Jelani Nelson for funding me for a semester as well as for the myriad of jokes and fun times we had discussing trading with him, Rob, Andrew, Kevin and Lux in the "Market is a Meme Chat" which made the pandemic more tolerable.

A profound thanks goes to Amit Deshpande for taking me on as a Research Fellow at Microsoft Research and for spending a significant amount of time with me explaining a myriad of topics and providing a source of interesting problems when I didn't really know how to proceed in theoretical CS research. I will forever be grateful to him for believing in me and this contrast is rather apparent when you experience others who don't really do so. I also thank many other people at MSR like Prateek Jain, Praneeth Netrapalli, Nagarajan Natarajan, Deeparnab Chakrabarty, Ravishankar Krishnaswamy, Pushmeet Kohli and Sebastien Tavenas who were very helpful in explaining a variety of topics to me in a myriad of topics in theoretical computer science and machine learning. I also thank Prof. S. Sudarshan and Indrajit Bhattacharya for working with me on research projects during my undergraduate studies. The project with Prof. Sudarshan in particular led to my first research project and taught me the important of perseverance in research as well as made me appreciate the value of working on applied problems which can benefit from a theoretical lens.

I am extremely grateful to Anup Rao for offering me an internship at Adobe Research and thank him, Raghav Addanki, Tung Mai, David Arbor, Difan Liu for a fun experience which also made me appreciate industrial research a lot more.

I'd like to thank people I met at Berkeley both in and outside the theory CS group which made time in graduate school a more uplifting experience. I especially thank Elizabeth Yang for being a sounding board to my random rants and discussing K-dramas, Chinmay Nirkhe for fun conversations at Bobby G's and for emphasizing the importance of enjoying the process and having a good work-life balance in order to ultimately do good research. I also thank them as well as Nick Spooner, Arun Ganesh and Rachel Lawrence for fun parties and trivia nights. I also thank Guru Vamsi Policharla, Malvika Raj Joshi, Lucas Gretta and Ansh Nagda for some fun conversations and movie nights. I especially thank Jeff Xu for a really good friendship and for continously motivating me and for also being equally, if not more obsessed, about his research problems as I was to mine and also for some really entertaining conversations unrelated to research, which always managed to lift my spirits. I also thank Shivam Nadimpalli for teaching me the nuances of classical Boolean Harmonic Analysis and for falling prey to my corruption and binge-watching One Piece. The senior graduate students also served as a source of inspiration early on. I thank Aaron Schild and Jonah Sherman for insightful conversations on flow and optimization, Pasin Manurangsi and Fotis Iliopoulos for very illuminating conversations about a variety of theory CS topics and Lynn Chua for some fun random conversations. I also thank Farzam Ebrahimnejad, Christian Ikeokwu, Bhaskar Roberts, Ishaq Aden-Ali, Orr Paradise, Omar Alrabiah, Seri Khoury, Nived Rajaraman, Nikhil Vyas and Mitali Bafna for being good friends with whom I had many illuminating technical and meta-research related discussions.

I also thank Prasad, Luca Trevisan and Satish Rao as well as Gireeja Ranade and Venkat Anantharaman for offering me GSI positions for CS170 and EECS127 respectively, which provided a very rewarding experience. Gireeja is I thank many of the fellow GSIs for these courses which made teaching a much more fun time. Special thanks go to Dee Guo, Arpita Singhal, Emaan Hariri, Yunzhi Zhang, Gillian Chu, Shivani Kishnani, Antares Chen, Jeff Xu, Jialin Li, Lynn Chua and Fotis Ilioupoulos as well as Druv Pai and Kshitij Kulkarni with whom I developed good friendships as well.

I certainly have learned a great deal from many of my collaborators - Swati Padmanabhan, Haotian Jiang, Yeshwanth Cherpanamjeri, Sam Hopkins, Nilesh Tripuraneni, Satyaki Mukherjee, Shivam Nadimpalli, Xin Lyu, Kewen Wu, Sachin Kumar, Reza Gheissari, Sam Hopkins, Zhao Song, Ankit Garg, Jonah Sherman, Yin Tat Lee, Nikhil Srivastava, Avishay Tal, Prasad Raghavendra, Satish Rao, Ramon van Handel and Dan Spielman.

I'd also like to thank Jean Nguyen and Shirley Salanio for being excellent at responding and navigating the many administrative questions and challenges that showed up during graduate school.

I would also like to thank Sachin Kumar, Tanvi Ranjan, Rohan Prinja and Kandarp Khandawala for being steadfast friends from my undergraduate days. Special thanks to Sachin for always listening to my many personal issues and being very supportive and also encouraging me to get into applied ML research.

I am extremely grateful to Aayush Jain for being a great friend. I first met Aayush at MSR and he has continously served as a source of great inspiration on how to do research and willing to talk to me about many random ideas. My main regret has to be that we didn't really talk as much again until Aayush moved to Oakland and talking to him to navigate graduate school early on would've been a great experience. I am extremely grateful for the weekends and travel plans with him and Eleena Mohanty who has also been a great friend and incredibly empathetic to my personal and work-related challenges.

Moving to a completely different country poses it's own set of challenges and there certainly can be no substitute for family in that regard. I'd like to thank Nidhi Mami, Santosh Mama, Aditya and Vikram for being very loving and always inviting me over to Seattle to spend holidays with them which offered much needed reprieve from graduate school stress.

Finally, I'd like to thank my family - Dadu, Dadi, Mama and Papa for raising me and providing me with infinite amounts of love, encouragement and support. I am acutely aware of how blessed I am to have been born in this family and I could never repay all that you have done for me. I especially need to thank my mother for her great many sacrifices she has made for me over my entire lifetime and yet gave me the freedom to make my own decisions. I could never fathom what it took to do so and I will forever be in your debt. Her strength and resilience are something I could never hope to reach but strive for everyday. I dedicate this thesis to all four of you.

Chapter 1

Introduction

Research progress in the design of algorithms for discrete/combinatorial optimization and continous optimization problems followed largely separate trajectories using a very different set of tools and techniques. However, over the last two decades, there has been significant progress in marrying ideas from these two areas leading to significant breakthroughs.

Many of these advancements stem from *relaxing* the combinatorial optimization problem of interest, which can be viewed as an optimization problem over the discrete hypercube, to optimization problems over continous domains which relax the hard combinatorial constraints to lie in some Real domain leading to convex optimization problems. Extremely successful examples of this strategy include relaxation to linear programs and semi-definite programs. This in particular has been an extremely successful program in the form of Sherali-Adams LP hierarchy [106] and the sums-of-squares hierarchy [98, 76] which has led to significant advancements in the design of approximation algorithms for a variety of NP-hard problems including max-cut [50] and sparsest-cut [11], which still remain the state-of-the-art.

Further research into this area has focused on analyzing the semi-definite programs specific to the combinatorial optimization problem and then, instead of using an off-the-shelf convex program solvers, refining specific convex optimization methods like the multiplicative weights update method combined with specific combinatorial primitives to design faster algorithms for these problems [8, 7, 108]. Typically however, these improved dependence on the size of the problem are accomplished by using convex optimization algorithms that depend polynomially in $1/\varepsilon$ where ε is an error parameter. Then problem specific structure is exploited to argue that solving these subproblems to low/constant accuracy actually suffice in recovering a good approximate solution to the original combinatorial problem. This however only works in specific instances and it is in general desirable to ask for improved dependence on the size of the problem which also guarantees a high-accuracy/exact solution, i.e., where the algorithm has a polylogarithmic dependence on $1/\varepsilon$.

Beyond just approximation algorithms for NP-hard problems, the paradigm of viewing combinatorial optimization problems as convex optimization problems has also been an extremely successful strategy for problems which were already known to be solvable in polynomial time by combinatorial methods. A canonical example of such a problem that we will be interested in is the maximum flow problem. The maximum flow problem and its dual, the minimum s-t cut problem, are two of the most well studied problems in combinatorial optimization which are also taught in undergraduate algorithms courses. These problems are key algorithmic primitives used extensively throughout both the theory and practice of computer science. Numerous problems in algorithm design efficiently reduce to the maximum flow problem and techniques developed in the study of this problem have had far reaching implications, for instance, they are a primitive used in getting a nearly linear time $O(\sqrt{\log n}$ approximation algorithm for the aforementioned sparsest cut problem [108, 110].

While [110, 67, 107] gave an algorithm for maximum flow running in time $\widetilde{O}(m/\varepsilon)$ which indeed suffices for many applications like to that of sparsest cut, it is of significant interest whether the maximum flow problem can be solved in $\widetilde{O}(m\log(1/\varepsilon))$ time, which in particular would also give us the maximum flow exactly in $\widetilde{O}(m)$, by rounding the flow using augmenting paths.

Finally, another instance where algorithms for combinatorial problems has benefited greatly from continuous optimization techniques is that of discrepancy theory. The canonical problem in this area corresponds to a universe of n elements, U, and a set system $\mathcal{S}$ comprising of n subsets of this universe. The question is whether we can find a 2-assignment/signing of each of the elements such that the *discrepancy/imbalance* in each set is as small as possible. While random signings give a discrepancy of $O(\sqrt{n\log n})$, Spencer [114] showed that there exists signings with discrepancy $O(\sqrt{n})$, which is tight. Spencer's argument to argue the existence of such signings consists of a rather involved pigeonhole principle and it's not clear how to algorithmically find such a signing outside of enumerating them. Furthermore, Spencer actually conjectured that finding such signings should be NP-hard. However, Bansal [15] showed that one can start a random walk at the origin inside the continuous hypercube $[-1,1]^n$ and keep taking small martingale steps to eventually terminate at a vertex of the hypercube, and hence getting a signing, which satisfies the required discrepancy bound. Since then, nearly all works on algorithms for discrepancy problems have followed a similar template of designing controlled random walks which ensure the target discrepancy bound is never violated as we move from the origin towards a vertex of the hypercube. The design of this control for the random walk however can be seen as somewhat mysterious in many cases and a common guiding principle for the design of the control would be quite desirable, especially in making progress on open questions in discrepancy theory.

While there are certainly many other problems where the interplay between discrete and continuous optimization is quite beneficial, in this dissertation, we will be focused on using interior point methods, another convex optimization algorithm, to design improved algorithms for three problem setups- maximum flow on unit capacity graphs, semidefinite programming and discrepancy theory.

1.1 Interior Point Methods

For convex optimization problems of the form

$$\begin{aligned} &\min c^\top x \\ &\text{s.t. } x \in \mathcal{K} \\ &\quad Ax = b \end{aligned}$$

where $\mathcal{K} \subseteq \mathbb{R}^n$ is a convex cone. Interior point methods relax the convex cone constraint into a barrier function $\phi_{\mathcal{K}}(x)$. While [64] proposed the first interior point method for linear programming with provable complexity bounds on the runtime of the algorithm, [97] proposed a general theory of what properties are desirable for barrier functions to facilitate provable runtime bounds for conic programming. The key property that is desirable for barrier functions for convex cones is that of self-concordance. The notion of self-concordance that we will use can be found in [103].

Definition 1 (Self-Concordant Barriers)**.** *For a convex cone $\mathcal{K}$, we call a C^2 function $\phi_{\mathcal{K}} : \mathcal{K} \to \mathbb{R}$, a self-concordant barrier for $\mathcal{K}$ if for any $x \in \mathcal{K}$, any $y \in \mathcal{K}$ such that $\|y - x\|_x < 1$ and for any $v \in \mathbb{R}^n$, we have*

$$1 - \|y - x\|_x \leq \frac{\|v\|_y}{\|v\|_x} \leq \frac{1}{1 - \|y - x\|_x}$$

where $\|u\|_x^2 := u^\top \nabla^2 \phi(x) u$ for any $u \in \mathbb{R}^n$ and $x \in \mathcal{K}$

In words, this asks for the Hessians of the barrier function to be " spectrally stable" in small Riemannian balls around every point, where the Riemannian metric for the manifold given by $\mathcal{K}$ is just the Hessian of the barrier function.

Interior point methods build a "homotopy path", known as the Central Path, of approximations to the original conic program by either exactly or approximately tracing minimizers of the following path of convex programs parameterized by a non-negative real parameter η:

$$x_\eta^* = \arg \min_{x \in \mathcal{K}, Ax = b} \eta c^\top x + \phi_{\mathcal{K}}(x)$$

With some modifications, it is usually easy to find a solution to x_η^* with η being very close to 0. Then, via some annealing schedule, the value of η is increased (typically multiplicatively) followed by using the (approximate) solution to the previous x_η^* as a starting point to compute the x_η^* for the new value of η, using Newton's method or variants. When η is sufficiently large, x_η^* is close to an optima of the original program. Hence, the main challenge remains in designing a good barrier of $\mathcal{K}$ and an annealing schedule which ensures that we can always approximately trace the path x_η^* in order to ensure that we converge in few iterations. The key parameter that controls the annealing schedule for a barrier is known as the self-concordance parameter ν defined as

$$\nu_{\phi_{\mathcal{K}}} = \sup_{x \in \mathcal{K}} \nabla \phi_{\mathcal{K}}(x)^\top \nabla^2 \phi_{\mathcal{K}}(x) \nabla \phi_{\mathcal{K}}(x)$$

The self-concordance theory of [95] suggests that increasing η multiplicatively by a factor of $1 + \varepsilon/\sqrt{\nu}$ for some sufficiently small constant ε suffices to trace the central path using Newton's method.

1.2 Our Results

In Chapter 2, we will design an interior point method, different than the central path, which allows us to get an improved runtime of $O(m^{4/3}U^{1/3})$ for s-t maximum flow. Formally, we state our main theorem from that chapter.

Theorem 1. *There exists an interior point method for solving s-t maximum flow in direction graphs of $O(m^{4/3+o(1)}U^{1/3})$ time complexity.*

In Chapter 3, we will design an interior point method for Semi-definite Programming (SDP) which utilizes methods from numerical linear algebra and improvements in rectangular matrix multiplication to speed up the computations that show up in each iteration of standard IPMs for SDPs. Formally, our theorem is stated next.

Theorem 2. *There exists an interior point method that solves a general SDP with $n \times n$ sized variable matrix and m constraints in $O(\sqrt{n}(mn^2 + m^\omega + n^\omega))$ time.*

We also remark that barring some major breakthrough, it is unlikely that our runtime can be improved in the interesting parameter regime of $n \leq m \leq n^2$.

Finally, in Chapter 4, we will design a random walk inside the hypercube for problems in discrepancy theory, which is guided by a self-concordant barrier functional for the discrepancy constraints. While such algorithms have been proposed and analysed before, we will suggest that unlike previous approaches, our approach may generalize to solving matrix discrepancy problems.

Theorem 3 (Informal)**.** *There exists a random walk based interior point method guided by a self-concordant barrier potential function for Spencer's theorem in discrepancy theory. The random walk also generalizes to a non-commutative setting that may resolve the Matrix Spencer Problem.*

Chapter 2

Maximum Flow in $O(m^{4/3+o(1)}U^{1/3})$ Time

This chapter is based on work obtained independently by the author as well as by Liu-Sidford and a merger of these results appeared at FOCS 2020 [65]

2.1 Introduction

The *s*-*t* maximum flow problem and its dual, the *s*-*t* minimum cut on graphs are amongst the most fundamental problems in combinatorial optimization with a wide range of applications. Furthermore, they serve as a testbed for new algorithmic concepts which have found uses in other areas of theoretical computer science and optimization. This is because the max-flow and min-cut problems demonstrate the prototypical primal-dual relation in linear programs. In the well-known *s*-*t* maximum flow problem we are given a graph $G = (V, E)$ with m edges and n vertices with edge capacities $u_e \leq U$, and aim to route as much flow as possible from s to t while restricting the magnitude of the flow on each edge to its capacity.

Several decades of work in combinatorial algorithms for this problem led to a large set of results culminating in the work of Goldberg-Rao [52] which gives a running time bound of $O(m \min\{m^{1/2}, n^{2/3}\} \log(\frac{n^2}{m}) \log U)$. This bound remained unimproved for many years. In a breakthrough paper, Christiano et al [37] show how to compute approximate maximum flows in $\widetilde{O}(mn^{1/3} \log(U) \mathsf{poly}(1/\varepsilon))$. Their new approach uses electrical flow computations which are Laplacian linear system solves which can be solved in nearly-linear time [115] to take steps to minimize a softmax approximation of the congestion of edges via a second order approximation. A straightforward analysis leads to a $O(\sqrt{m})$ iteration algorithm. However, they present an insight by trading off against another potential function and show that $O(m^{1/3})$ iterations suffice. This work led to an extensive line of work exploiting Laplacian system solving and continuous optimization techniques for faster max flow algorithms. Lee et al. [79] also present another $O(n^{1/3}\text{poly}(1/\varepsilon))$ iteration algorithm for unit-capacity graphs also using electrical flow primitives. Finally Kelner et al. [67] and Sherman [110, 109] present algorithms achieving $O(m^{o(1)}\text{poly}(1/\varepsilon))$ iteration algorithm for max-flow and its variants, which are based on congestion approximators and oblivious routing schemes as opposed to electrical flow computations. This has now been improved to near linear time [99, 107]. Crucially this line of work can only guarantee weak approximations to max flow due to the $\text{poly}(1/\varepsilon)$ in the iteration complexity.

In order to get highly accurate solutions which depend only polylogarithmically on $1/\varepsilon$, work has relied on second-order optimization techniques which use first and second-order information (the Hessian of the optimization function). To solve the max flow problem to high accuracy, several works have used interior point methods (IPMs) for linear programming [97, 103]. These algorithms approximate non-negativity/ℓ_∞ constraints by approximating them by a *self-concordant* barrier, an approximation to an indicator function of the set which satisfies local smoothness and strong convexity properties and hence can be optimized using Newton's method. In particular, Daitch and Spielman [42] show how to combine standard path-following IPMs and Laplacian linear system solves to obtain $\widetilde{O}(m\sqrt{m}\log(U/\varepsilon))$ iterations, matching Goldberg and Rao up to logarithmic factors. The $O(\sqrt{m})$ iterations is a crucial bottleneck here due to the ℓ_∞ norm being approximated by ℓ_2 norm to a factor of $\sqrt{m}$. Then Lee and Sidford [82] devised a faster IPM using weighted logarithmic barriers to achieve a $\widetilde{O}(m\sqrt{n}\log(U/\varepsilon)$ time algorithm. Madry [90, 89] opened up the weighted barriers based IPM algorithms for max flow to show that instead of ℓ_2 norm governing the progress of each iteration, one can actually make the progress only maintaining bounds on the ℓ_4 norm. Combining this with insights from [37], by using another potential function, which again depends on the energy of the next flow step and carefully tuning the weights in the barriers, he achieved an $\widetilde{O}(m^{3/7})$ iteration algorithm which leads to a $\widetilde{O}(m^{11/7}U^{1/7}\log(m/\varepsilon))$ time. Note that the algorithm depends polynomially on the maximum capacity edge U and hence is mainly an improvement for mildly large edge capacities. This work can also be used to solve min cost flow problems in the same running time [40].

Another line of work beyond IPMs is to solve p-norm regression problems on graphs. Such problems interpolate between electrical flow problems $p = 2$, maximum flow problems $p = \infty$ and transshipment problems $p = 1$. While these problems can also be solved in $O(\sqrt{m})$ iterations to high accuracy using IPMs[97], it was unclear if this iteration complexity could be improved depending on the value of p. Bubeck et al. [30] showed that for any self-concordant barrier for the ℓ_p ball, the iteration complexity has to be at least $O(\sqrt{m})$ thus making progress using IPMs unlikely. They however showed another *homotopy-based* method, of which IPMs are also a part of, can be used to solve the problem in $\widetilde{O}_p(m^{\frac{1}{2}-\frac{1}{p}}\log(1/\varepsilon))$ iterations, where O_p hides dependencies on p in the runtime. This leads to improvements on the runtime for constant values of p. Next, Adil et al. [2], inspired by the work of [30] showed that one can measure the change in p-norm using a second order term based on a different function which allows them to obtain approximations to the p-norm function in different norms with strong condition number. These results can be viewed in the framework of relative convexity [88]. Thus, they can focus on just solving the optimization problem arising from the residual. Using insights from [37], they arrive at a $\widetilde{O}_p(m^{4/3}\log(1/\varepsilon)$-time algorithm. Then follow-up work by Kyng et al. [74] opened up the tools used by Spielman and Teng [115] for ℓ_2-norm flow problems to show that one can construct strong preconditioners for the residual problems for mixed ℓ_2-ℓ_p-norm flow problems, a generalization of ℓ_p-norm flow and obtain an $\widetilde{O}_p(m^{1+o(1)}\log(1/\varepsilon)$ algorithm. These results however do not lead to faster max flow algorithms however due to their large dependence on p.

However, Liu and Sidford [86] improving on Madry [89] showed that instead of carefully tuning the weights based on the electrical energy, one can consider the separate problem of finding a new set of weights under a certain budget constraint to maximize the energy. They showed that a version of this problem reduce to solving ℓ_2-ℓ_p norm flow problems and hence can be solved in almost-linear time using the work of [74, 1]. This leads to a $O(m^{11/8+o(1)}U^{1/4})$-time algorithm for max flow. However, this result still relies on the amount of progress one can take in each iteration being limited to the bounds one can ensure on the ℓ_4 norm of the congestion vector, as opposed to the ideal ℓ_∞ norm. We remark here that there are IPMs for linear programming which only measure centrality in ℓ_∞ norm as opposed to the ℓ_2 or ℓ_4 norm. In particular [39, 84, 27] show how to take a step with respect to a softmax function of the duality gap and trace the central path only maintaining ℓ_∞ norm bounds. [119, 118] also designed potential reduction based IPMs which trace the central path only maintaining centrality in ℓ_∞.

Our Contribution

In this work, we devise a faster interior point method for s-t maximum flow in directed graphs. Precisely, our algorithm runs in time $\widetilde{O}(m^{4/3+o(1)}U)$. Our algorithm builds on top of both Madry [89] and Liu-Sidford [86] and is arguably simpler than both in some regards.

In particular, our algorithm is based on potential reduction algorithms which are a kind of interior point methods for linear programs. These algorithms are based on a potential function which measures both the duality gap as well as accounts for closeness to the boundary via a barrier function. The algorithms differ from path-following IPMs in that they have the potential to not strictly follow the path closely but only trace it loosely, which is also experimentally observed. Usually, the step taken is a scaled gradient step/Newton step on the potential function. Provided that we can guarantee sufficient decrease of the potential function and relate the potential function to closeness to optimality, we can show convergence. We refer to [5, 117, 97] for excellent introductions to potential reduction IPMs.

We will however use a different step; instead of a Newton step, we consider taking the step, subject to augmenting a certain amount of flow in each iteration, which maximizes the decrease in the potential function after taking the step. We then show that this optimization problem can be efficiently solved in $\widetilde{O}(m)$ time using electrical flow computations. While we can show that the potential function decreases by a large amount which guarantees that we can solve the max flow problem in $O(\sqrt{m})$ iterations, we forego writing it in this manner as we are unable to argue such a statement when the weights and hence the potential function is also changed. Instead, we stick to keeping track of the centrality of our flow vector while making sufficient progress. Crucially however, the amount of progress made by our algorithm only depends on bounds on the ℓ_∞ of the congestion vector of the update step rather than the traditional ℓ_2 or ℓ_4 norm bounds in [89, 86]. In order to improve the iteration complexity by obtaining stronger bounds on the ℓ_∞ norm of the congestion vector, we show that like in Liu-Sidford [86], we can change weights on the barrier term for each edge. Instead of using energy as a potential function to be maximized, inspired by oracles designed for multiplicative weights algorithms, we use the change in the potential function itself as the quantity to be maximized subject to a ℓ_1 budget constraint on the change in weights. While we are unaware of how to maximize the ℓ_1 constrained problem, we relax it to an ℓ_q constrained problem, which we solve using a mixed ℓ_2-ℓ_p norm flow problem using the work of [74, 1]. Combining this with an application of 's inequality gives us sufficiently good control on the ℓ_1-norm of the weight change while ensuring that our step has significantly better ℓ_∞ norm bounds on the congestion vector. We believe our potential reduction framework as well as the concept of changing weights based on the update step might be useful in designing faster algorithms for max flow beyond our $m^{4/3}$ running time.

2.2 Preliminaries

Throughout this chapter, we will view graphs as having both forward and backward capacities. Specifically, we will denote by $G = (V, E, u)$, a directed graph with vertex set V of size n, an edge set E of size m, and two non-negative capacities u_e^- and u_e^+ for each edge $e \in E$. For the purpose of this chapter, all edge capacities are bounded by $U = 1$. Each edge $e = (u, v)$ has a head vertex u and a tail vertex v. For a vector $v \in \mathbb{R}^m$, we define $\|v\|_p = (\sum_{i=1}^{m} |v_i|^p)^{1/p}$ and $\|v\|_\infty = \max_{i=1}^{m} |v_i|$ and refer to $\mathrm{Diag}(v) \in \mathbb{R}^{m\times m}$ as the diagonal matrix with the i^{th} diagonal entry equal to v_i.

Maximum Flow Problem Given a graph G, we call any assignment of real values to the edges of E, i.e., $f \in \mathbb{R}^m$, a flow. For a flow vector f, we view f_e as the amount of the flow on edge e and if this value is negative, we interpret it as having a flow of $|f_e|$ flowing in the direction opposite to the edge's orientation. We say that a flow f is an σ-flow, for some demands $\sigma \in \mathbb{R}^n$ if and only if it satisfies *flow conservation constraints* with respect to those demands. That is, we have

$$\sum_{e\in E^+(v)} f_e - \sum_{e\in E^-(v)} f_e = \sigma_v \text{ for every vertex } v \in V$$

where $E^+(v)$ and $E^-(v)$ is the set of edges of G that are entering and leaving vertex v respectively. We will require $\sum_{v\in V} \sigma_v = 0$.

Furthermore, we say that a σ-flow f is feasible in G if and only if f satisfies the capacity constraints

$$-u_e^- \leq f_e \leq u_e^+ \text{ for each edge } e \in E$$

One type of flows that will be of interest to us are $s-t$ flows, where s (the *source*) and t(the *sink*) are two distinguishing vertices of G. Formally, an $s-t$ flow is a σ-flow whose demand vector $\sigma = F\chi_{s,t}$, where F is the value of the flow and $\chi_{s,t}$ is a vector with -1 and $+1$ at the coordinates corresponding to s and t respectively and zero elsewhere.

Now, the maximum flow problem corresponds to the problem in which we are given a directed graph $G = (V, E, u)$ with integer capacities as well as a source vertex s and a sink vertex t and want to find a feasible s-t flow of maximum value. We will denote this maximum value F^*

Residual Graphs A fundamental object in many maximum flow algorithms is the notion of a residual graph. Given a graph G and a feasible flow σ-flow f in that graph, we define the *residual graph* G_f as a graph $G = (V, E, \hat{u}(f))$ over the same vertex and edge set as G and such that, for each edge $e = (u, v)$, it's forward and backward residual capacities are defined as

$$\hat{u}_e^+(f) = u_e^+ - f_e \text{ and } \hat{u}_e^-(f) = u_e^- + f_e$$

We will also denote $\hat{u}_e(f) = \min\{\hat{u}_e^+(f), \hat{u}_e^-(f)\}$. When the value of f is clear from context, we will omit writing it explicitly. Observe that the feasibility of f implies that all residual capacities are always non-negative.

Electrical Flows and Laplacian Systems Let G be a graph and let $r \in \mathbb{R}^m_{++}$ be a vector of edge resistances, where the resistance of edge e is denoted by r_e. For a flow $f \in \mathbb{R}^E$ on G, we define the energy of f to be $\mathcal{E}_r(f) = f^\top R f = \sum_{e\in E} r_e f_e^2$ where $R = \text{Diag}(f)$. For a demand χ, we define the electrical χ-flow f_r to be the χ-flow which minimizes energy $f_r = \arg\min_{B^\top f=\chi} \mathcal{E}_r(f)$, where $B \in \mathbb{R}^{m\times n}$ is the edge-vertex incidence matrix. This flow is unique as the energy is a strictly convex function. The Laplacian of a graph G with resistances r is defined as $L = B^\top R^{-1} B$. The electrical χ flow is given by the formula $f_r = R^{-1} B L^\dagger \chi$. We also define electrical potentials as $\phi = L^\dagger \chi$ There is a long line of work starting from Spielman and Teng which shows how to solve $L\phi = \chi$ in nearly linear time [115, 71, 66, 100, 41, 73, 75].

p-Norm Flows As mentioned above, a line of work [30, 2, 74] shows how to solve more general p-norm flow problems. Precisely, given a "gradient" vector $g \in \mathbb{R}^E$, resistances $r \in \mathbb{R}^E_+$ and a demand vector χ, the problem under consideration is

$$OPT = \min_{B^\top f=\chi} \sum_{e\in E} g_e f_e + r_e f_e^2 + |f_e|^p$$

[74] call such a problem as a mixed ℓ_2-ℓ_p-norm flow problem and denote the expression inside the min as $val(f)$. The main result of the paper is

Theorem 4 (Theorem 1.1 in [74]). *For any even $p \in [\omega(1), o(\log^{2/3-o(1)} n)]$ and an initial solution $f^{(0)}$ such that all parameters are bounded by $2^{\text{poly}(\log(n))}$, we can compute a flow $\widetilde{f}$ satisfying the demands χ such that*

$$val(\widetilde{f}) - OPT \leq \frac{1}{2^{O(\text{poly}(\log m))}}(val(f^{(0)}) - OPT) + \frac{1}{2^{O(\text{poly}(\log m))}}$$

in $2^{O(p^{3/2})}m^{1+O(1/\sqrt{p})}$ time.

We remark that strictly speaking the theorem in [74] states the error to be polynomial but [86] observe that their proof actually implies quasi-polynomial error as stated above. While our subproblems that we need to solve to change weights cannot be exactly put into this form, we show that mild modifications to their techniques can be done to then use their algorithm as a black-box. Hence, we elaborate on their approach below.

One main thing to establish in their paper is how the p-norm changes when we move from f to $f + \delta$.

Lemma 1 (Lemma in [74]). *We have for any $f \in \mathbb{R}^E$ and $\delta \in \mathbb{R}^E$ that*

$$f_i^p + pf_i^{p-1}\delta_i + 2^{-O(p)}h_p(f_i^{p-2}, \delta_i) \leq (f_i + \delta_i)^p \leq f_i^p + pf_i^{p-1}\delta_i + 2^{O(p)}h_p(f_i^{p-2}, \delta_i)$$

where $h_p(x, \delta) = x\delta^2 + \delta^p$

Hence, given an initial solution, it suffices to solve the residual problem of the form

$$\min_{B^\top f = 0} g(f)^\top \delta + \sum_{e \in E} h_p(f_i^{p-2}, \delta_i)$$

where $g(f)_i = pf_i^{p-1}$. Next, they notice that bounding the condition number with respect to the function $h_p(\cdot, \cdot)$ actually suffices to get linear convergence and hence tolerate quasi-polynomially low errors. The rest of the paper goes into designing good preconditioners which allow them to solve the above subproblem quickly.

We will also need some basics about min-max saddle point problems [23]. Given a function $f(x, y)$ such that $\mathsf{dom}(f, x) = \mathcal{X}$ and $\mathsf{dom}(f, y) = \mathcal{Y}$. The problem we will be interested in is of the form

$$\min_{x \in \mathcal{X}} \max_{y \in \mathcal{Y}} f(x, y)$$

Define the functions $f_y(y) = \min_{x \in \mathcal{X}} f(x, y)$ and $f_x(x) = \max_{y \in \mathcal{Y}} f(x, y)$ for every fixed $y \in \mathcal{Y}$. We have the following theorem from Section 2.6 in [23]

Theorem 5. *If $f(x, y)$ is convex in x and concave in y and let $\mathcal{X}, \mathcal{Y}$ be convex and closed. Then f_x is a convex function and f_y is a concave function.*

2.3 Warm up : $\sqrt{m}$ Iteration Algorithm

In this section, we first set up our IPM framework and show how to recover the $\sqrt{m}$ iterations bound for max flow. In the next section, we will then change the weights to obtain our improved runtime. Our framework is largely inspired by [89] and [86] and indeed a lot of the arguments can be reused with some modifications.

IPM Setup

For every edge $e = (u, v)$, we consider assigning two non-negative weights for the forward and backward edges w_e^+ and w_e^-. Based on the weights and the edge capacities, for any feasible flow, we define a barrier functional

$$\phi_w(f) = -\sum_{e \in E} w_e^+ \log(u_e^+ - f_e) + w_e^- \log(u_e^- + f_e)$$

IPMs iterate towards the optimal solution by trading off the amount of progress of the current iterate, i.e., $B^\top f = F\chi$ and the proximity of the point to the constraints measured through the barrier $\phi_w(f)$, known as centrality. Previous IPMs taking a Newton step with respect to the barrier with a size which ensures that we increase the value of the flow F by a certain amount. Due to the fact that a Newton step is the minimization of a second order optimization problem, it can be shown that the step can be computed via electrical flow computations. Typically, taking a Newton step can be decomposed into progress and centering steps where one first takes a progress step which increases the flow value which causes us to lose centrality by some amount. Then one takes a centering step which improves the centrality without increasing the flow value. Depending on the amount of progress we can make in each iteration such that we can still recenter determines the number of iterations our algorithm will take. [89, 86] follow this prototype and loosely speaking the amount of flow value we can increase in each iteration for the progress step depends on the ℓ_∞ norm of the congestion vector, which measures how much flow we can add before we saturate an edge. However, the bottleneck ends up being the centering step which requires that the flow value can only be increased by an amount depending on the ℓ_4 norm of the congestion vector which is a stronger condition than ℓ_∞ norm.

[90, 89] notes that when the ℓ_∞ and ℓ_4 norms of the congestion vector are large then increasing the resistances of the congested edges increases the energy of the *resulting* electrical flow. So he repeatedly increases the weights of the congested edges (called boosting) until the congested vector has sufficiently small norm. By using electrical energy of the resulting step as a global potential function and analyzing how it evolves over the progress, centering and boosting steps, they can control the amount of weight change and number of boosting steps necessary to reduce the norm of the congestion vector. Carefully trading these quantities yields their runtime of $\widetilde{O}(m^{11/7})$. To improve on this, Liu and Sidford [86] consider the problem of finding a set of weight increases which maximize the energy of the resulting flow. As we need to ensure that the weights don't increase by too much, they place a budget

constraint on the weight vector. By showing that a small amount of weight change suffices to obtain good bounds on the congestion vector. Fortunately, this optimization problem ends up being efficiently solvable in almost linear time by using the mixed ℓ_2-ℓ_p norm flow problem of [74]. However, this step still essentially requires ℓ_4-norm bounds to ensure centering is possible.

In this chapter, we will consider taking steps with respect to a potential function. The potential function Φ_w comes from potential reduction IPM schemes and trades off the duality gap with the barrier.

$$\Phi_w(f,s) = m\log\left(1+\frac{f^\top s}{m}\right) + \phi_w(f)$$

For self-concordant barriers like weighted log barriers are, the negative gradient $-\nabla\phi_w(f)$ is feasible for the dual [103] and so for any f' feasible for the primal, we have $f'^\top(-\nabla\phi_w(f)) \geq 0$. We will consider dual "potential" variables $y \in \mathbb{R}^V$. Now, like in [89, 86], we consider a centrality condition

$$y_v - y_u = \frac{w_e^+}{u_e^+ - f_e} - \frac{w_e^-}{u_e^- + f_e} \text{ for all } e = (u,v) \tag{2.1}$$

If (f,y,w) satisfy the above condition, we call it *well-coupled.* Also, given a tuple (f,y,w) and a candidate step $\hat{f}$, define the forward and backward congestion vectors $\rho^+, \rho^- \in \mathbb{R}^E$ as

$$\rho_e^+ = \frac{|\hat{f}_e|}{u_e^+ - f_e} \text{ and } \rho_e^- = \frac{|\hat{f}_e|}{u_e^- + f_e} \text{ for all } e \in E \tag{2.2}$$

We can now assume via binary search that we know the optimal flow value F^* [89]. [89, 86] consider preconditioning the graph which allows them to ensure that for a well-coupled point we can ensure sufficient progress. The preconditioning strategy to ensure this is to add m extra (undirected) edges between s and t of capacity $2U$ each. So the max flow value increases at most by $2mU$. The following lemma can be seen from the proof of Lemma 4.5 in [86]

Theorem 6. *Let (f,y,w) be a well-coupled point for flow value F in a preconditioned graph G. Then we have for every preconditioned edge e that $\hat{u}_e(f) = \min\{u_e^+ - f_e, u_e^- + f_e\} \geq \frac{F^*-F}{7\|w\|_1}$. In particular, if $\|w\|_1 \leq 3m$, then we have $\hat{u}_e(f) \geq \frac{F^*-F}{21m}$. If we also have $F^* - F \geq m^{1/2-\eta}$, then $\hat{u}_e(f) \geq m^{-(1/2+\eta)}/21$*

Now that our setup is complete, we can focus on the step that we will be taking. In this section, we will keep the weights all fixed to 1, i.e., $w_e^+ = w_e^- = 1$ for all $e \in E$. Hence $\|w\|_1 = 2m$. Consider the change in the potential function when we move from f to $f+\hat{f}$ while keeping the dual variable $-\nabla\phi_w(f) = By$ fixed. This change is

$$m\log\left(1 - \frac{(f+\hat{f})^\top \nabla\phi_w(f)}{m}\right) - m\log\left(1 - \frac{f^\top\nabla\phi_w(f)}{m}\right) + \phi_w(f+\hat{f}) - \phi_w(f)$$

We are interested in minimizing this quantity which corresponds to maximizing the decrease in the potential function value while guaranteeing that we send say δ more units of flow $\hat{f}$. Hence the problem is

$$\arg\min_{B^\top \hat{f}=\delta\chi} m\log\left(1-\frac{(f+\hat{f})^\top \nabla\phi_w(f)}{m}\right)+\phi_w(f+\hat{f})$$

Unfortunately, this problem is not convex as the duality gap term is concave in $\hat{f}$. However, we instead can minimize an upper bound to this term which is convex:

$$\begin{aligned}&\min_{B^\top \hat{f}=\delta\chi} \phi_w(f+\hat{f})-(f+\hat{f})^\top\nabla\phi_w(f)\\&=\min_{B^\top \hat{f}=\delta\chi} -\sum_{e\in E} w_e^+\log\left(1-\frac{\hat{f}_e}{u_e^+-f_e}\right)+w_e^-\log\left(1+\frac{\hat{f}_e}{u_e^-+f_e}\right)-\hat{f}_e\left(\frac{w_e^+}{u_e^+-f_e}-\frac{w_e^-}{u_e^-+f_e}\right)\end{aligned}$$

as $\log(1+x)\leq x$ for non-negative x which holds from duality as mentioned above. We will refer to the value of the problem in the last line as the *potential decrement* and will henceforth denote the function inside the minimization as $\Delta\Phi_w(f,\hat{f})$. It is instructive to first see how the coupling condition changes if we were to take the optimal step of the above problem, while remaining feasible. To calculate this, from the optimality conditions of the above program, we can say that there exists a $\hat{y}$ such that for all $e=(u,v)$

$$\begin{aligned}\hat{y}_v-\hat{y}_u&=\left(\frac{w_e^+}{u_e^+-f_e-\hat{f}_e}-\frac{w_e^-}{u_e^-+f_e+\hat{f}_e}\right)-\left(\frac{w_e^+}{u_e^+-f_e}-\frac{w_e^-}{u_e^-+f_e}\right)\\&=\left(\frac{w_e^+}{u_e^+-f_e-\hat{f}_e}-\frac{w_e^-}{u_e^-+f_e+\hat{f}_e}\right)-(y_v-y_u)\end{aligned}$$

Hence, if we update y to $y+\hat{y}$ and f to $f+\hat{f}$, we get a flow of value $F+\delta$ such that the coupling condition with respect to the new y and f still hold.

Hence, we can now focus on actually computing the step and showing what δ we can take to ensure that we still satisfy feasibility, i.e., bounds on the ℓ_∞ norm of the congestion vector. The function we are trying to minimize comprises of a self-concordant barrier term and a linear term. Unfortunately, we cannot control the condition number of such a function to optimize it in efficiently over the entire space as this is arguably as hard as the original problem itself. However, due to self-concordance, the function behaves smoothly enough (good condition number) in a box around the origin but that seemingly doesn't help us solve the problem over the entire space. Fortunately, a fix for this was already found in [30]. In particular they (smoothly) extend the function quadratically outside a box to ensure that the (global) smoothness and strong convexity properties inside the box carries over to that outside the box as well while still arguing that the minimizer is the same provided the minimizer of the original problem was inside the box. Specifically, the following lemma can be inferred from Section 2.2 of [30].

Lemma 2. *Given a function $f(x)$ which is L-smooth and μ-strongly convex inside an interval $[-\ell, \ell]$. Then, we define the quadratic extension of f, defined as*

$$f_\ell(x) = \begin{cases} f(x), & \text{for } -\ell \leq x \leq \ell \\ f(-\ell) + f'(-\ell)(x+\ell) + \frac{1}{2}f''(-\ell)(x+\ell)^2, & \text{for } x < -\ell \\ f(\ell) + f'(\ell)(x-\ell) + \frac{1}{2}f''(\ell)(x-\ell)^2, & \text{for } x > \ell \end{cases}$$

The function f_ℓ is C^2, L-smooth and μ-strongly convex. Furthermore, for any convex function $\psi(x)$ provided $x^* = \arg\min_{x\in\mathcal{X}} \psi(x) + \sum_{i=1}^{n} f(x_i)$ lies inside $\prod_{i=1}^{n}[-\ell_i, \ell_i]$, then $\arg\min_{x\in\mathcal{X}} \psi(x) + \sum_{i=1}^{n} f_{\ell_i}(x_i) = x^*$

Hence, it suffices to consider a δ small enough such that the minimizer is the same as for the original problem and we can focus on minimizing this quadratic extension of the function. For minimization, we can use Accelerated Gradient Descent or Newton's method.

Theorem 7 ([93])**.** *Given a convex function f which satisfies $D \preceq \nabla^2 f(x) \preceq \kappa D \forall x \in R^n$ with some given fixed diagonal matrix D and some fixed κ. Given an initial point x_0 and an error parameter $0 < \varepsilon < 1/2$, the accelerated gradient descent (AGD) outputs x such that*

$$f(x) - \min_x f(x) \leq \varepsilon(f(x_0) - \min_x f(x))$$

in $O(\sqrt{\kappa}\log(\kappa/\varepsilon))$ iterations. Each iteration involves computing ∇f at some point x and projecting the function onto the subspace defined by the constraints and some linear-time calculations.

Notice that the Hessian of the function in the potential decrement problem is a diagonal matrix with the e^{th} entry being

$$\frac{w_e^+}{(u_e^+ - f_e - \hat{f}_e)^2} + \frac{w_e^-}{(u_e^- + f_e + \hat{f}_e)^2}$$

So provided ρ_e^+, ρ_e^- are less than some small constant, the condition number κ of the Hessian is constant with respect to the diagonal matrix which is $\nabla^2\phi_w(f)$ and hence we can use Theorem 7 to solve it in $\widetilde{O}(1)$ to quasi-polynomially good error. Furthermore notice that the algorithm is just computing a gradient and then doing projection and so can be computing using a Laplacian linear system solve and hence runs in nearly linear time. Furthermore, quasi-polynomially small error will suffice for our purposes [90, 89, 86].

Now, we just need to ensure that we can control the ℓ_∞-norm of the congestion vector, as that controls how much flow we can still send without violating constraints. Note further, that we need to set ℓ while solving the quadratic extension of the potential decrement problem so that it's greater than the ℓ_∞ norm that we can guarantee. We will want both of these to be some constants.

As mentioned above, the point of preconditoning the graph is to ensure that the preconditioned edges themselves can facilitate sufficient progress. To bound the congestion, we show an analog of Lemma 3.9 in [89].

Lemma 3. *Let (f, y, w) be a well-coupled solution with value F and let $\delta = \frac{F^*-F}{1000\sqrt{m}}$. Let $\hat{f}$ be the solution to the potential decrement problem. Then we have, $\rho_e^+, \rho_e^- \leq 0.1$ for all edges e.*

Proof. Consider a flow f' which sends $\frac{2\delta}{m}$ units of flow on each of the $m/2$ preconditioned edges. Certainly the potential decrement flow $\hat{f}$ will have smaller potential decrement than that of f' which is

$$\begin{aligned}
&\Delta\Phi_w(f, f') \\
&= -\sum_{e\in E} w_e^+ \log\left(1 - \frac{f'_e}{u_e^+ - f_e}\right) + w_e^- \log\left(1 + \frac{f'_e}{u_e^+ - f_e}\right) - f'_e\left(\frac{w_e^+}{u_e^+ - f_e} - \frac{w_e^-}{u_e^- + f_e}\right) \\
&\leq \sum_{e\in E} w_e^+ \left(\frac{f'_e}{\hat{u}_e^+(f)}\right)^2 + w_e^- \left(\frac{f'_e}{\hat{u}_e^-(f)}\right)^2 \\
&\leq \|w\|_1 \left(\frac{42\delta}{F^* - F}\right)^2 \\
&< \frac{0.002\|w\|_1}{m} \leq 0.004
\end{aligned}$$

where the second inequality follows from $-\log(1-x) \leq x + x^2$ and $-\log(1+x) \leq -x + x^2$ for non-negative x and the third inequality follows from plugging in the value of the flow on the preconditioned edges and using Lemma 6. Finally we use $\|w\|_1 = 2m$. Now it suffices to prove a lower bound on the potential decrement in terms of the congestion vector. For this, we start by considering the inner product of $\hat{f}$ with the gradient of the $\Delta\Phi_w(f, \hat{f})$

$$\begin{aligned}
&\sum_{e\in E} \hat{f}_e \left(\frac{w_e^+}{u_e^+ - f_e - \hat{f}_e} - \frac{w_e^-}{u_e^- + f_e + \hat{f}_e} - \frac{w_e^+}{u_e^+ - f_e} + \frac{w_e^-}{u_e^- + f_e}\right) \\
&= \sum_{e\in E} \left(\frac{w_e^+ \hat{f}_e^2}{(\hat{u}_e^+ - \hat{f}_e)\hat{u}_e^+} + \frac{w_e^- \hat{f}_e^2}{(\hat{u}_e^- + \hat{f}_e)\hat{u}_e^-}\right) \\
&\leq \sum_{e\in E} 1.1 \left(\frac{w_e^+ \hat{f}_e^2}{(\hat{u}_e^+)^2} + \frac{w_e^- \hat{f}_e^2}{(\hat{u}_e^-)^2}\right)
\end{aligned}$$

$$\begin{aligned}
&\leq 1.1 \sum_{e\in E} \left(\frac{w_e^+ \hat{f}_e^2}{(\hat{u}_e^+)^2} + \frac{w_e^- \hat{f}_e^2}{(\hat{u}_e^-)^2} \right) \\
&\leq 2.2 \sum_{e\in E} -w_e^+ \log\left(1 - \frac{f'_e}{\hat{u}_e^+}\right) - w_e^- \log\left(1 + \frac{f'_e}{\hat{u}_e^+}\right) - f'_e \left(\frac{w_e^+}{\hat{u}_e^+} - \frac{w_e^-}{\hat{u}_e^-} \right) \\
&= 2.2 \Delta\Phi_w(f, \hat{f}) \\
&\leq 0.0088
\end{aligned}$$

where the second-to-last inequality follows from $x + x^2/2 \leq -\log(1-x)$ and $-x + x^2/2 \leq -\log(1+x)$. Strictly speaking, the first inequality only holds for $\hat{f}_e \leq \hat{u}_e(f)/10$. However, instead of considering the inner product of $\hat{f}$ with the gradient of $\Delta\Phi_w(f, \hat{f})$, we will instead consider it's quadratic extension with $\ell_e = \hat{u}_e(f)/10$ for each edge e. It is easy to see that if $\hat{f}$ is outside the box, then also the desired inequality still holds (by computing the value the quadratic extension takes on f' in the cases outside the box). To finish the proof,

$$\begin{aligned}
&\sum_{e\in E} \hat{f}_e \left(\frac{w_e^+}{u_e^+ - f_e - \hat{f}_e} - \frac{w_e^-}{u_e^- + f_e + \hat{f}_e} - \frac{w_e^+}{u_e^+ - f_e} + \frac{w_e^-}{u_e^- + f_e} \right) \\
&= \sum_{e\in E} \left(\frac{w_e^+ \hat{f}_e^2}{(\hat{u}_e^+ - \hat{f}_e)\hat{u}_e^+} + \frac{w_e^- \hat{f}_e^2}{(\hat{u}_e^- + \hat{f}_e)\hat{u}_e^-} \right) \\
&\geq 9/10 \sum_{e\in E} \left(\frac{w_e^+ \hat{f}_e^2}{(\hat{u}_e^+)^2} + \frac{w_e^- \hat{f}_e^2}{(\hat{u}_e^-)^2} \right) \\
&\geq 0.9 \|\rho\|_\infty^2
\end{aligned}$$

Hence, combining the above, we get that $\|\rho\|_\infty \leq 0.1$ □

Notice that since $\|\rho\|_\infty < 0.1$, the minimizer of the quadratic smoothened function is the same as the function without smoothing and hence the new step is well-coupled as per the argument above. Hence, in every iteration, we decrease the amount of flow that we could send multiplicatively by a factor of $1 - 1/\sqrt{m}$ and hence in $\sqrt{m}$ iterations we will get to a sufficiently small amount of remaining flow that we can round using one iteration of augmenting paths. This completes our $\sqrt{m}$ iteration algorithm.

2.4 Improved $m^{4/3+o(1)}U^{1/3}$ Time Algorithm

In this section, we show how to change weights to improve the number of iterations in our algorithm. We will follow the framework of Liu and Sidford [86] of finding a set of weights to add under a norm constraint such that the step one would take with respect to the new set of weights maximizes a potential function. In their case, since the step they are taking is an electrical flow, the potential function considered is the energy of such a flow. As our step is different, we will instead take the potential decrement as the potential function with respect to the new set of weights. Perhaps surprisingly however, we can make almost all their arguments go through with minor modifications. Let the initial weights be w and say we would like to add a set of weights w'. Then we are interested in maximizing the potential decrement with respect to the new set of weights. This can be seen as similar to designing oracles for multiplicative weight algorithms for two-player games where a player plays a move to penalize the other player the most given their current move. Our algorithm first finds a finds a new set of weights and then takes the potential decrement step with respect to the new weights. Finally, for better control of the congestion vector, we show that one can decrease some of the weight increase like in [86]. We first focus on the problem of finding the new set of weights. We are going to introduce a set $r' \in \mathbb{R}^E_{++}$ of "resistances" and will optimize these resistances and then obtain the weights from them. Let w be the current set of weights and w' be the set of desired changes. Without loss of generality, assume that $\hat{u}_e(f) = \hat{u}^+_e(f)$ and now given a resistance vector r', we define the weight changes as

$$(w^+_e)' = r'_e(\hat{u}^+_e(f))^2 \text{ and } (w^-_e)' = \frac{(w^+_e)'\hat{u}^-_e(f)}{\hat{u}^+_e(f)}$$

This is the same set of weight changes done in [86] in the context of energy maximization. This set of weights ensures that our point (f, y, w) is well-coupled with respect to $w + w'$ as well, i.e.,

$$\frac{(w^+_e)'}{\hat{u}^+_e(f)} = \frac{(w^-_e)'}{\hat{u}^-_e(f)}$$

The problem we would now like to solve is

$$g(W) = \max_{r'>0, \|r'\|_1 \le W} \min_{B^\top \hat{f} = \delta\chi} \Delta\Phi_{w+w'}(f, \hat{f})$$

Here w' is based on r' in the form written above. While this is the optimization problem we would like to solve, we are unable to do so due to the ℓ_1 norm constraint on the resistances. We will however be able to solve a relaxed q-norm version of the problem.

$$\begin{aligned} g_q(W) &= \max_{r'>0 \|r'\|_q \le W} \min_{B^\top \hat{f} = \delta\chi} \Delta\Phi_{w+w'}(f, \hat{f}) \\ &= \max_{r'>0, \|r'\|_q \le W} \min_{B^\top \hat{f} = \delta\chi} \Delta\Phi_w(f, \hat{f}) \qquad -\sum_{e \in E} (w^+_e)' \left(\log\left(1 - \frac{f'_e}{u^+ - f_e}\right) + \frac{f'_e}{u^+_e - f_e}\right) \\ &\qquad\qquad + (w^-_e)' \left(\log\left(1 + \frac{f'_e}{u^+ - f_e}\right) + \frac{(f^-_e)'}{u^-_e + f_e}\right) \end{aligned}$$

Notice that this is a linear (and hence concave) function in w' and hence in r' and is closed and convex in $\hat{f}$ and the constraints are convex as they are only linear and norm ball constraints.

Hence, using Theorem 5, we can say that

$$\min_{B^\top \hat{f}=\delta\chi} \Delta\Phi_w(f,\hat{f}) - \sum_{e\in E}(w_e^+)' \log\left(1-\frac{f_e'}{u^+ - f_e}\right) + (w_e^-)' \log\left(1+\frac{f_e'}{u^+ - f_e}\right) - f_e'\left(\frac{(w_e^+)'}{u_e^+ - f_e} - \frac{(w_e^-)'}{u_e^- + f_e}\right)$$

is concave in r' and

$$\max_{r'>0,\|r'\|_q\le W} \Delta\Phi_w(f,\hat{f}) - \sum_{e\in E}(w_e^+)' \log\left(1-\frac{f_e'}{u^+ - f_e}\right) + (w_e^-)' \log\left(1+\frac{f_e'}{u^+ - f_e}\right) - f_e'\left(\frac{(w_e^+)'}{u_e^+ - f_e} - \frac{(w_e^-)'}{u_e^- + f_e}\right)$$

is convex in $\hat{f}$. Now, as in [86], we use Sion's minimax lemma to get

$$\min_{B^\top \hat{f}=\delta\chi} \Delta\Phi_w(f,\hat{f}) + \max_{r'>0,\|r'\|_q\le W} - \sum_{e\in E}(w_e^+)' \log\left(1-\frac{\hat{f}_e}{u^+ - f_e}\right) + (w_e^-)' \log\left(1+\frac{\hat{f}_e}{u^+ - f_e}\right) - \hat{f}_e\left(\frac{(w_e^+)'}{u_e^+ - f_e} - \frac{(w_e^-)'}{u_e^- + f_e}\right)$$

$$\min_{B^\top \hat{f}=\delta\chi} \Delta\Phi_w(f,\hat{f}) + W\left[\sum_{e\in E} g_e(\hat{f})^p\right]^{1/p} \tag{2.3}$$

where $g_e(\hat{f}) = (\hat{u}_e^+(f))^2 \log\left(1-\frac{\hat{f}_e}{\hat{u}^+}\right) + \hat{u}_e^+(f)\hat{u}_e^-(f)\log\left(1+\frac{\hat{f}_e}{\hat{u}_e^-(f)}\right)$ and we plugged in the value of w' in terms of r' and used that $\max_{\|x\|_q\le W} y^\top x = W\|y\|_p$ with $1/p+1/q=1$. As mentioned above, the function inside the minimization problem is convex. Furthermore, from the proof of Theorem 5, it can be inferred that any smoothness and strong convexity properties that the function inside the min-max had carries over on the function after the maximization. Hence as in Section 2.3, we will consider the quadratic extension of the function (as a function of f for the function inside the min-max with $\ell_e = \hat{u}_e(f)/10$. This is just the quadratic extension of $\Delta\Phi_w(f,\hat{f})$ and the quadratic extension of $g_e(f)$. Now, the strategy will be to consider adding flow using this step while the remaining flow to be routed $F^* - F \ge m^{1/2-\eta}$. After which, running $m^{1/2-\eta}$ iterations of augmenting paths gets us to the optimal solution. We will need to ensure that that throughout the course of the algorithm the ℓ_1 norm of the weights doesn't get too large. For doing that, we will first compute the weight changes and then do a weight reduction procedure [86] in order to always ensure that $\|w\|_1 \le 3m$.

We will take $\eta = 1/6 - o(1) - \frac{1}{3}\log_m(U)$ and $W = m^{6\eta}$. Provided we can ensure that the $\|w\|_1 \le 3m$ throughout the course of the algorithm, that the ℓ_∞ of the congestion vector is always bounded by a constant and that we can solve the resulting step in almost-linear time, we will obtain an algorithm which runs in time $m^{4/3+o(1)}U^{1/3}$ time.

Theorem 8. *There exists an algorithm for solving $s-t$ maximum flow in directed graphs in time $m^{4/3+o(1)}U^{1/3}$ time.*

To summarize, our algorithm starts off with $(f, y) = (0, 0)$ and $w_e^+ = w_e^- = 1$ for all edges e. Then in each iteration, starting with a well-coupled (f, y, w) with flow value F and $\delta = (F^* - F)/m^{1/2-\eta}$ and $W = m^{6\eta}$ we then solve Equation 2.3 (which is the potential decrement problem with the new weights) problem to obtain $\hat{f}$ which will be the step we will take (and has flow value $F + \delta$ and then all that remains is to actually find the update weights w' which will have a closed form expression in terms of $\hat{f}$ and then we perform a weight reduction step to obtain the new w' which ensures that we still remain well-coupled for $\hat{f}$ and repeat while $F^* - F \geq m^{1/2-\eta}$. Finally, we round the remaining flow using $m^{1/2-\eta}$ iterations of augmenting paths. We first state the lemma the proof of which is similar to Lemma 3

Lemma 4. *Let (f, y, w) be a well-coupled solution with value F and let $\delta = \frac{F^*-F}{5000m^{1/2-\eta}}$. Let $\hat{f}$ be the solution to the potential decrement problem considered in Equation 2.3. Then, we have for all edges e that $\rho_e^+, \rho_e^- \leq 0.1$ and $|\hat{f}_e| \leq 9m^{-2\eta}$*

Proof. We follow the strategy used in the proof of Lemma 3. Recall that the problem we are trying to understand is

$$\min_{B^\top \hat{f} = \delta\chi} \Delta\Phi_w(f, \hat{f}) + W \left[\sum_{e \in E} g_e(\hat{f})^p\right]^{1/p}$$

where $g_e(\hat{f}) = (\hat{u}_e^+(f))^2 \log\left(1 - \frac{\hat{f}_e(f)}{\hat{u}_e^+(f)}\right) + \hat{u}_e^+(f)\hat{u}_e^-(f) \log\left(1 + \frac{\hat{f}_e}{\hat{u}_e^-(f)}\right)$. As in Lemma 3, we will consider a flow f' which sends $\frac{2\delta}{m}$ units of flow on each of the $m/2$ preconditioned edges. Certainly, the objective value of the above function at $\hat{f}$ will have a smaller value than that at f'. For the first term $\Delta\Phi_w(f, \hat{f})$, running the same argument as in Lemma 3, we get that

$$\begin{aligned}\Delta\Phi_w(f, \hat{f}) &\leq \|w\|_1 \left(\frac{42\delta}{F^* - F}\right)^2 \\ &\leq 0.000071m^{2\eta}\end{aligned}$$

For the the second term, we use $\log(1 - x) \leq -x + x^2$ and $\log(1 + x) \leq x + x^2$, to get that $g_e(f) \leq \hat{f}_e^2 \left(1 + \frac{\hat{u}_e^+(f)}{\hat{u}_e^+(f)}\right) \leq 2\hat{f}_e^2$ where we have used that $\hat{u}_e^+(f) \leq \hat{u}_e^-(f)$. Now, since there is non-zero flow on the preconditioned edges, we get that

$$\begin{aligned}W \left[\sum_{e \in E} g_e(f')^p\right]^{1/p} &\leq 2W(\delta/m)^2 m^{o(1)} \\ &\leq 2m^{6\eta+o(1)} \left(\frac{F^* - F}{5000m(m^{1/2-\eta})}\right)^2 \\ &\leq 0.0000004m^{8\eta-1+o(1)}U^2\end{aligned}$$

using $p = \sqrt{\log n}$, the fact that $F^* - F \leq mU$ and the value of $\delta = \frac{F^*-F}{5000m^{1/2-\eta}}$. Also using the value of η, we can see that this term is less than $0.0000001m^{2\eta}$. Hence, combining the two, we get that the objective value at $\hat{f}$ is less than $0.000072m^{2\eta}$. As the objective function is made up of two non-negative quantities, we can obtain two inequalities using this upper bound by dropping one term from the objective value each time. For the second part, we ignore the first term of the objective function and lower bound the second term using the fact that $\log(1+x) \geq x + x^2/2$ and $\log(1-x) \geq -x + x^2/2$

$$\begin{aligned} 0.000072m^{2\eta} &\geq W\left[\sum_{e\in E} g_e(\hat{f})^p\right]^{1/p} \geq W|g_e(\hat{f})| \\ &\geq W\hat{f}_e^2(1 + \hat{u}_e^+(f)/\hat{u}_e(f)) \\ &\geq W\hat{f}_e^2 \end{aligned}$$

This gives us that $|\hat{f}_e| \leq 0.009m^{-2\eta}$ by plugging in the value of $W = m^{6\eta}$

For the first part now, assume for the sake of contradiction that $\rho_e > 0.1$, otherwise we are done. Now, dropping the second term we want to establish that $\frac{1}{\hat{u}_e(f)} \leq 9m^{2\eta}$, which we will do so by a proof similar to the proof of Lemma 4.3 in [89]. Now using the argument as in Lemma 3, we get for an edge $e = (u, v)$,

$$\begin{aligned}
0.000072m^{2\eta} &\geq \Delta\Phi_w(f,\hat{f}) \\
&\geq \frac{1}{2.2}\sum_{e\in E}\hat{f}_e\left(\frac{w_e^+}{u_e^+ - f_e - \hat{f}_e} - \frac{w_e^-}{u_e^- + f_e + \hat{f}_e} - \frac{w_e^+}{u_e^+ - f_e} + \frac{w_e^-}{u_e^- + f_e}\right) \\
&= \frac{1}{2.2}\hat{f}^\top B\hat{y} \\
&= \frac{1}{2.2}\delta\chi^\top\hat{y} \\
&= \frac{F^* - F}{11000m^{1/2-\eta}}\chi^\top\hat{y} \\
&\geq \chi^\top\hat{y}/11000 \\
&= (\hat{y}_s - \hat{y}_t)/11000 \\
&\geq (\hat{y}_u - \hat{y}_v)/11000 \\
&= \frac{1}{11000}\left(\frac{w_e^+}{u_e^+ - f_e - \hat{f}_e} - \frac{w_e^-}{u_e^- + f_e + \hat{f}_e} - \frac{w_e^+}{u_e^+ - f_e} + \frac{w_e^-}{u_e^- + f_e}\right) \\
&\geq \frac{9\hat{f}_e}{110000}\left(\frac{w_e^+}{(u_e^+ - f_e)^2} + \frac{w_e^-}{(u_e^- + f_e)^2}\right) \\
&\geq \frac{9\rho_e}{110000\hat{u}_e(f)} \\
&\geq \frac{0.9}{110000\hat{u}_e(f)}
\end{aligned}$$

where the first and second equalities follows from optimality and feasibility conditions of the potential decrement problem respectively and the third inequality follows from the condition that we run the program while the flow left to augment is at least $m^{1/2-\eta}$. This implies that $1/\hat{u}_e(f) \leq 9m^{2\eta}$. Multiplying this with $|\hat{f}_e| \leq 0.009m^{-2\eta}$, we get that $\rho_e \leq 0.1$, which finishes the proof. We also need to argue the inequality $\hat{y}_s - \hat{y}_t \geq \hat{y}_u - \hat{y}_v$. The optimality conditions of

$$\hat{y}_u - \hat{y}_v = \hat{f}_e\left(\frac{w_e^+}{(u_e^+ - f_e - \hat{f}_e)(u_e - f_e)} + \frac{w_e^-}{(u_e^- + f_e)(u_e^- + f_e + \hat{f}_e)}\right)$$

and noticing that the quantity in brackets in the right hand side above is non-negative, tells us that there is a fall in potential along the flow. This along with noticing that the sum of the potential difference in a directed cycle is zero, tells us that the graph induced by just the flow $\hat{f}$ is a DAG. Since, it's a DAG, it can be decomposed into disjoint $s-t$ paths along which flow is sent and every edge belongs to one of these paths. Hence, the potential difference across an edge is less than the potential difference across the whole path which is the potential difference between s and t and hence, we are done.

As before, all these arguments go through with the quadratically smoothened cases cases rather than the original function to still get the same bounds and since $\rho_e \leq 0.1$, the minimizers of the two are the same which completes the proof. □

Next notice that (f, y) are still a well-coupled solution with respect to the new weights $w + w'$ as the weights were chosen to ensure that the coupling condition is unchanged.

Lemma 5. *Our new weights, after weight reduction, satisfy $\|w'\|_1 \leq m^{4\eta+o(1)}U \leq m/2$ and $(f + \hat{f}, y + \hat{y})$ is well-coupled with respect to $w + w'$*

Proof. Using optimality conditions of the program in Equation 2.3, we see that there exists a $\hat{y}$ such that

$$\hat{y}_v - \hat{y}_u = \hat{f}_e \left(\frac{w_e^+}{(\hat{u}_e^+ - \hat{f}_e)\hat{u}_e^+} - \frac{w_e^-}{(\hat{u}_e^- + \hat{f}_e)\hat{u}_e^-} \right) + W \hat{f}_e \frac{g_e^{p-1}}{\|g\|_p^{p-1}} \left(\frac{\hat{u}_e^+}{\hat{u}_e^+ - \hat{f}_e} - \frac{\hat{u}_e^-}{\hat{u}_e^- + \hat{f}_e} \right)$$

where $g \in \mathbb{R}^E$ is the vector formed by taking $g_e(\hat{f})$ for the e^{th} coordinate. We will take

$$(r_e)' = W \frac{g_e^{p-1}}{\|g\|_p^{p-1}} \text{ and } (w_e^+)' = W \frac{g_e^{p-1}}{\|g\|_p^{p-1}} (\hat{u}_e^+)^2 \text{ and } (w_e^-)' = W \frac{g_e^{p-1}}{\|g\|_p^{p-1}} (\hat{u}_e^+ \hat{u}_e^-)$$

which satisfies the well-coupling condition we want to ensure. Also notice that $\|r\|_q = W$ so we satisfy the norm ball condition as well. Now, we need to upper bound the ℓ_1 norm of w'. We will take $p = \sqrt{\log m}$

$$\begin{aligned} \|w'\|_1 &\leq m^{1/p} \|w'\|_q \\ &\leq m^{o(1)} \left(\sum_{e \in E} (w_e^+)' + (w_e^-)' \right)^{1/q} \\ &\leq 2m^{o(1)} W U^2 = O(m^{6\eta + o(1)} U^2) \end{aligned}$$

as $\hat{u}_e^+, \hat{u}_e^- \leq U$. Plugging in the value of η, we get that this is less than $m/2$. Now, we will perform weight reductions to obtain a new set of weights w'' such that they still ensure the coupling condition doesn't change and we can establish better control on the weights. The weight reduction is procedure is the same as that in [86] where we find the smallest non-negative w'' such that for all edges

$$\frac{(w_e^+)'}{\hat{u}_e^+ - \hat{f}_e} - \frac{(w_e^-)'}{\hat{u}_e^- + \hat{f}_e} = \frac{(w_e^+)''}{\hat{u}_e^+ - \hat{f}_e} - \frac{(w_e^-)''}{\hat{u}_e^- + \hat{f}_e}$$

Notice that we also have that $\frac{(w_e^+)'}{\hat{u}_e^+} = \frac{(w_e^-)'}{\hat{u}_e^-}$ and

$$\frac{\hat{u}_e^+ - \hat{f}_e}{\hat{u}_e^- + \hat{f}_e} = (1 \pm O(\max\{\rho_e^+, \rho_e^-\})) \frac{\hat{u}_e^+}{\hat{u}_e^-}$$

Hence, it follows that

$$(w_e^+)'' + (w_e^-)'' \leq O(\max\{\rho_e^+, \rho_e^-\})((w_e^+)' + (w_e^-)')$$

As $|\hat{f}_e| \leq 9m^{-2\eta}$ from Lemma 4, we get

$$\begin{aligned}\|w''\|_1 &\leq m^{-2\eta} \sum_{e \in E} \frac{W g_e^{p-1}}{\|g\|_p^{p-1}} (\hat{u}_e^+ + \hat{u}_e^-) \\ &\leq O(m^{4\eta+o(1)}U) \leq m/2\end{aligned}$$

As before, while this argument is done for the non-quadratically extended function while we are optimizing the quadartically extended function, as our $\rho_e^+, \rho_e^- \leq 0.1$, the minimizers are the same and hence the above argument works. □

Now, provided that we can show how to solve Equation 2.3 in almost-linear time, we are done. This is because we run the algorithm for $m^{1/2-\eta}$ iterations and the ℓ_1 norm of the weights increases by at most $m^{4\eta+o(1)}U$ in each iteration. Hence the final weights are $\|w\|_1 \leq 2m + m^{1/2+3\eta+o(1)}U \leq 5m/2$. So we can use Lemma 6 throughout the course of our algorithm. Also, as mentioned above, notice that the flow $\hat{f}$ that we augment in every iteration is just the solution to the potential decrement problem with the new weights. Hence, from the argument in Section 2.3, we always maintain the well-coupled condition.

To show that we can solve the problem in Equation 2.3, we will appeal to the work of [74]. As mentioned above, their work establishes Lemma 1 and then shows that for any function which can be sandwiched in that form plus a quadratic term which is the same on both sides, one can just minimize the resulting upper bound to get a solution to the optimization problem with quasi-polynomially low error. Hence, we will focus on showing that the objective function in our problem can also be sandwiched into terms of this form after which appealing to their algorithm, we will get a high accuracy solution to our problem in almost linear time. The first issue that arises is that strictly speaking, their algorithm only works for minimizing objectives of the form

$$OPT = \min_{B^\top f = \chi} \sum_{e \in E} g_e f_e + r_e f_e^2 + |f_e|^p$$

whereas for our objective, the p-norm part is not raised to the power p but is just the p-norm itself. The solution for this however was already given in Liu-Sidford [86] where they show (Lemma B.3 in their paper) that for sufficiently nice functions minimizing problems of the form $\min f(x) + h(g(x))$ can be obtained to high accuracy if we can obtain minimizers to functions of the form $f(x) + g(x)$. The conditions they require on the functions are also satisfied for our functions and is a straightforward calculation following the proof in their paper [86]. Hence, we can focus on just showing how to solve the following problem whose value we denote by OPT

$$\min_{B^\top \hat{f} = \chi} \sum_{e \in E} - \left(w_e^+ \log_{0.1} \left(1 - \frac{\hat{f}_e}{\hat{u}_e^+}\right) + w_e^- \log_{0.1} \left(1 + \frac{\hat{f}_e}{\hat{u}_e^-}\right) + \hat{f}_e \left(\frac{w_e^+}{\hat{u}_e^+} - \frac{w_e^+}{\hat{u}_e^-}\right)\right) + (g_e)_{0.1}(\hat{f})^p$$

Where the subscripts of 0.1 denote that we are solving the quadratically smoothened function with the box size being $\hat{u}_e(f)/10$ for each e and $g_e(\hat{f}) = (\hat{u}_e^+(f))^2 \log\left(1 - \frac{\hat{f}_e}{\hat{u}^+}\right) + \hat{u}_e^+(f)\hat{u}_e^-(f) \log\left(1 + \frac{\hat{f}_e}{\hat{u}_e^-(f)}\right)$ Call the term in the sum for a given edge e as $val_e(\hat{f})$ and the overall objective function is $val(\hat{f})$. In particular, we consider for a single edge and prove the following lemma

Lemma 6. *We have the following for any feasible f and $\delta \geq 0$*

$$\begin{aligned}&\delta \partial_f val_e(f) + (0.9)^2 \delta^2 \left(\frac{w_e^+}{(\hat{u}_e^+ - f)^2} + \frac{w_e^-}{(\hat{u}_e^- + f)^2}\right) + 2^{-O(p)}(f_e^{2p-4}\delta^2 + \delta^p)\\ &\leq val_e(f+\delta) - val_e(f)\end{aligned}$$

and

$$\begin{aligned}&val_e(f+\delta) - val_e(f)\\ &\leq \delta \partial_f val_e(f) + (1.1)^2 \delta^2 \left(\frac{w_e^+}{(\hat{u}_e^+ - f)^2} + \frac{w_e^-}{(\hat{u}_e^- + f)^2}\right) + 2^{O(p)}(f_e^{2p-4}\delta^2 + \delta^p)\end{aligned}$$

where ∂_x denotes the derivative of a function with respect to x.

Proof. Note that while we are solving for the quadratically smoothened version of the problem, we can assume we solve it for the non-smoothened version in the box corresponding to a congestion of at most 0.1 as the extension is C^2 and will ensure that any inequalities we need henceforth (upto the second order terms) are bounded as well.

There are two terms, one corresponding to the potential decrement term and the other is a similar expression but raised to the p^{th} power. We tackle the first term first. This is easily done using Taylor's theorem. The function is $g(x+y) = -\log(1-(x+y)/u) - (x+y)/u$. Computing the first two derivatives with respect to y, we get that $g'(x+y) = \frac{1}{u-x-y} - 1/u$ and $g''(x+y) = \frac{1}{(u-x-y)^2}$. Now, using Taylor's theorem, we get that

$$\begin{aligned}g(x+y) &= g(x) + g'(x)y + \frac{1}{2}g''(x+\zeta)y^2\\ &= g(x) + y\left(\frac{1}{u-x-y} - \frac{1}{u}\right) + y^2\left(\frac{1}{(u-x-\zeta)^2}\right)\end{aligned}$$

for some ζ such that $-u/10 \leq x + \zeta \leq u/10$ which easily gives us the bounds

$$\begin{aligned}g(x) + y\left(\frac{1}{u-x-y} - \frac{1}{u}\right) + (9/10)^2 y^2 \left(\frac{1}{(u-x)^2}\right) &\leq g(x+y)\\ g(x+y) \leq g(x) + y\left(\frac{1}{u-x-y} - \frac{1}{u}\right) + (11/10)^2 y^2 \left(\frac{1}{(u-x)^2}\right)&\end{aligned}$$

The calculation and bounds are similar for $-\log(1+x/u) + x/u$.

Now, for the second term, we will largely follow the strategy of [74]. Now for the p^{th} order term, we have a function $g(x) = u_1^2 \log(1 - x/u_1) + u_1 u_2 \log(1 - x/u_2)$. We first use Lemma 1 with $f_i = g(x)$ and $\delta_i = g(x+y) - g(x)$ to get

$$\begin{aligned} &g(x+y)^p - g(x)^p \\ &\le pg(x)^{p-1}(g(x+y) - g(x)) + 2^{O(p)}(g(x)^{p-2}(g(x+y) - g(x))^2 + (g(x+y) - g(x))^p) \end{aligned}$$

Now, adding and subtracting $pg(x)^{p-1}yg'(x)$ from both sides and noticing that $g(x+y) - g(x) - yg'(x) \le 0$ from concavity of g , we get

$$g(x+y)^p \le g(x)^p + pyg(x)^{p-1}g'(x) + 2^{O(p)}(g(x)^{p-2}(g(x+y) - g(x))^2 + (g(x+y) - g(x))^p)$$

Now, notice that using inequalities of $\log(1 - x/u)$ and $\log(1 + x/u)$, to get $x^2 \le g(x) \le 2x^2$ and we also use Taylor's theorem get that $g(x+y) - g(x) \le 10(|xy| + |y|^2)$

$$\begin{aligned} g(x+y)^p &\le g(x)^p + pyg(x)^{p-1}g'(x) + 2^{O(p)}(x^{2p-4}(x^2y^2 + y^4) + 2^{p-1}(x^py^p + y^{2p}) \\ &\le g(x)^p + pyg'(x) + 2^{O(p)}(x^{2p-2}y^2 + y^{2p}) \end{aligned}$$

where we have used $(x+y)^p \le 2^{p-1}(x^p + y^p)$ and that $y \le x$ because that's the neighborhood we are considering. Beyond that neighborhood, we could just do the calculation with the quadratic extension parts (as we only used upto the second order information so that's still preserved)

The proof of the lower bound is similar. □

Let $r_e = \left(\frac{w_e^+}{(\hat{u}_e^+ - f)^2} + \frac{w_e^-}{(\hat{u}_e^- + f)^2}\right)$

Lemma 7. *Now, given an initial point f_0 such that $B^\top f_0 = \chi$ and an almost linear time solver for the following problem*

$$\min_{B^\top \delta = 0} \sum_{e \in E} \delta_e \alpha_e + (11/10)^2 2^{O(p)}((r_e + f_e^{2p-4})\delta^2 + \delta^p)$$

where the α_e vector is the gradient of val at a given point f, we can obtain an $\hat{f}$ in $\widetilde{O}_p(1)$ calls to the solver such that $val(\hat{f}) \le OPT + 1/2^{\text{poly}\log m}$

The proof is similar to the proof of the iteration complexity of gradient descent for smooth and strongly convex function and it follows from [88, 74]. Note that since [74] give an almost linear time solver for exactly the subproblem in the above lemma provided the resistances are quasipolynomially bounded, we are done. This is because Section D.1 in [86] already proves that the resistances are quasi-polynomially bounded.

2.5 Conclusion

In this chapter, we showed how to use steps inspired by potential reduction IPMs to solve max flow in directed graphs in $O(m^{4/3+o(1)}U^{1/3})$ time. After the appearance of this work, an almost linear time algorithm for maximum flow appeared in [33] after a line of work of [47, 26]. Their work also uses a potential reduction based IPM algorithm. However, they use ℓ_1 augmenting steps and hence takes $O(m)$ iterations but they design highly efficient data structures which allow the IPM updates to be implemented in $m^{o(1)}$ time per iteration. The author does, however, believe it should be possible to obtain an almost linear time algorithm for exact max flow which runs in $m^{o(1)}$ iterations (and is hence efficiently parallelizable) with each iteration taking $\widetilde{O}(m)$ time. This would parallel work of [110, 67, 99] which provides such a result for approximate max flow. It is also an extremely interesting problem if general linear programs, beyond just unit capacity max flow, can be solved in $m^{1/3}$ iterations of linear system solving, improving on the current $m^{1/2}$ iteration complexity. Such a result likely requires generalizing ideas from accelerated gradient descent [93] to the Riemannian/IPM setting and this inherently requires designing another interior point method, different than the central path, which allows us to argue that the (average) curvature (as measured by the length integral) of this new trajectory is significantly shorter than that corresponding to the central path.

Chapter 3

A Faster IPM for Semi-definite Programming

This chapter is based on joint work with Haotian Jiang, Yin Tat Lee, Swati Padmanabhan and Zhao Song and appeared at FOCS 2020 [61]

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 [123]. Applications in theoretical computer science include improved approximation algorithms for fundamental problems (e.g., Max-Cut [49], coloring 3-colorable graphs [63], and sparsest cut [10]), quantum complexity theory [59], robust learning and estimation [35, 34, 36], and algorithmic discrepancy and rounding [17, 19, 16]. We formally define SDPs with variable size $n \times n$ and m constraints:

Definition 2 (Semidefinite programming)**.** *Given symmetric*[1] *matrices $C, A_1, \cdots, A_m \in \mathbb{R}^{n \times n}$ and $b_i \in \mathbb{R}$ for all $i \in [m]$, the goal is to solve the convex optimization problem*

$$\max \langle C, X \rangle \text{ subject to } X \succeq 0, \langle A_i, X \rangle = b_i \ \forall i \in [m] \tag{3.1}$$

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

Cutting plane and interior point methods Two prominent methods for solving SDPs, with runtimes depending *logarithmically* on the accuracy parameter ϵ, are the *cutting plane method* and the *interior point method.*

[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 maintains a convex set containing the optimal solution. In each iteration, the algorithm queries a separation oracle, which returns a hyperplane that divides the convex set into two subsets. The convex set is then updated to contain the subset with the optimal solution. This process is repeated until the volume of the maintained set becomes small enough and a near-optimal solution can be found. Since Khachiyan proved [68] that the ellipsoid method solves linear programs in polynomial time, cutting plane methods have played a crucial role in both discrete and continuous optimization [53, 51].

In contrast, interior point methods add a barrier function to the objective and, by adjusting the weight of this barrier function, solve a different optimization problem in each iteration. The solutions to these successive problems form a well-defined *central path.* Since Karmarkar proved [64] that interior point methods can solve linear programs in polynomial time, these methods have become an active research area. Their number of iterations is usually the square root of the number of dimensions, as opposed to the linear dependence on dimensions 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 methods[2] of [83, 62] solve a general SDP in time $m(mn^2 + m^2 + n^\omega)$, while the fastest SDP solvers based on interior point methods in the work of [94] and [6] achieve runtimes of $\sqrt{n}(m^2n^2 + mn^\omega + m^\omega)$ and $(mn)^{1/4}(m^4n^2 + m^3n^\omega)$, respectively, which are slower in the most common regime of $m \in [n, n^2]$ (see Table 3.1.2). This apparent paradox raises the following natural question:

How fast can SDPs be solved using interior point methods?

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 given in Theorem 12.

Theorem 9 (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*[3] $O^*(\sqrt{n}(mn^2 + m^\omega + n^\omega))$.

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.

[2][62] improves upon the runtime of [83] in terms of the dependence on $\log(n/\epsilon)$, while the polynomial factors are the same in both runtimes.

[3]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.

- 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 3.1, 3.1, and 3.1 for more detail.

Table 3.1.1 compares our result with previous SDP solvers. The first takeaway of this table and Theorem 9 is that our interior point method always runs faster than that in [94] and is faster than that in [95] and [6] when $m \geq n^{1/13}$. A second consequence is that whenever $m \geq \sqrt{n}$, our interior point method is faster than the current fastest cutting plane method [83, 62]. We note that $n \leq m \leq n^2$ is satisfied in most SDP applications known to us, such as classical combinatorial optimization problems over graphs, experiment design problems in statistics and machine learning, and sum-of-squares problems. An explicit comparison to previous algorithms in the cases of $m = n$ and $m = n^2$ is shown in Table 3.1.2. We also include two works in Table 3.1.1 that contain results which appeared after the publication of our work which contains improvements when $m = \Omega(n^2)$ however our result remains the state of the art in the $n \leq m \leq n^2$ regime.

Year	**References**	**Method**	**#Iters**	**Cost per iter**
1979	[111, 127, 68]	CPM	m^2	$mn^2 + m^2 + n^\omega$
1988	[69, 96]	CPM	m	$mn^2 + m^{3.5} + n^\omega$
1989	[120]	CPM	m	$mn^2 + m^\omega + n^\omega$
1992	[94]	IPM	$\sqrt{n}$	$m^2n^2 + mn^\omega + m^\omega$
1994	[95, 6]	IPM	$(mn)^{1/4}$	$m^4n^2 + m^3n^\omega$
2003	[72]	CPM	m	$mn^2 + m^\omega + n^\omega$
2015	[83]	CPM	m	$mn^2 + m^2 + n^\omega$
2020	[62]	CPM	m	$mn^2 + m^2 + n^\omega$
2020	Our result	IPM	$\sqrt{n}$	$mn^2 + m^\omega + n^\omega$
2022	[57]	IPM	$\sqrt{n}$	$m^2 + n^2 + (m^\omega + n^{2\omega})n^{-1/2}$
2023	[124]	IPM	$n^2 + n^{3/2}m^{1/4}$	$mn^2 + n^\omega$

Table 3.1.1: Summary of key SDP algorithms. CPM stands for cutting plane method, and IPM, interior point method. n is the size of the variable matrix, and $m \leq n^2$ is the number of constraints. Runtimes hide $n^{o(1)}$, $m^{o(1)}$ and $\operatorname{poly}\log(1/\epsilon)$ factors, where ϵ is the accuracy parameter. [6] simplifies the proofs in [95, Section 5.5]. Neither [6] nor [95] explicitly analyzed their runtimes, and their runtimes shown here are our best estimates.

Even in the more general case where the SDP might not be dense, where $\mathrm{nnz}(A)$ 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 methods[83, 62], which run in time $O^*(m(\mathrm{nnz}(A) + m^2 + n^\omega))$.

Year	References	Method	Runtime	
			$m = n$	$m = n^2$
1979	[111, 127, 68]	CPM	n^5	n^8
1988	[69, 96]	CPM	$n^{4.5}$	n^9
1989	[120]	CPM	n^4	$n^{6.746}$
1992	[94]	IPM	$n^{4.5}$	$n^{6.5}$
1994	[95, 6]	IPM	$n^{6.5}$	$n^{10.75}$
2003	[72]	CPM	n^4	$n^{6.746}$
2015	[83]	CPM	n^4	n^6
2020	[62]	CPM	n^4	n^6
2020	Our result	IPM	$n^{3.5}$	$n^{5.246}$

Table 3.1.2: Total runtimes for the algorithms in Table 3.1.1 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 $\operatorname{poly}\log(1/\epsilon)$ factors, where ϵ is the accuracy parameter and assume ω to equal its currently best known upper bound of 2.373.

Theorem 10 (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* $\operatorname{nnz}(A)$ *input size, faster than the current best cutting plane method [83, 62], over all regimes of* $\operatorname{nnz}(A)$.

Technique overview

Interior point method for solving SDPs

By removing redundant constraints, we can, without loss of generality, assume $m \leq n^2$ in the primal formulation of the SDP (3.1). Thereafter, instead of solving the primal SDP, which has variable size $n \times n$, we solve its dual formulation, which has dimension $m \leq n^2$:

$$\min b^\top y \text{ subject to } S = \sum_{i=1}^{m} y_i A_i - C, \text{ and } S \succeq 0. \tag{3.2}$$

Interior point methods solve (3.2) by minimizing the penalized objective function:

$$\min_{y \in \mathbb{R}^m} f_\eta(y), \text{ where } f_\eta(y) := \eta \cdot b^\top y + \phi(y), \tag{3.3}$$

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 (3.3) becomes sufficiently large, at which point the minimizer of f_η is provably close to the optimal solution of (3.2). The iterates y generated by this method follow a *central path*. Different choices of the barrier function ϕ lead to different run times in solving (3.3), as we next describe.

The log barrier Nesterov and Nemirovski [94] use the *log barrier* function,

$$\phi(y) = g(y) := -\log\det\left(\sum_{i=1}^{m} y_i A_i - C\right), \tag{3.4}$$

in (3.3) and, in $O(\sqrt{n}\log(n/\epsilon))$ iterations, obtain a feasible dual solution y that satisfies $b^\top y \le b^\top y^* + \epsilon$, where $y^* \in \mathbb{R}^m$ is the optimal solution for (3.2). 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

$$H_{j,k} = \operatorname{tr}[S^{-1}A_j S^{-1}A_k]. \tag{3.5}$$

The analysis of [94] 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 $\operatorname{tr}[S^{-1}A_j S^{-1}A_k]$ 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 $O^*(\sqrt{n}(m^2n^2 + mn^\omega + m^\omega))$.

The volumetric barrier Vaidya [120] introduced the *volumetric barrier* for a polyhedral set $\{x \in \mathbb{R}^n : Ax \ge c\}$, where $A \in \mathbb{R}^{m\times n}$ and $c \in \mathbb{R}^m$. Nesterov and Nemirovski [95] studied the following extension of the volumetric barrier to the convex subset $\{y \in \mathbb{R}^m : \sum_{i=1}^m y_i A_i \succeq C\}$ of the polyhedral cone:

$$V(y) = \frac{1}{2}\log\det(\nabla^2 g(y)),$$

where $g(y)$ is the log barrier function defined in (3.4). They proved that choosing $\phi(y) = \sqrt{n}V(y)$ in (3.3) 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 [94] when $m \le \sqrt{n}$. 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 $m \le n$, this iteration complexity is lower than $\widetilde{O}(\sqrt{n})$ of [94]. We refer readers to the much simpler proofs in [6] 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 $(j,k) \in [m] \times [m]$,

$$\begin{aligned} Q(y)_{j,k} &= \operatorname{tr}\left[\mathcal{A}H^{-1}\mathcal{A}^\top \left(\left(S^{-1}A_jS^{-1}A_kS^{-1}\right)\widehat{\otimes}S^{-1}\right)\right], \\ R(y)_{j,k} &= \operatorname{tr}\left[\mathcal{A}H^{-1}\mathcal{A}^\top \left(\left(S^{-1}A_jS^{-1}\right)\widehat{\otimes}\left(S^{-1}A_kS^{-1}\right)\right)\right], \\ T(y)_{j,k} &= \operatorname{tr}\left[\mathcal{A}H^{-1}\mathcal{A}^\top \left(\left(S^{-1}A_jS^{-1}\right)\widehat{\otimes}S^{-1}\right)\mathcal{A}H^{-1}\mathcal{A}^\top \left(\left(S^{-1}A_kS^{-1}\right)\widehat{\otimes}S^{-1}\right)\right]. \end{aligned} \tag{3.6}$$

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 3.2for formal definition). Due to the complicated formulas in (3.6), 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.

Our techniques

Given the inefficiency of implementing the volumetric and volumetric-logarithmic barriers discussed above, this work uses the log barrier in (3.4). We now describe some of our key techniques that improve the runtime of the Nesterov-Nemirovski interior point method [94].

Hessian computation using fast rectangular matrix multiplication As noted in Section 3.1, the runtime bottleneck in [94] is computing the inverse of the Hessian of the log barrier function, where the Hessian is described in (3.5). In [94], each of these m^2 entries is computed separately, resulting in a runtime of $O(m^2n^2)$ per iteration.

Instead contrast, we show below how to group these computations using rectangular matrix multiplication. The expression from (3.5) can be re-written as

$$H_{j,k} = \operatorname{tr}[S^{-1/2}A_jS^{-1/2} \cdot S^{-1/2}A_kS^{-1/2}]. \tag{3.7}$$

We first compute the key quantity $S^{-1/2}A_jS^{-1/2} \in \mathbb{R}^{n\times n}$ 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 $\mathcal{T}_{\mathrm{mat}}(n, mn, n)$[4] using fast rectangular matrix multiplication. We then flatten each $S^{-1/2}A_jS^{-1/2}$ into a row vector of length n^2 and stack all m vectors to form a matrix $\mathcal{B}$ of size $m \times n^2$, i.e., the j-th row of $\mathcal{B}$ is $\mathcal{B}_j = \mathsf{vec}(S^{-1/2}A_jS^{-1/2})$. It follows that the Hessian can be computed as

$$H = \mathcal{B}\mathcal{B}^\top, \tag{3.8}$$

[4]See Section 3.3 for the definition.

which takes time $\mathcal{T}_{\mathrm{mat}}(m, n^2, m)$ by applying fast rectangular matrix multiplication. By leveraging recent developments in this area [46], this approach already improves upon the runtime in [94].

Thus far, we have reduced the per iteration cost of $O^*(m^2n^2 + mn^\omega)$ 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 (3.7) might change arbitrarily in the Nesterov-Nemirovski interior point method.

To overcome this problem, we propose a new interior point method that maintains 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 in $\widetilde{S}_t$. This process of 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 intuitively 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 following *rank inequality* (see Theorem 14for the formal statement).

Theorem 11 (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

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

The key component to proving Theorem 11 is the 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 11.

Specifically, first we prove that whenever S is updated in an iteration, the potential function *increases* by at most $\widetilde{O}(1)$ (see Lemma 21). 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 13 that the consecutive slack matrices S_t and S_{t+1} satisfy

$$\|S_t^{-1/2} S_{t+1} S_t^{-1/2} - I\|_F = O(1) \tag{3.9}$$

and combine this bound with the Hoffman-Wielandt theorem [56], which relates the ℓ_2 distance between the spectrum of two matrices with the Frobenius norm of their difference (see Fact 2). Next, when $\widetilde{S}$ gets 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 $\widetilde{S}_t$, results in a potential decrease of at least $\widetilde{O}(\sqrt{r_t})$ (see Lemma 22).

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

$$\widetilde{H}_{j,k} = \mathrm{tr}[\widetilde{S}^{-1} A_j \widetilde{S}^{-1} A_k] \tag{3.10}$$

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 17). This approximate Hessian therefore suffices for our algorithm to approximately follow the central path.

To efficiently update the approximate Hessian $\widetilde{H}$ in (3.10), 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 4). The change in $\widetilde{S}^{-1}$ can be expressed as

$$\Delta(\widetilde{S}^{-1}) = V_+ V_+^\top - V_- V_-^\top, \tag{3.11}$$

where $V_+, V_- \in \mathbb{R}^{n\times r}$. Plugging (3.11) into (3.10), we can express $\Delta\widetilde{H}_{j,k}$ as the sum of multiple terms, among the costliest of which are those of the form $\mathrm{tr}[\widetilde{S}^{-1}A_jVV^\top A_k]$, where $V \in \mathbb{R}^{n\times r}$ is either V_+ or V_-. We compute $\mathrm{tr}[\widetilde{S}^{-1}A_jVV^\top A_k]$ 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_jV 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 8). Finally, by flattening each $\widetilde{S}^{-1/2}A_jV$ 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

$$\widetilde{\mathcal{B}}_j = \mathsf{vec}(\widetilde{S}^{-1/2}A_jV),$$

the task of computing $\mathrm{tr}[\widetilde{S}^{-1}A_jVV^\top A_k]$ for all $(j,k)\in[m]\times[m]$ reduces to computing $\widetilde{\mathcal{B}}\widetilde{\mathcal{B}}^\top$, which costs $\mathcal{T}_{\mathrm{mat}}(m,nr,m)$.

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

$$\sum_{t=1}^{T}\left(\mathcal{T}_{\mathrm{mat}}(r_t,n,mn)+\mathcal{T}_{\mathrm{mat}}(n,mr_t,n)+\mathcal{T}_{\mathrm{mat}}(m,nr_t,m)\right), \tag{3.12}$$

where r_t is the rank of the update on $\widetilde{S}_t$. Applying Theorem 11 with several properties of fast rectangular matrix multiplication that we prove in Section 3.3 , we upper bound the runtime in (3.12) by

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

which implies Theorem 9. In Section 3.1 and 3.1, we discuss bottlenecks to further improving our runtime.

Bottlenecks of our interior point method

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 as

$$S=\sum_{i\in[m]} y_iA_i - C.$$

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. This is the first appearance of the mn^2 cost per iteration.

Gradient computation. Recall from (3.3) that our interior point method follows the central path defined via the penalized objective function

$$\min_{y\in\mathbb{R}^m} f_\eta(y) \quad \text{where} \quad f_\eta(y) := \eta b^\top y + \phi(y),$$

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 as

$$g_\eta(y)_j = \eta \cdot b_j - \mathrm{tr}[S^{-1}A_j] \tag{3.13}$$

for each coordinate $j \in [m]$. Even if we are given S^{-1}, it still requires mn^2 time to compute (3.13) for all $j \in [m]$. This is the second appearance of the per iteration cost of mn^2.

Approximate Hessian computation. Recall from Section 3.1 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

$$\Delta_{j,k} = \mathrm{tr}[\widetilde{S}^{-1/2}A_jV \cdot V^\top A_k\widetilde{S}^{-1/2}],$$

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. This is the third bottleneck that leads to the mn^2 term in the cost per iteration.

LP techniques unlikely to improve SDP runtime

The preceding discussion of bottlenecks suggests that reading the entire input in each iteration, which takes mn^2 time per iteration, stands as a natural barrier to further improving the runtime of SDP solvers based on interior point methods.

In the context of linear programming (LP), several recent results [38, 29] yield faster interior point methods that bypass reading the entire input in every iteration. Two techniques crucial to these results are: (1) showing that the Hessian (projection matrix) admits low-rank updates, and (2) speeding computation of the Hessian via sampling.

We now describe these techniques in the context of SDP and argue that they are unlikely to improve our runtime.

Showing that the Hessian admits low-rank updates. We saw in Section 3.1 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 (3.9). One natural question that follows is whether a similar upper bound can be obtained for the Hessian. If such a result could be proved, then one could maintain an approximate Hessian that admitted low-rank updates, which would speed up the approximate Hessian computation. Indeed, in the context of LP, such a bound for the Hessian can be proved (e.g., [29, Lemma 47]).

Unfortunately, it is impossible to prove such a statement for the Hessian in the context of SDP. To show this, it is convenient to express the Hessian using the Kronecker product (Section 3.2)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 $c > 0$. Consider the simple example where $\mathcal{A} = I$ (we are assuming here that $m = n^2$ so that $\mathcal{A}$ is a square matrix), which implies that the change in the Hessian can be approximately computed as

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

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 LP and SDP.

Sampling for faster Hessian computation. Recall from (3.8) 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. This could reduce the dimension of the matrix multiplication and speed up the Hessian computation. Indeed, sampling techniques have been successfully used to obtain faster LP solvers [38, 29].

For SDP, 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.

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 [43, 68, 64, 121, 122, 81, 80, 113, 77, 38, 28, 29].

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 [111, 127, 68, 69, 96, 120, 12, 24, 83, 62].

First-Order SDP Algorithms. As the focus of this work, 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 [9, 48, 4, 32, 78, 128], positive SDPs [58, 101, 3, 60].)

3.2 Preliminaries

Notation

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 $\mathrm{diag}(v)$ to denote the diagonal $n \times n$ matrix with $\mathrm{diag}(v)_{i,i} = v_i$. 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 $B \in \mathbb{R}^{k\times l}$, the *Kronecker product* of A and B, denoted as $A \otimes B$, is defined as the $mk \times nl$ 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), 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) is denoted by $\mathrm{nnz}(A)$.

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 work), 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 invertible 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 1 (Generalized Lieb-Thirring Inequality [44, 3, 60])**.** *Given a symmetric matrix B, a positive semi-definite matrix A and $\alpha \in [0,1]$, we have*

$$\mathrm{tr}[A^\alpha B A^{1-\alpha} B] \le \mathrm{tr}[AB^2].$$

Fact 2 (Hoffman-Wielandt Theorem, [54, 56])**.** *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 $\sum_{i\in[n]} |\widehat{\lambda}_{\sigma(i)} - \lambda_i|^2 \le \|E\|_F^2 = \mathrm{tr}[E^*E]$.*

Fact 3 (Corollary of the Hoffman-Wielandt Theorem, [56])**.** *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 4 (Woodbury matrix identity, [126, 125])**.** *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*

$$(A+UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}.$$

3.3 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 13, 14, and 15):

- 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 3.7, these results (as well as several intermediate results) might be of independent interest.

Exponent of matrix multiplication

We need the following definitions to describe the cost of certain fundamental matrix operations we use.

Definition 3. *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 4. *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 8 ([31, 25])**.** *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 [46] for the latest upper bounds on $\omega(k)$ for different values of k. In particular, we need the following upper bounds in our work.

Lemma 9 ([46])**.** *We have:*

- $\omega = \omega(1) \leq 2.372927$,
- $\omega(1.5) \leq 2.79654$,
- $\omega(1.75) \leq 3.02159$,
- $\omega(2) \leq 3.251640$.

Technical results for matrix multiplication

In this section, we derive some technical results on $\mathcal{T}_{\mathrm{mat}}$ and ω that we extensively use for our runtime analysis. Some of these results can be derived using tensors, and we demonstrate this in Section 3.10. We hope that the use of tensors can yield better runtimes for this problem in future.

Lemma 10 (Sub-linearity)**.** *For any $p \geq q \geq 1$, we have*

$$\omega(p) \leq p - q + \omega(q).$$

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 11 (Convexity)**.** *The fast rectangular matrix multiplication time exponent $\omega(k)$ as defined in Definition 4 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 $\alpha p \leq k$, 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.

$$\begin{aligned}\mathcal{T}_{\mathrm{mat}}(n, n^k, n) \leq & \mathcal{T}_{\mathrm{mat}}(n^\alpha, n^{\alpha p}, n^\alpha) \cdot \mathcal{T}_{\mathrm{mat}}(n/n^\alpha, n^k/n^{\alpha p}, n/n^\alpha) \\ = & \mathcal{T}_{\mathrm{mat}}(n^\alpha, n^{\alpha p}, n^\alpha) \cdot \mathcal{T}_{\mathrm{mat}}(n^{(1-\alpha)}, n^{(1-\alpha)q}, n^{(1-\alpha)}) \\ \leq & n^{\alpha \cdot \omega(p) + o(1)} \cdot n^{(1-\alpha)\cdot\omega(q)+o(1)}.\end{aligned}$$

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 4, $m^{\omega(p)+o(1)}$, which is exactly $n^{\alpha(\omega(p)+o(1))}$. Applying Definition 4 and comparing exponents, this implies that

$$\omega(k) \leq \alpha \cdot \omega(p) + (1-\alpha) \cdot \omega(q),$$

which proves the convexity of the function $\omega(k)$. □

Claim 1. $\omega(1.68568) \leq 2.96370$.

Proof. We can upper bound $\omega(1.68568)$ in the following sense

$$\begin{aligned}\omega(1.68568) &= \omega(0.25728 \cdot 1.5 + (1 - 0.25728) \cdot 1.75) \\ &\leq 0.25728 \cdot \omega(1.5) + (1 - 0.25728) \cdot \omega(1.75) \\ &\leq 0.25728 \cdot 2.79654 + (1 - 0.25728) \cdot 3.02159 \\ &\leq 2.96370,\end{aligned}$$

where the first step follows from convexity of ω (Lemma 11), the third step follows from $\omega(1.5) \leq 2.79654$ and $\omega(1.75) \leq 3.02159$ (Lemma 9). □

Lemma 12. *Let $\mathcal{T}_{\mathrm{mat}}$ be defined as in Definition 3. Then for any positive integers h, ℓ, and k, we have*

$$\mathcal{T}_{\mathrm{mat}}(h, \ell k, h) \leq O(\mathcal{T}_{\mathrm{mat}}(hk, \ell, hk)).$$

Proof. Given any matrices $A, B^\top \in \mathbb{R}^{h \times \ell k}$, by Definition 3, 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 $A' \in \mathbb{R}^{hk \times \ell}$. Similarly, we stack the k matrices $B_1, \cdots, B_k$ horizontally to form a matrix $B' = (B_1, \cdots, B_k) \in \mathbb{R}^{\ell \times hk}$. By Definition 3, we can compute $A'B' \in \mathbb{R}^{hk,hk}$ in time $\mathcal{T}_{\mathrm{mat}}(hk, \ell, hk)$. 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_jB_j, and summing up the k diagonal $h \times h$ blocks of $A'B'$ gives AB. □

General upper bound on $\mathcal{T}_{\mathrm{mat}}(n, mn, n)$ and $\mathcal{T}_{\mathrm{mat}}(m, n^2, m)$

Lemma 13. *Let $\mathcal{T}_{\mathrm{mat}}$ be defined as in Definition 3.*
If $m \geq n$, then we have

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

If $m \leq n$, then we have

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

Proof. We only prove the case of $m \geq n$, as the other case where $m < n$ is similar. This is an immediate consequence of Lemma 12 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 $\mathcal{T}_{\mathrm{mat}}(m, n^2, m)$ when $m \geq n$ and $\mathcal{T}_{\mathrm{mat}}(n, mn, n)$ when $m < n$, which is crucial to our runtime analysis.

Lemma 14. *Let $\mathcal{T}_{\mathrm{mat}}$ be defined as in Definition 3 and ω be defined as in Definition 4.*
Property I. We have

$$\mathcal{T}_{\mathrm{mat}}(n, mn, n) \leq O(mn^{\omega+o(1)}).$$

Property II. We have

$$\mathcal{T}_{\mathrm{mat}}(m, n^2, m) \leq O\left(\sqrt{n}\left(mn^2 + m^{\omega}\right)\right).$$

Proof. **Property I.**

Recall from Definition 3 that $\mathcal{T}_{\mathrm{mat}}(n, mn, n)$ 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 $\mathcal{T}_{\mathrm{mat}}(n, mn, n) \leq O(mn^{\omega+o(1)})$, as claimed.

Property II.

Let $m = n^a$, where $a \in (0, \infty)$. By definition, $\mathcal{T}_{\mathrm{mat}}(m, n^2, m)$ 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 4, that

$$\mathcal{T}_{\mathrm{mat}}(m, n^2, m) = m^{\omega(2/a)+o(1)} = n^{a\cdot\omega(2/a)+o(1)}$$

Property II is then an immediate consequence of the following inequality, which we prove next:

$$\omega(2/a) < \max(1 + 2.5/a, \omega(1) + 0.5/a) \qquad \forall a \in (0, \infty). \tag{3.14}$$

Define $b = 2/a \in (0, \infty)$. Then the desired inequality in (3.14) can be expressed in terms of b as

$$\omega(b) < \max(1 + 5b/4, \omega(1) + b/4) \qquad \forall b \in (0, \infty). \tag{3.15}$$

Notice that the RHS of (3.15) 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 11, it suffices to verify (3.15) at the endpoints $b \to 0$, $b \to \infty$ and $b = b^*$. In the case where $b = \delta$ for any $\delta < 1$, (3.15) follows immediately from the observation that $\omega(\delta) < \omega(1)$. We next argue about the case $b \to \infty$. By Lemma 9 we have $\omega(2) \le 3.252$. Using Lemma 10, we have $\omega(b) \le b - 2 + \omega(2)$. Combining these two facts implies that for any $b > 2$, we have

$$\omega(b) \le b - 2 + \omega(2) \le 1 + 5b/4,$$

which again satisfies (3.15). The final case is $b = b^* = \omega(1) - 1$, for which (3.15) is equivalent to

$$\omega(\omega(1) - 1) < 5\omega(1)/4 - 1/4. \tag{3.16}$$

By Lemma 9, we have that $\omega(1) - 2 \in [0, 0.372927]$. Then to prove (3.16), it is sufficient to show that

$$\omega(t + 1) < 5t/4 + 9/4 \qquad \forall t \in [0, 0.372927]. \tag{3.17}$$

By the convexity of $\omega(\,\cdot\,)$ as proved in Lemma 11, the upper bound of $\omega(2) \le 3.251640$ in Lemma 9, 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

$$\begin{aligned} \omega(t+1) - 5t/4 - 9/4 &\le (t+2) + t \cdot (1.251640 - t) - 5t/4 - 9/4 \\ &= -t^2 + 1.00164t - 1/4, \end{aligned}$$

which is negative on the entire interval $[0, 0.372927]$. This establishes (3.17) and finishes the proof. $\square$

Specific upper bound on $\mathcal{T}_{\mathrm{mat}}(m, n^2, m)$

Lemma 15. *For any two positive integers n and m, we have*

$$\mathcal{T}_{\mathrm{mat}}(m, n^2, m) = o\left(m^3 + mn^{2.37}\right).$$

Proof. Let $m = n^a$ where $a \in (0, \infty)$. Recall that $\mathcal{T}_{\mathrm{mat}}(m, n^2, m) = m^{\omega(2/a)+o(1)} = n^{a\omega(2/a)+o(1)}$. We consider the following two cases according to the range of a.

Case 1: $a \in [1.18647, \infty)$. In this case, we have $\omega(2/a) \leq \omega(2/1.18647) \leq \omega(1.68568) < 3$, where the last inequality follows from Claim 1. This implies that

$$\mathcal{T}_{\mathrm{mat}}(m, n^2, m) = o(n^{3a}) = o(m^3). \tag{3.18}$$

Case 2: $a \in (0, 1.18647]$. In this case, we have $2/a \in [1.68567, \infty)$. Consider the linear function

$$y(t) = 1 + 2.37 \cdot \frac{t}{2}. \tag{3.19}$$

By Claim 1, we have

$$\omega(1.68567) < 2.997 \leq y(1.68567). \tag{3.20}$$

By Lemma 9, we have

$$\omega(2) < 3.37 = y(2). \tag{3.21}$$

An application of Lemma 10 then gives, for any $t \geq 2$, the inequality

$$\omega(t) \leq t - 2 + \omega(2) < t - 2 + y(2) \leq y(t), \tag{3.22}$$

where the last inequality is by definition of $y(t)$ from (3.19). Therefore, combining the convexity of $\omega(\,\cdot\,)$, as proved in Lemma 11, with (3.20), (3.21), and (3.22), 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

$$\mathcal{T}_{\mathrm{mat}}(m, n^2, m) = n^{a \cdot \omega(2/a)+o(1)} = o(n^{a+2.37}) = o(mn^{3.27}). \tag{3.23}$$

Combining the results from (3.18) and (3.23) finishes the proof of the lemma. □

3.4 Main Theorem

In this section, we give the formal statement of our main result.

Theorem 12 (Main result, formal)**.** *Consider a semidefinite program with variable size $n \times n$ and m constraints (assume there are no redundant constraints):*

$$\max \langle C, X\rangle \text{ subject to } X \succeq 0, \langle A_i, X\rangle = b_i \text{ for all } i \in [m]. \tag{3.24}$$

Assume that any feasible solution $X \in \mathbb{S}_{\geq 0}^{n\times n}$ satisfies $\|X\|_{\mathrm{op}} \leq R$. 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 $X \in \mathbb{R}_{\geq 0}^{n\times n}$ such that

$$\langle C, X\rangle \geq \langle C, X^*\rangle - \delta \cdot \|C\|_{\mathrm{op}} \cdot R \quad \textit{and} \quad \sum_{i\in[m]} \left|\langle A_i, \widehat{X}\rangle - b_i\right| \leq 4n\delta \cdot (R \sum_{i\in[m]} \|A_i\|_1 + \|b\|_1),$$

where ω is the exponent of matrix multiplication, X^* is any optimal solution to the semidefinite program in (3.24)*, and $\|A_i\|_1$ is the Schatten 1-norm of matrix A_i.*

The proof of Theorem 12 is given in the subsequent sections.

3.5 Approximate Central Path via Approximate Hessian

Main result for approximate central path

Our main result of this section is the following.

Theorem 13 (Approximate central path)**.** *Consider a semidefinite program as in Definition 2 with no redundant constraints. Assume that any feasible solution $X \in \mathbb{S}_{\geq 0}^{n\times n}$ satisfies $\|X\|_{\mathrm{op}} \leq R$. 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 1 outputs, in $T = \frac{40}{\epsilon_N}\sqrt{n}\log(n/\delta)$ iterations, a positive semidefinite matrix $X \in \mathbb{R}_{\geq 0}^{n\times n}$ that satisfies*

$$\langle C, X\rangle \geq \langle C, X^*\rangle - \delta \cdot \|C\|_{\mathrm{op}} \cdot R \quad \textit{and} \quad \sum_{i\in[m]} \left|\langle A_i, \widehat{X}\rangle - b_i\right| \leq 4n\delta \cdot (R \sum_{i\in[m]} \|A_i\|_1 + \|b\|_1), \tag{3.25}$$

where X^* is any optimal solution to the semidefinite program in Definition 2, and $\|A_i\|_1$ is the Schatten 1-norm of matrix A_i. Further, in each iteration of Algorithm 1, the following invariant holds for $\alpha_H = 1.03$:

$$\|S^{-1/2} S_{\mathrm{new}} S^{-1/2} - I\|_F \leq \alpha_H \cdot \epsilon_N. \tag{3.26}$$

Proof. At the start of Algorithm 1, Lemma 25 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 $g_\eta(y)^\top H(y)^{-1} g_\eta(y) \leq \epsilon_N^2$ holds at the start of the algorithm. Therefore, by Lemma 18 and Lemma 19, 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 1 grows to $\eta = (1 + \frac{\epsilon_N}{20\sqrt{n}})^T/(n+2) \geq 2n/\delta^2$. It then follows from Lemma 20 that

$$b^\top y \leq b^\top y^* + \frac{n}{\eta} \cdot (1 + 2\epsilon_N) \leq b^\top y^* + \delta^2.$$

Thus when the algorithm stops, the dual solution y has duality gap at most δ^2 for the modified SDP. Lemma 25 then shows how to obtain an approximate solution to the original SDP that satisfies the guarantees in (3.25).

To prove (3.26), 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. We rewrite $\|S^{-1/2} S_{\text{new}} S^{-1/2} - I\|_F^2$ as

$$\begin{aligned}
\|S^{-1/2} S_{\text{new}} S^{-1/2} - I\|_F^2 &= \operatorname{tr}\left[(S^{-1/2}(\Delta_S)S^{-1/2})^2\right] \\
&= \operatorname{tr}\left[S^{-1}\left(\sum_{i=1}^m \delta_{y,i} A_i\right) S^{-1}\left(\sum_{j=1}^m \delta_{y,j} A_j\right)\right] \\
&= \sum_{i,j=1}^m \delta_{y,i}\delta_{y,j} \operatorname{tr}[S^{-1}A_i S^{-1} A_j] \\
&= (\delta_y)^\top H(y) \delta_y \\
&= g_\eta(y)^\top \widetilde{H}(y)^{-1} H(y) \widetilde{H}(y)^{-1} g_\eta(y),
\end{aligned} \tag{3.27}$$

where we used the fact that $\Delta_S = \sum_{i=1}^m (\delta_y)_i A_i$. It then follows from Lemma 18 and the invariant $g_\eta(y)^\top H(y)^{-1} g_\eta(y) \leq \epsilon_N^2$ that

$$g_\eta(y)^\top \widetilde{H}(y)^{-1} H(y) \widetilde{H}(y)^{-1} g_\eta(y) \leq \alpha_H^2 \cdot \epsilon_N^2, \tag{3.28}$$

where $\alpha_H = 1.03$. Combining Equation (3.27) with Inequality (3.28) completes the proof of the theorem. □

Table 3.5.1: Summary of parameters in approximate central path for SDP.

Notation	**Choice**	**Appearance**	**Meaning**
α_H	1.03	Lemma 18	Approx. factor $\alpha_H^{-1} \cdot H \preceq \widetilde{H} \preceq \alpha_H \cdot H$
ϵ_N	0.1	Lemma 19	Upper bound on Newton step size $(g_\eta^\top H^{-1} g_\eta)^{1/2}$
ϵ_S	0.01	Algorithm 2	Approx. error $(1-\epsilon_S) \cdot S \preceq \widetilde{S} \preceq (1+\epsilon_S) \cdot S$

Algorithm 1

```
1: procedure MAIN($n, m, \delta, \epsilon_N, C, A, b$)    ▷ $C \in \mathbb{S}^{n \times n}$, $\{A_i\}_{i=1}^m \in \mathbb{S}^{n \times n}$, vector $b \in \mathbb{R}^m$, error parameter $0 < \delta < 0.1$, Newton step size parameter $0 < \epsilon_N < 0.1$
2:     Modify the SDP and obtain an initial dual solution $y$ according to Lemma 25
3:     $\eta \leftarrow 1/(n+2)$
4:     $T \leftarrow \frac{40}{\epsilon_N}\sqrt{n}\log\left(\frac{n}{\delta}\right)$
5:     $\widetilde{S} \leftarrow S \leftarrow \sum_{i\in[m]} y_i A_i - C$.
6:     for iter $= 1 \to T$ do
7:         $\eta_{\mathrm{new}} \leftarrow \eta\left(1 + \frac{\epsilon_N}{20\sqrt{n}}\right)$
8:         for $j = 1, \cdots, m$ do                                  ▷ Gradient computation
9:             $g_{\eta_{\mathrm{new}}}(y)_j \leftarrow \eta_{\mathrm{new}} \cdot b_j - \mathrm{tr}[S^{-1} \cdot A_j]$
10:        end for
11:        for $j = 1, \cdots, m$ do                                  ▷ Hessian computation
12:            for $k = 1, \cdots, m$ do
13:                $\widetilde{H}_{j,k}(y) \leftarrow \mathrm{tr}[\widetilde{S}^{-1} \cdot A_j \cdot \widetilde{S}^{-1} \cdot A_k]$
14:            end for
15:        end for
16:        $\delta_y \leftarrow -\widetilde{H}(y)^{-1} g_{\eta_{\mathrm{new}}}(y)$                      ▷ Update on $y$
17:        $y_{\mathrm{new}} \leftarrow y + \delta_y$                                ▷ Approximate Newton step
18:        $S_{\mathrm{new}} \leftarrow \sum_{i\in[m]} (y_{\mathrm{new}})_i A_i - C$
19:        $\widetilde{S}_{\mathrm{new}} \leftarrow$ APPROXSLACKUPDATE($S_{\mathrm{new}}, \widetilde{S}$)     ▷ Approximate slack computation
20:        $y \leftarrow y_{\mathrm{new}}$, $S \leftarrow S_{\mathrm{new}}$, $\widetilde{S} \leftarrow \widetilde{S}_{\mathrm{new}}$                 ▷ Update variables
21:    end for
22:    Return an approximate solution to the original SDP according to Lemma 25
23: end procedure
```

Approximate slack update

Lemma 16. *Given positive definite matrices $S_{\mathrm{new}}, \widetilde{S} \in \mathbb{S}^{n\times n}_{>0}$ and any parameter $0 < \epsilon_S < 0.01$, there is an algorithm (procedure* APPROXSLACKUPDATE *in Algorithm 2) that takes $O(n^{\omega+o(1)})$ time to output a positive definite matrix $\widetilde{S}_{\mathrm{new}} \in \mathbb{S}^{n\times n}_{>0}$ such that*

$$\|S_{\mathrm{new}}^{-1/2}\widetilde{S}_{\mathrm{new}}S_{\mathrm{new}}^{-1/2} - I\|_{\mathrm{op}} \leq \epsilon_S. \tag{3.29}$$

Proof. The 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. To prove (3.29), we notice that λ_{new} are the eigenvalues of $S_{\mathrm{new}}^{-1/2}\widetilde{S}_{\mathrm{new}}S_{\mathrm{new}}^{-1/2} - I$ and by the algorithm description (lines 6 - 13), the upper bound $(\lambda_{\mathrm{new}})_i \leq \epsilon_S$ holds for each $i \in [n]$. □

Algorithm 2 Approximate Slack Update

```
1: procedure ApproxSlackUpdate($S_{\text{new}}, \widetilde{S}$)    ▷ $S_{\text{new}}, \widetilde{S} \in \mathbb{S}^{n\times n}_{\geq 0}$ are positive definite matrices
2:     $\epsilon_S \leftarrow 0.01$    ▷ Spectral approximation constant
3:     $Z_{\text{mid}} \leftarrow S_{\text{new}}^{-1/2} \cdot \widetilde{S} \cdot S_{\text{new}}^{-1/2} - I$
4:     Compute spectral decomposition $Z_{\text{mid}} = U \cdot \Lambda \cdot U^\top$
5:         ▷ $\Lambda = \text{diag}(\lambda_1, \cdots, \lambda_n)$ are the eigenvalues of $Z_{\text{mid}}$, and $U \in \mathbb{R}^{n\times n}$ is orthogonal
6:     Let $\pi : [n] \to [n]$ be a sorting permutation such that $|\lambda_{\pi(i)}| \geq |\lambda_{\pi(i+1)}|$
7:     if $|\lambda_{\pi(1)}| \leq \epsilon_S$ then
8:         $\widetilde{S}_{\text{new}} \leftarrow \widetilde{S}$
9:     else
10:        $r \leftarrow 1$
11:        while $|\lambda_{\pi(2r)}| > \epsilon_S$ or $|\lambda_{\pi(2r)}| > (1 - 1/\log n)|\lambda_{\pi(r)}|$ do
12:            $r \leftarrow r + 1$
13:        end while
14:        $(\lambda_{\text{new}})_{\pi(i)} \leftarrow \begin{cases} 0 & \text{if } i = 1, 2, \cdots, 2r; \\ \lambda_{\pi(i)} & \text{otherwise.} \end{cases}$
15:        $\widetilde{S}_{\text{new}} \leftarrow \widetilde{S} + S_{\text{new}}^{1/2} \cdot U \cdot \text{diag}(\lambda_{\text{new}} - \lambda) \cdot U^\top \cdot S_{\text{new}}^{1/2}$
16:    end if
17:    return $\widetilde{S}_{\text{new}}$
18: end procedure
```

Closeness of slack implies closeness of Hessian

Lemma 17. *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*

$$\widetilde{H}_{j,k} = \text{tr}[\widetilde{S}^{-1} A_j \widetilde{S}^{-1} A_k] \qquad \textit{and} \qquad H_{j,k} = \text{tr}[S^{-1} A_j S^{-1} A_k].$$

Then both $\widetilde{H}$ and H are positive semidefinite. For any accuracy parameter $\alpha_S \geq 1$, if

$$\alpha_S^{-1} \cdot S \preceq \widetilde{S} \preceq \alpha_S \cdot S,$$

then we have that

$$\alpha_S^{-2} \cdot H \preceq \widetilde{H} \preceq \alpha_S^2 \cdot H.$$

Proof. For any vector $v \in \mathbb{R}^n$, we define $A(v) = \sum_{i=1}^m v_i A_i$. We can rewrite $v^\top H v$ as follows.

$$v^\top H v = \sum_{i=1}^m \sum_{j=1}^m v_i v_j H_{i,j} = \sum_{i=1}^m \sum_{j=1}^m v_i v_j \text{tr}[S^{-1} A_i S^{-1} A_j] = \text{tr}[S^{-1/2} A(v) S^{-1} A(v) S^{-1/2}]. \tag{3.30}$$

Similarly, we have

$$v^\top \widetilde{H} v = \mathrm{tr}[\widetilde{S}^{-1/2} A(v) \widetilde{S}^{-1} A(v) \widetilde{S}^{-1/2}]. \tag{3.31}$$

As the RHS of (3.30) and (3.31) 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 3.2), which gives the following inequalities

$$\begin{aligned} \mathrm{tr}[S^{-1/2} A(v) S^{-1} A(v) S^{-1/2}] &\le \alpha_S \cdot \mathrm{tr}[S^{-1/2} A(v) \widetilde{S}^{-1} A(v) S^{-1/2}] \\ &\le \alpha_S^2 \cdot \mathrm{tr}[\widetilde{S}^{-1/2} A(v) \widetilde{S}^{-1} A(v) \widetilde{S}^{-1/2}], \end{aligned} \tag{3.32}$$

where the first inequality follows from viewing $\mathrm{tr}[S^{-1/2} A(v) S^{-1} A(v) S^{-1/2}]$ 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

$$\mathrm{tr}[S^{-1/2} A(v) S^{-1} A(v) S^{-1/2}] \ge \alpha_S^{-2} \cdot \mathrm{tr}[\widetilde{S}^{-1/2} A(v) \widetilde{S}^{-1} A(v) \widetilde{S}^{-1/2}]. \tag{3.33}$$

Combining (3.32) and (3.33) with (3.30) and (3.31) along with the fact that v can be any arbitrary n-dimensional vector finishes the proof of the lemma. □

Approximate Hessian maintenance

Lemma 18. *In each iteration of Algorithm 1, for $\alpha_H = 1.03$, the approximate Hessian $\widetilde{H}(y)$ satisfies that*

$$\alpha_H^{-1} H(y) \preceq \widetilde{H}(y) \preceq \alpha_H \cdot H(y).$$

Proof. By Lemma 16, given as input two positive definite matrices S_{new} and $\widetilde{S}$, Algorithm 2 outputs a matrix $\widetilde{S}_{\mathrm{new}}$ such that

$$\|S_{\mathrm{new}}^{-1/2} \widetilde{S}_{\mathrm{new}} S_{\mathrm{new}}^{-1/2} - I\|_{\mathrm{op}} \le \epsilon_S,$$

where $\epsilon_S = 0.01$ as in Algorithm 2. By definition of operator norm, this implies that in each iteration of Algorithm 1, we have, for $\alpha_S = 1.011$,

$$\alpha_S^{-1} \cdot S \preceq \widetilde{S} \preceq \alpha_S \cdot S.$$

The statement of this lemma then follows from Lemma 17. □

Invariance of Newton step size

The following lemma is standard in the theory of interior point methods (e.g. see [102]).

Lemma 19 (Invariance of Newton step [102]). *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 $\widetilde{H} \in \mathbb{S}_{>0}^{n\times n}$ 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

$$g_{\eta_{\text{new}}}(y_{\text{new}})^\top H(y_{\text{new}})^{-1} g_{\eta_{\text{new}}}(y_{\text{new}}) \le \epsilon_N^2.$$

Approximate optimality

The following lemma is also standard in interior point method.

Lemma 20 (Approximate optimality [102]). *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 Newton step size:*

$$g_\eta(y)^\top H(y)^{-1} g_\eta(y) \le \epsilon_N^2.$$

Let y^* be an optimal solution to the dual formulation (3.2). Then we have

$$b^\top y \le b^\top y^* + \frac{n}{\eta} \cdot (1 + 2\epsilon_N).$$

3.6 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 3.7.

Theorem 14 (Rank inequality). *Let $r_0 = n$ and r_i be the rank of the update to the approximate slack matrix $\widetilde{S}$ when calling Algorithm 2 in iteration i of Algorithm 1. Then, over T iterations of Algorithm 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 14. To this end, we define the "error" matrix $Z \in \mathbb{R}^{n\times n}$ as follows

$$Z = S^{-1/2} \widetilde{S} S^{-1/2} - I \tag{3.34}$$

and the potential function $\Phi : \mathbb{R}^{n\times n} \to \mathbb{R}$

$$\Phi(Z) = \sum_{i=1}^{n} \frac{|\lambda(Z)|_{[i]}}{\sqrt{i}}, \tag{3.35}$$

where $|\lambda(Z)|_{[i]}$ denotes the i'th entry in the list of absolute eigenvalues of Z sorted in descending order. The following lemma bounds, from above, the change in the potential described by Equation (3.35), when S is updated to S_{new}.

Lemma 21 (Potential change when S changes)**.** *Suppose matrices S, S_{new} and $\widetilde{S}$ satisfy the inequalities*

$$\|S^{-1/2}S_{\text{new}}S^{-1/2} - I\|_F \leq 0.02 \qquad \textit{and} \qquad \|S^{-1/2}\widetilde{S}S^{-1/2} - I\|_{\text{op}} \leq 0.01. \tag{3.36}$$

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

$$\Phi(Z_{\text{mid}}) - \Phi(Z) \leq \sqrt{\log n}.$$

Proof. Our goal is to prove

$$\sum_{i=1}^{n}(\lambda(Z)_{[i]} - \lambda(Z_{\text{mid}})_{[i]})^2 \leq 10^{-3}. \tag{3.37}$$

We first show that the lemma statement is implied by (3.37). We rearrange the order of the eigenvalues of Z_{mid} and Z so that $\lambda(Z_{\text{mid}})_i$ and $\lambda(Z)_i$ are 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 (3.37) is equivalent to $\|\Delta\|_2^2 \leq 10^{-3}$. Let τ be the descending order of the magnitudes of eigenvalues of Z_{mid}, i.e. $|\lambda(Z_{\text{mid}})_{\tau(1)}| \geq \cdots \geq |\lambda(Z_{\text{mid}})_{\tau(n)}|$. The potential change $\Phi(Z_{\text{mid}}) - \Phi(Z)$ can be upper bounded as

$$\begin{aligned}
\Phi(Z_{\text{mid}}) &= \sum_{i=1}^{n} \frac{1}{\sqrt{i}}|\lambda(Z_{\text{mid}})_{\tau(i)}| \\
&\leq \sum_{i=1}^{n} \left(\frac{1}{\sqrt{i}}|\lambda(Z)_{\tau(i)}| + \frac{1}{\sqrt{i}}|\Delta_{\tau(i)}|\right) \\
&\leq \Phi(Z) + \left(\sum_{i=1}^{n}\frac{1}{i}\right)^{1/2}\left(\sum_{i=1}^{n}|\Delta_i|^2\right)^{1/2} \\
&\leq \Phi(Z) + \sqrt{\log n},
\end{aligned}$$

where the third line follows from

$$\sum_i \frac{1}{\sqrt{i}}|\lambda(Z)_{\tau(i)}| \leq \sum_i \frac{1}{\sqrt{i}}|\lambda(Z)|_{[i]}$$

and Cauchy-Schwarz inequality. This proves the lemma.

The remaining part of this proof is therefore devoted to proving (3.37). 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.

$$\begin{aligned}
Z_{\text{mid}} &= (S_{\text{new}})^{-1/2}\widetilde{S}(S_{\text{new}})^{-1/2} - I \\
&= (S_{\text{new}})^{-1/2}S^{1/2}S^{-1/2}\widetilde{S}S^{-1/2}S^{1/2}(S_{\text{new}})^{-1/2} - I \\
&= WZW^\top + WW^\top - I.
\end{aligned} \tag{3.38}$$

Let $\lambda(M)_{[i]}$ denote the i'th (ordered) eigenvalue of a matrix M. We then have

$$\begin{aligned}\sum_{i=1}^{n}(\lambda(Z_{\text{mid}})_{[i]}-\lambda(WZW^\top)_{[i]})^2 &\le \|Z_{\text{mid}}-WZW^\top\|_F^2\\ &= \|W^\top W-I\|_F^2, \end{aligned}\tag{3.39}$$

where the first inequality is by Fact 3 (which is applicable here because Z_{mid} and $WZW^\top$ are both normal matrices) and the second step is by (3.38). Denote the eigenvalues of $S^{-1/2}S_{\text{new}}S^{-1/2}$ by $\{\nu_i\}_{i=1}^n$. Then the first assumption in (3.36) implies that $\sum_{i\in[n]}(\nu_i-1)^2 \le 4\times 10^{-4}$. It follows that

$$\|W^\top W-I\|_F^2 = \|S^{1/2}S_{\text{new}}^{-1}S^{1/2}-I\|_F^2 = \sum_{i\in[n]}(1/\nu_i-1)^2 \le 5\times 10^{-4}, \tag{3.40}$$

where the last inequality is because the first assumption from (3.36) implies $\nu_i \ge 0.98$ for all $i\in[n]$. Plugging (3.40) into the right hand side of (3.39), we have

$$\sum_{i=1}^{n}(\lambda(Z_{\text{mid}})_{[i]}-\lambda(WZW^\top)_{[i]})^2 \le 5\times 10^{-4}. \tag{3.41}$$

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, (3.40) is then equivalent to

$$\|\Sigma^2-I\|_F = \sum_{i=1}^{n}(\sigma_i^2-1)^2 \le 5\times 10^{-4}. \tag{3.42}$$

Since U and V are unitary, the matrix $WZW^\top = U\Sigma V^\top ZV\Sigma U^\top$ is similar to $\Sigma V^\top ZV\Sigma$, and the matrix $Z'=V^\top ZV$ is similar to Z. Therefore,

$$\begin{aligned}\sum_{i=1}^{n}(\lambda(WZW^\top)_{[i]}-\lambda(Z)_{[i]})^2 &= \sum_{i=1}^{n}(\lambda(\Sigma Z'\Sigma)_{[i]}-\lambda(Z')_{[i]})^2\\ &\le \|\Sigma Z'\Sigma-Z'\|_F^2, \end{aligned}\tag{3.43}$$

where the last inequality is by Fact 3. We rewrite the Frobenius norm as

$$\begin{aligned}\|\Sigma Z'\Sigma-Z'\|_F &= \|(\Sigma-I)Z'(\Sigma-I)+(\Sigma-I)Z'+Z'(\Sigma-I)\|_F\\ &\le \|(\Sigma-I)Z'(\Sigma-I)\|_F+2\|(\Sigma-I)Z'\|_F. \end{aligned}\tag{3.44}$$

The first term can be bounded as:

$$\begin{aligned}\|(\Sigma - I)Z'(\Sigma - I)\|_F^2 &= \operatorname{tr}[(\Sigma - I)Z'(\Sigma - I)^2 Z'(\Sigma - I)] \\ &\le \operatorname{tr}[(\Sigma - I)^4 \cdot (Z')^2] \\ &\le 0.01^2 \cdot \operatorname{tr}[(\Sigma - I)^4] \\ &= \sum_{i=1}^n (\sigma_i - 1)^4 \\ &\le 5 \times 10^{-8}, \end{aligned} \tag{3.45}$$

The first inequality above uses Fact 1, the second used the observation that $\|Z'\|_{\text{op}} = \|Z\|_{\text{op}} \le 0.01$, and the last inequality follows from (3.42) 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

$$\begin{aligned}\|(\Sigma - I)Z'\|_F^2 &= \operatorname{tr}[(\Sigma - I)(Z')^2(\Sigma - I)] \\ &\le \operatorname{tr}[(\Sigma - I)^2 (Z')^2] \\ &\le 0.01^2 \cdot \operatorname{tr}[(\Sigma - I)^2] \ \le\ 10^{-7}. \end{aligned} \tag{3.46}$$

It follows from (3.43), (3.44) and (3.46) that

$$\sum_{i=1}^n (\lambda(WZW^\top)_{[i]} - \lambda(Z)_{[i]})^2 \le 10^{-6}. \tag{3.47}$$

Combining (3.41) and (3.47), we get that $\sum_{i=1}^n (\lambda(Z)_{[i]} - \lambda(Z_{\text{mid}})_{[i]})^2 \le 10^{-3}$ which establishes (3.37). This completes the proof of the lemma. $\square$

Lemma 22 (Potential change when $\widetilde{S}$ changes)**.** *Given positive definite matrices $S_{\text{new}}, \widetilde{S} \in \mathbb{S}^n_{>0}$, let $\widetilde{S}_{\text{new}}$ and r be generated during the run of Algorithm 2 when the inputs are S_{new} and $\widetilde{S}$. Define the matrices $Z_{\text{mid}} = (S_{\text{new}})^{-1/2}\widetilde{S}(S_{\text{new}})^{-1/2} - I$ and $Z_{\text{new}} = (S_{\text{new}})^{-1/2}\widetilde{S}_{\text{new}}(S_{\text{new}})^{-1/2} - I$. Then we have*

$$\Phi(Z_{\text{mid}}) - \Phi(Z_{\text{new}}) \ge \frac{10^{-4}}{\log n}\sqrt{r}.$$

Proof. The setup of the lemma considers the eigenvalues of Z when $\widetilde{S}$ changes. For the sake of notational convenience, we define $y = |\lambda(Z_{\text{mid}})|$, the vector of absolute values of eigenvalues of $Z_{\text{mid}} = S_{\text{new}}^{-1/2}\widetilde{S}S_{\text{new}}^{-1/2} - I$. Recall from Table 3.5.1 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 $\Phi(Z_{\text{mid}}) - \Phi(Z_{\text{new}}) \ge \epsilon_S\sqrt{n} \ge \frac{10^{-4}}{\log n}\sqrt{r}$.

- Case (b). There exists a minimum index $i \leq n/2$ such that $y_{[2j]} < \epsilon_S$ holds for all j in the range $i \leq j \leq n/2$. In this case, for all j in the above range, we have that $y_{[2j]} \geq (1 - 1/10 \log n) y_{[j]}$. In particular, picking $j = i, 2i, \cdots$ gives

  $$y_{[n]} \geq y_{[i]} \cdot (1 - 1/(10 \log n))^{\lceil \log n \rceil} \geq \epsilon_S/10.$$

  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 $\Phi(Z_{\mathrm{mid}}) - \Phi(Z_{\mathrm{new}}) \geq \epsilon_S/10 \cdot \sqrt{n} \geq \frac{10^{-4}}{\log n} \cdot \sqrt{r}$.

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 $y^{\mathrm{new}}_{[i]}$ is set to 0 for each $i \in [2r]$, we have $y^{\mathrm{new}}_{[i]} = y_{[i+2r]} \leq y_{[i]}$. Further, $y_{[2r]} < (1 - 1/10 \log n) y_{[r]}$ implies that for each $i \in [r]$, we have

$$y_{[i]} - y^{\mathrm{new}}_{[i]} \geq \frac{1}{10 \log n} \cdot y_{[r]} \geq \frac{10^{-2} \epsilon_S}{\log n} = \frac{10^{-4}}{\log n},$$

where $\epsilon_S = 0.01$ by Table 3.5.1. Therefore, we can bound, from below, the decrease in potential function as

$$\Phi(Z_{\mathrm{mid}}) - \Phi(Z_{\mathrm{new}}) \geq \sum_{i=1}^{r} \frac{y_{[i]} - y^{\mathrm{new}}_{[i]}}{\sqrt{i}} \geq \frac{10^{-4}}{\log n} \sqrt{r}.$$

This finishes the proof of the lemma. □

Proof of Theorem 14. Recall the definition of the potential function in (3.35) 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 1. Define $Z^{(i)} = (S^{(i)})^{-1/2} \widetilde{S}^{(i)} (S^{(i)})^{-1/2} - I$ and $Z^{(i)}_{\mathrm{mid}} = (S^{(i+1)})^{-1/2} \widetilde{S}^{(i)} (S^{(i+1)})^{-1/2} - I$. By Lemma 21, we have that

$$\Phi(Z^{(i)}_{\mathrm{mid}}) - \Phi(Z^{(i)}) \leq \sqrt{\log n}.$$

From Lemma 22, we have the following potential decrease:

$$\Phi(Z^{(i)}_{\mathrm{mid}}) - \Phi(Z^{(i+1)}) \geq \frac{10^{-4}}{\log n} \sqrt{r_i}.$$

These together imply that

$$\Phi(Z^{(i+1)}) - \Phi(Z^{(i)}) \leq \sqrt{\log n} - \frac{10^{-4}}{\log n}\sqrt{r_i}. \tag{3.48}$$

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 (3.48) over all T iterations. □

3.7 Runtime Analysis

Our main result of this section is the following bound on the runtime of Algorithm 1.

Theorem 15 (Runtime bound). *The total runtime of Algorithm 1 for solving an SDP with variable size $n \times n$ and m constraints is at most $O^*\left(\sqrt{n}\,(mn^2 + \max(m,n)^\omega)\right)$, where ω is the matrix multiplication exponent as defined in Definition 4.*

To prove Theorem 15, we first upper bound the runtime in terms of fast rectangular matrix multiplication times. The iteration complexity of Algorithm 1 is $T = \widetilde{O}(\sqrt{n})$.

Lemma 23 (Total cost). *The total runtime of Algorithm 1 over T iterations is upper bounded as*

$$\mathcal{T}_{\text{Total}} \leq O^*\left(\min\left(n \cdot \text{nnz}(A), mn^{2.5}\right) + \sqrt{n}\max(m,n)^\omega \right. \tag{3.49}$$

$$\left. + \sum_{i=0}^{T} (\mathcal{T}_{\text{mat}}(n, mr_i, n) + \mathcal{T}_{\text{mat}}(m, nr_i, m)\right), \tag{3.50}$$

where $\text{nnz}(A)$ *is the total number of non-zero entries in all the constraint matrices, r_i, as defined in Theorem 14, 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 Definitions 4 and 3, respectively.*

Remark 1. *A more careful analysis can improve the first term in the RHS of* (3.49) *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 work, however, we will only need the simpler bound given in Lemma 23.*

Proof. The total runtime of Algorithm 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 11 - 15.
- **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

$$\widetilde{H}_{j,k} = \mathrm{tr}\Big[(\widetilde{S}^{-1/2}A_j\widetilde{S}^{-1/2})(\widetilde{S}^{-1/2}A_k\widetilde{S}^{-1/2})\Big].$$

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:

$$\mathcal{T}_{\widetilde{S}^{-1/2}A_j\widetilde{S}^{-1/2} \text{ for all } j\in[m]} \le O(\mathcal{T}_{\mathrm{mat}}(n, mn, n)). \tag{3.51}$$

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,

$$\mathcal{T}_{\text{computing } \widetilde{H} \text{ from } B} \le O(\mathcal{T}_{\mathrm{mat}}(m, n^2, m)). \tag{3.52}$$

Combining (3.51), (3.52), and the initial cost of inverting $\widetilde{S}$ gives the following cost for computing $\widetilde{H}$ for the first iteration:

$$\mathcal{T}_{\text{part 1a}} \le O^*(\mathcal{T}_{\mathrm{mat}}(m, n^2, m) + \mathcal{T}_{\mathrm{mat}}(n, mn, n) + n^\omega). \tag{3.53}$$

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 15 of Algorithm 2). If the update from $\widetilde{S}$ to $\widetilde{S}_{\mathrm{new}}$ has rank r_i, Fact 4 implies that we can compute, in time $O(n^{\omega+o(1)})$, the $n \times r_i$ matrices V_+ and V_- satisfying $\widetilde{S}_{\mathrm{new}}^{-1} = \widetilde{S}^{-1} + V_+V_+^\top - V_-V_-^\top$. The cost of updating $\widetilde{H}$ is then dominated by the computation of $\mathrm{tr}[\widetilde{S}^{-1/2}A_jVV^\top A_k\widetilde{S}^{-1/2}]$, where $V \in \mathbb{R}^{n\times r_i}$ is either V_+ or V_-. We note that

$$\mathcal{T}_{A_jV \text{ for all } j\in[m]} \le O^*\left(\min\left(r_i \cdot \mathrm{nnz}(A), mn^2r_i^{\omega-2+o(1)}\right)\right), \tag{3.54}$$

where $\mathrm{nnz}(A)$ 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 3, costs $\mathcal{T}_{\mathrm{mat}}(n, mr_i, n)$. This, together with (3.54), gives

$$\mathcal{T}_{\widetilde{S}^{-1/2}A_jV \text{ for all } j\,\in\,[m]} \le O^*\left(\mathcal{T}_{\mathrm{mat}}(n, mr_i, n) + \min\left(r_i \cdot \mathrm{nnz}(A), mn^2r_i^{\omega-2+o(1)}\right)\right). \tag{3.55}$$

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 (3.55) gives the following run time for one update to the approximate Hessian:

$$\mathcal{T}_{\text{rank } r_i \text{ Hess. update}} \le O^*\left(\mathcal{T}_{\text{mat}}(n, mr_i, n) + \min\left(r_i \cdot \text{nnz}(A), mn^2 r_i^{\omega-2}\right) + \mathcal{T}_{\text{mat}}(m, nr_i, m) + n^\omega\right) \tag{3.56}$$

Using this bound over all $T = \widetilde{O}(\sqrt{n})$ iterations, and applying $\sum_{i=0}^{T}\sqrt{r_i} \le \widetilde{O}(\sqrt{n})$ from Theorem 14, gives

$$\mathcal{T}_{\text{part 1b}} \le O^*\left(\min(n \cdot \text{nnz}(A), mn^{2.5}) + \sqrt{n} \cdot n^\omega + \sum_{i=1}^{T}(\mathcal{T}_{\text{mat}}(n, mr_i, n) + \mathcal{T}_{\text{mat}}(m, nr_i, m))\right). \tag{3.57}$$

Combining Part 1a and 1b.

Combining (3.53) and (3.57), we have

$$\begin{aligned}\mathcal{T}_{\text{part 1}} &\le \mathcal{T}_{\text{part 1a}} + \mathcal{T}_{\text{part 1b}} \\ &\le O^*\left(\min(n \cdot \text{nnz}(A), mn^{2.5}) + \sqrt{n} \cdot n^\omega + \sum_{i=0}^{T}(\mathcal{T}_{\text{mat}}(n, mr_i, n) + \mathcal{T}_{\text{mat}}(m, nr_i, m))\right),\end{aligned} \tag{3.58}$$

where we incorporated the bound from (3.53) into the $i = 0$ case.

Part 2.

Observe that there are four operations performed in Algorithm 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 $g_\eta(y)_i = \eta b_i - \text{tr}[S^{-1}A_i]$. 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 16.

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

$$\begin{aligned}\mathcal{T}_{\text{part 2}} &\leq \mathcal{T}_{\text{part 2a}} + \mathcal{T}_{\text{part 2b}} + \mathcal{T}_{\text{part 2c}} + \mathcal{T}_{\text{part 2d}} \\ &\leq O^*(\sqrt{n}(\text{nnz}(A) + \max(m,n)^\omega)). \end{aligned} \tag{3.59}$$

Combining Part 1 and Part 2.

Combining (3.58) and (3.59) and using $r_0 = n$ finishes the proof of the lemma.

$$\begin{aligned}\mathcal{T}_{\text{total}} &\leq \mathcal{T}_{\text{part 1}} + \mathcal{T}_{\text{part 2}} \\ &\leq O^*\left(\min\left(n \cdot \text{nnz}(A), mn^{2.5}\right) + \sqrt{n}\max(m,n)^\omega \right.\\ &\quad \left. + \sum_{i=0}^{T} \left(\mathcal{T}_{\text{mat}}(n, mr_i, n) + \mathcal{T}_{\text{mat}}(m, nr_i, m)\right)\right)\end{aligned}$$

□

Lemma 24. *Let $\mathcal{T}_{\text{mat}}$ be as defined in Definition 3. 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

$$\sum_{i=1}^{T} \mathcal{T}_{\text{mat}}(n, mr_i, n) \leq O^*(\sqrt{n}\max(m^\omega, n^\omega) + \mathcal{T}_{\text{mat}}(n, mn, n)).$$

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

$$\mathcal{T}_{\text{mat}}(m, nr_i, m) = \mathcal{T}_{\text{mat}}(n^a, n^{1+b_i}, n^a) = n^{a\omega((1+b_i)/a)+o(1)}. \tag{3.60}$$

For each number $k \in \{0, 1, \cdots, \log n\}$, define the set of iterations

$$I_k = \{i \in [T] \ : \ 2^k \leq r_i \leq 2^{k+1}\}.$$

Then our assumption on the sequence $\{r_1, \cdots, r_T\}$ can be expressed as $\sum_{k=0}^{\log n} |I_k| \cdot 2^{k/2} \leq O(T \log^{1.5} n)$. This implies that for each $k\{0, 1, \cdots, \log n\}$, we have $|I_k| \leq O(T \log^{1.5} n / 2^{k/2})$. Next, taking the summation of Eq. (3.60) over all $i \in [T]$, we have

$$\begin{aligned}
\sum_{i=1}^{T} \mathcal{T}_{\mathrm{mat}}(m, nr_i, m) &= \sum_{i=1}^{T} n^{a \cdot \omega((1+b_i)/a)} \\
&= \sum_{k=0}^{\log n} \sum_{i \in I_k} n^{a \cdot \omega((1+b_i)/a)} \\
&\leq O(\log n) \cdot \max_k \max_{i \in I_k} \frac{T \log^{1.5} n}{2^{k/2}} \cdot n^{a \cdot \omega((1+b_i)/a)} \\
&\leq \widetilde{O}(1) \cdot \max_k \max_{2^k \leq n^{b_i} \leq 2^{k+1}} \frac{\sqrt{n}}{2^{k/2}} \cdot n^{a \cdot \omega((1+b_i)/a)} \\
&\leq \widetilde{O}(1) \cdot \max_{b_i \in [0,1]} n^{1/2 - b_i/2 + a \cdot \omega((1+b_i)/a)},
\end{aligned}$$

where the fourth step follows from $T = \widetilde{O}(\sqrt{n})$. To bound the exponent on n above, we define the function g,

$$g(b_i) = 1/2 - b_i/2 + a \cdot \omega((1 + b_i)/a). \tag{3.61}$$

This function is convex in b_i due to the convexity of the function ω (Lemma 11). 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 $a \geq 1$, then we have

$$g(0) = 1/2 + a \cdot \omega(1/a) \leq 1/2 + a\omega(1) = 1/2 + a\omega$$

Case 1b. If $a \in (0, 1)$, then we define $k = 1/a > 1$. It follows from Lemma 10 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.

$$\max_{b_i \in [0,1]} n^{1/2 - b_i + a \cdot \omega((1+b_i)/a)} \leq \sqrt{n} \max(m^{\omega}, n^{\omega}) + n^{a \cdot \omega(2/a)}.$$

$\square$

Proof of Theorem 15. In light of Lemma 24, the upper bound on runtime given in Lemma 23 can be written as

$$\mathcal{T}_{\text{Total}} \le O^*\left(\min\left\{n\cdot \text{nnz}(A), mn^{2.5}\right\} + \sqrt{n}\max(m,n)^{\omega} + \mathcal{T}_{\text{mat}}(n, mn, n) + \mathcal{T}_{\text{mat}}(m, n^2, m)\right). \tag{3.62}$$

Combining this with 14, we have the following upper bound on the total runtime of Algorithm 1:

$$\begin{aligned}\mathcal{T}_{\text{Total}} &\le O^*\left(\min\left\{n\cdot \text{nnz}(A), mn^{2.5}\right\} + \sqrt{n}\max(m,n)^{\omega} + \sqrt{n}\left(mn^2 + m^{\omega}\right)\right)\\ &\le O^*\left(\sqrt{n}\left(mn^2 + \max(m,n)^{\omega}\right)\right).\end{aligned}$$

This finishes the proof of the theorem. □

3.8 Comparison with Cutting Plane Method

In this section, we prove Theorem 10, restated below.

Theorem 10 (Comparison with Cutting Plane Method)**.** *When $m \ge n$, there is an interior point method that solves an SDP with $n \times n$ matrices, m constraints, and* nnz(A) *input size, faster than the current best cutting plane method [83, 62], over all regimes of* nnz(A).

Remark 2. *In the dense case with* nnz$(A) = \Theta(mn^2)$*, Algorithm 1 is faster than the cutting plane method whenever $m \ge \sqrt{n}$.*

Proof of Theorem 10. Recall that the current best runtime of the cutting plane method for solving an SDP (3.1) is $\mathcal{T}_{\text{CP}} = O^*(m \cdot \text{nnz}(A) + mn^{2.372927} + m^3)$ [83, 62], where 2.372927 is the current best upper bound on the exponent of matrix multiplication ω. By Lemma 23 and 24, we have the following upper bound on the total runtime of Algorithm 1:

$$\mathcal{T}_{\text{Total}} \le O^*\left(\min\left\{n\cdot \text{nnz}(A), mn^{2.5}\right\} + \sqrt{n}\max(m,n)^{\omega} + \mathcal{T}_{\text{mat}}(n, mn, n) + \mathcal{T}_{\text{mat}}(m, n^2, m)\right)$$

Since $m \ge n$ by assumption, Lemma 13 and 13 further simplify the runtime to

$$\mathcal{T}_{\text{Total}} \le O^*\left(\min\left\{n\cdot \text{nnz}(A), mn^{2.5}\right\} + \sqrt{n}m^{\omega} + \mathcal{T}_{\text{mat}}(m, n^2, m)\right) \tag{3.63}$$

Note that $\min\left\{n\cdot \text{nnz}(A), mn^{2.5}\right\} \le m\cdot\text{nnz}(A) \le O(\mathcal{T}_{\text{CP}})$ and that $\sqrt{n}m^{\omega} = o(m^3) \le o(\mathcal{T}_{\text{CP}})$ since $m \ge n$. Furthermore, Lemma 15 states that $\mathcal{T}_{\text{mat}}(m, n^2, m) = o(m^3 + mn^{2.37}) \le o(\mathcal{T}_{\text{CP}})$. Since each term on the RHS of (3.63) is upper bounded by $\mathcal{T}_{\text{CP}}$, we make the stated conclusion. □

3.9 Initialization

Lemma 25 (Initialization). *Consider a semidefinite program as in Definition 2 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 $\|X\|_{\mathrm{op}} \leq R$.*
2. *Lipschitz objective: $\|C\|_{\mathrm{op}} \leq L$.*

For any $0 < \delta \leq 1$, the following modified *semidefinite program*

$$\max_{\overline{X}\succeq 0} \ \langle \overline{C}, \overline{X}\rangle \quad \text{s.t. } \langle \overline{A}_i, \overline{X}\rangle = \overline{b}_i, \forall i \in [m+1],$$

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:*

$$\begin{aligned} \langle C, \widehat{X}\rangle &\geq \langle C, X^*\rangle - LR \cdot \delta, \\ \widehat{X} &\succeq 0, \\ \sum_{i\in[m]} \left| \langle A_i, \widehat{X}\rangle - b_i \right| &\leq 4n\delta \cdot (R \sum_{i\in[m]} \|A_i\|_1 + \|b\|_1), \end{aligned}$$

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 $\overline{X} \in \mathbb{R}^{(n+2)\times(n+2)}$

$$\overline{X} = \begin{bmatrix} \frac{1}{R}X & 0_n & 0_n \\ 0_n^\top & n+1-\frac{1}{R}\mathrm{tr}[X] & 0 \\ 0_n^\top & 0 & 0 \end{bmatrix}.$$

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 $\widehat{X} \succeq 0$. Notice that

$$\frac{\delta}{L} \cdot \langle C, \overline{X}_{[n]\times[n]} \rangle - \theta = \langle \overline{C}, \overline{X} \rangle \geq \overline{\mathsf{OPT}} - \delta^2 \geq \frac{\delta}{LR} \cdot \mathsf{OPT} - \delta^2. \tag{3.64}$$

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 (3.64). By matrix Hölder inequality, we have

$$\begin{aligned} \frac{\delta}{L} \cdot \langle C, \overline{X}_{[n]\times[n]} \rangle &\leq \frac{\delta}{L} \cdot \|C\|_{\mathrm{op}} \cdot \mathrm{tr}\left[\overline{X}_{[n]\times[n]}\right] \\ &\leq \frac{\delta}{L} \cdot \|C\|_{\mathrm{op}} \cdot \langle \overline{A}_{m+1}, \overline{X} \rangle \\ &\leq (n+1)\delta, \end{aligned}$$

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

$$\theta \leq \frac{\delta}{L} \cdot \langle C, \overline{X}_{[n]\times[n]} \rangle + \delta^2 - \frac{\delta}{LR} \cdot \mathsf{OPT} \leq (2n+1)\delta + \delta^2 \leq 4n\delta, \tag{3.65}$$

where the first step follows from (3.64), 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 (3.65). Summing the above inequality up over all $i \in [m]$ finishes the proof of the lemma. □

3.10 Matrix Multiplication: A Tensor Approach

The main goal of this section is to rederive, using tensors, some of the technical results from Section 3.3. In particular, we use tensors to derive upper bounds on the time to perform the following two rectangular matrix multiplication tasks (Lemma 34 and 35):

- 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 $mn \times n$.

Our hope is that these techniques will eventually be useful in further improving the results of this work.

Exponent of matrix multiplication

We recall two definitions to describe the cost of certain fundamental matrix operations, along with their properties.

Definition 5. *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 6. *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 exponent of matrix multiplication (in other words, the cost of multiplying two $n \times n$ matrices is n^{ω}), and let α denote the dual exponent of matrix multiplication. Thus, we have $\omega(1) = \omega$ and $\omega(\alpha) = 2$.*

Lemma 26 ([46])**.** *We have :*

- $\omega = \omega(1) \leq 2.372927$,
- $\omega(1.5) \leq 2.79654$,
- $\omega(1.75) \leq 3.02159$,
- $\omega(2) \leq 3.251640$.

Lemma 27 ([31, 25])**.** *For any three positive integers n, m, r, we have*

$$\mathcal{T}_{\mathrm{mat}}(n, r, m) = O(\mathcal{T}_{\mathrm{mat}}(n, m, r)) = O(\mathcal{T}_{\mathrm{mat}}(m, n, r)).$$

Matrix multiplication tensor

The rank of a tensor T, denoted as $R(T)$, is the minimum number of simple tensors that sum up to T. For any two tensors $S = (S_{i,j,k})_{i,j,k}$ and $T = (T_{a,b,c})_{a,b,c}$, we write $S \le T$ if there exist three matrices A, B and C (of appropriate sizes) such that $S_{i,j,k} = \sum_{a,b,c} A_{i,a} B_{j,b} C_{k,c} T_{a,b,c}$ for all i, j, k. For any i, j, k, denote $e_{i,j,k}$ the tensor with 1 in the (i, j, k)-th entry, and 0 elsewhere.

Definition 7 (Matrix-multiplication tensor)**.** *For any three positive integers a, b, c, we define*

$$\langle a, b, c\rangle := \sum_{i\in[a]} \sum_{j\in[b]} \sum_{k\in[c]} e_{i(b-1)+j, j(c-1)+k, k(a-1)+i}$$

to be the matrix-multiplication tensor corresponding to multiplying a matrix of size $a \times b$ with one of size $b \times c$.

It's not hard to show that for any n_i and m_i where $i = 1, 2, 3$, we have

$$\langle n_1, n_2, n_3\rangle \otimes \langle m_1, m_2, m_3\rangle = \langle n_1m_1, n_2m_2, n_3m_3\rangle.$$

Let $\langle n\rangle = \sum_{i\in[n]} e_{i,i,i}$ be the identity tensor. For any three tensors S, T_1 and T_2, if $T_1 \le T_2$, then we have

$$S \otimes T_1 \le S \otimes T_2.$$

Lemma 28 (Monotonicity of tensor rank, [116])**.** *Tensor rank is monotone under the relation $\le$, i.e. if $T_1 \le T_2$, then we have*

$$R(T_1) \le R(T_2).$$

Lemma 29 (Sub-multiplicity of tensor rank, [116])**.** *For any tensors T_1 and T_2, we have*

$$R(T_1 \otimes T_2) \le R(T_1) \cdot R(T_2).$$

Lemma 30. *The tensor rank of a matrix multiplication tensor is equal to the cost of multiplying the two corresponding sized matrices up to some constant factor, i.e.,*

$$R(\langle a, b, c\rangle) = \Theta(\mathcal{T}_{\mathrm{mat}}(a, b, c)).$$

Implication of matrix multiplication technique

Lemma 31 (Sub-linearity)**.** *For any $p \ge q \ge 1$, we have*

$$\omega(p) \le p - q + \omega(q).$$

Proof. We have

$$\langle n, n^p, n\rangle = \langle n, n^q, n\rangle \otimes \langle 1, n^{p-q}, 1\rangle.$$

Applying tensor rank on both sides

$$\begin{aligned} R(\langle n, n^p, n\rangle) &= R(\langle n, n^q, n\rangle \otimes \langle 1, n^{p-q}, 1\rangle) \\ &\leq R(\langle n, n^q, n\rangle) \cdot R(\langle 1, n^{p-q}, 1\rangle), \end{aligned}$$

where the last line follows from Lemma 29. Applying Lemma 30, we have

$$\mathcal{T}_{\mathrm{mat}}(n, n^p, n) \leq O(1) \cdot \mathcal{T}_{\mathrm{mat}}(n, n^q, n) \cdot n^{p-q}$$

Using the definition of $\omega(p)$, we have

$$n^{\omega(p)+o(1)} \leq O(1) \cdot n^{\omega(q)+o(1)} \cdot n^{p-q}.$$

Comparing the exponent on both sides completes the proof. □

The next lemma establishes the convexity of $\omega(k)$ as a function of k.

Lemma 32 (Convexity of $\omega(k)$)**.** *The fast rectangular matrix multiplication time exponent $\omega(k)$ as defined in Definition 6 is convex in k.*

Proof. Let $k = \alpha \cdot p + (1-\alpha) \cdot q$ for $\alpha \in (0, 1)$. We have

$$\langle n, n^k, n\rangle = \langle n^\alpha, n^{\alpha \cdot p}, n^\alpha\rangle \otimes \langle n^{1-\alpha}, n^{(1-\alpha)p}, n^{1-\alpha}\rangle.$$

Applying the tensor rank on both sides,

$$\begin{aligned} R(\langle n, n^k, n\rangle) &= R(\langle n^\alpha, n^{\alpha \cdot p}, n^\alpha\rangle \otimes \langle n^{1-\alpha}, n^{(1-\alpha)p}, n^{1-\alpha}\rangle) \\ &\leq R(\langle n^\alpha, n^{\alpha \cdot p}, n^\alpha\rangle) \cdot R(\langle n^{1-\alpha}, n^{(1-\alpha)p}, n^{1-\alpha}\rangle), \end{aligned}$$

where the last line follows from Lemma 29. By Lemma 30, we have

$$\mathcal{T}_{\mathrm{mat}}(n, n^k, n) \leq O(1) \cdot \mathcal{T}_{\mathrm{mat}}(n^\alpha, n^{\alpha p}, n^\alpha) \cdot \mathcal{T}_{\mathrm{mat}}(n^{1-\alpha}, n^{(1-\alpha)p}, n^{1-\alpha})$$

By definition of $\omega(\cdot)$, we have

$$n^{\omega(k)+o(1)} \leq O(1) \cdot n^{\alpha \cdot \omega(p)} \cdot n^{(1-\alpha)\omega(1-p)}.$$

By comparing the exponent, we know that

$$\omega(k) \leq \alpha \cdot \omega(p) + (1-\alpha) \cdot \omega(1-p).$$

□

Lemma 33. *Let $\mathcal{T}_{\mathrm{mat}}$ be defined as in Definition 5. Then for any positive integers a, b, c and k, we have*

$$\mathcal{T}_{\mathrm{mat}}(a, bk, c) \leq O(\mathcal{T}_{\mathrm{mat}}(ak, b, ck)).$$

Proof. Notice that

$$\langle 1, k, 1\rangle \leq \langle k, 1, k\rangle.$$

Therefore, we have

$$\begin{aligned}\langle a, bk, c\rangle &= \langle a, b, c\rangle \otimes \langle 1, k, 1\rangle \\ &\leq \langle a, b, c\rangle \otimes \langle k, 1, k\rangle \\ &= \langle ak, b, ck\rangle.\end{aligned}$$

It then follows from Lemma 28 that

$$R(\langle a, bk, c\rangle) \leq R(\langle ak, b, ck\rangle).$$

Finally, using Lemma 30 gives

$$\mathcal{T}_{\mathrm{mat}}(a, bk, c) \leq O(\mathcal{T}_{\mathrm{mat}}(ak, b, ck)).$$

Thus we complete the proof. □

General bound on $\mathcal{T}_{\mathrm{mat}}(n, mn, n)$ and $\mathcal{T}_{\mathrm{mat}}(m, n^2, m)$

Lemma 34. *Let $\mathcal{T}_{\mathrm{mat}}$ be defined as in Definition 5.*
If $m \geq n$, then we have

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

If $m \leq n$, then we have

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

Proof. We only prove the case of $m \geq n$, as the other case where $m < n$ is similar. This is an immediate consequence of Lemma 33 by taking $a = c = n$, $b = n^2$, 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 $\mathcal{T}_{\mathrm{mat}}(m, n^2, m)$ when $m \geq n$ and $\mathcal{T}_{\mathrm{mat}}(n, mn, n)$ when $m < n$, which is crucial to our runtime analysis.

Lemma 35. *Let $\mathcal{T}_{\mathrm{mat}}$ be defined as in Definition 5 and ω be defined as in Definition 6. Property I. We have*

$$\mathcal{T}_{\mathrm{mat}}(n, mn, n) \leq O(mn^{\omega+o(1)}).$$

Property II. We have

$$\mathcal{T}_{\mathrm{mat}}(m, n^2, m) \leq O\left(\sqrt{n}\left(mn^2 + m^{\omega}\right)\right).$$

Proof. **Property I.**

Since

$$\langle n, mn, n \rangle = \langle n, n, n \rangle \otimes \langle 1, m, 1 \rangle.$$

Applying the tensor rank on both sides, we have

$$\begin{aligned} R(\langle n, mn, n \rangle) &= R(\langle n, n, n \rangle \otimes \langle 1, m, 1 \rangle) \\ &\leq R(\langle n, n, n \rangle) \cdot R(\langle 1, m, 1 \rangle) \end{aligned}$$

Thus, we complete the proof.

Property II.

Let $m = n^a$, where $a \in (0, \infty)$. We have

$$\langle m, n^2, m \rangle = \langle n^a, (n^a)^{2/a}, n^a \rangle$$

It implies that

$$\mathcal{T}_{\mathrm{mat}}(m, n^2, m) = n^{a \cdot \omega(2/a) + o(1)}$$

The Property II is then an immediate consequence of the following inequality, which we prove next:

$$\omega(2/a) < \max(1 + 2.5/a, \omega(1) + 0.5/a) \qquad \forall a \in (0, \infty).$$

Define $b = 2/a \in (0, \infty)$. Then the above desired inequality can be expressed in terms of b as

$$\omega(b) < \max(1 + 5b/4, \omega(1) + b/4) \qquad \forall b \in (0, \infty). \tag{3.66}$$

Notice that the RHS of (3.15) 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 32, it suffices to verify (3.15) at the endpoints $b \to 0$, $b \to \infty$ and $b = b^*$. In the case where $b = \delta$ for any $\delta < 1$, (3.15) follows immediately from the observation that $\omega(\delta) < \omega(1)$. For the case $b \to \infty$, by Lemma 26 we have $\omega(2) \leq 3.252$. It then follows from Lemma 31 that for any $b > 2$, we have

$$\omega(b) \leq b - 2 + \omega(2) \leq 1 + 5b/4.$$

The final case is where $b = b^* = \omega(1) - 1$, for which (3.15) is equivalent to

$$\omega(\omega(1) - 1) < 5\omega(1)/4 - 1/4. \tag{3.67}$$

By Lemma 26, we have that $\omega(1) - 2 \in [0, 0.372927]$. Then to prove (3.67), it is sufficient to show that

$$\omega(t + 1) < 5t/4 + 9/4 \qquad \forall t \in [0, 0.372927]. \tag{3.68}$$

By the convexity of $\omega(\,\cdot\,)$ as proved in Lemma 32 and the upper bound of $\omega(2) \leq 3.251640$ in Lemma 26, we have for $k \in [1, 2]$,

$$\omega(k) \leq \omega(1) + (k - 1) \cdot (3.251640 - (t + 2)) = t + 2 + (k - 1) \cdot (1.251640 - t).$$

In particular, using this inequality for $k = t + 1$, we have

$$\begin{aligned} \omega(t + 1) - 5t/4 - 9/4 &\leq (t + 2) + t \cdot (1.251640 - t) - 5t/4 - 9/4 \\ &= -t^2 + 1.00164t - 1/4, \end{aligned}$$

which is negative on the entire interval $[0, 0.372927]$. This establishes (3.68) and finishes the proof of the lemma. $\square$

Chapter 4

An IPM Inspired Approach to Discrepancy

In this chapter, we first describe an interior point method style algorithm for the usual Spencer problem. While this approach is heavily inspired from previous algorithms and analyses of the usual Spencer problem, we remark that our "covariance control" is different than the previous approaches and unlike the previous controls which seemed intrinsic to the case of diagonal matrices/polyhedral constraint sets, our approach generalizes to non-commutative settings and we show that this suggests a natural approach to resolving matrix discrepancy problems. We first describe the usual Spencer problem and a conjectured non-commutative generalization that has been dubbed the Matrix Spencer problem.

Problem 1 (Spencer Problem)**.** *Given a universe of n elements, denoted $U = \{1, \dots, n\}$ and n subsets of U denoted $\mathcal{S} = \{S_1, \dots, S_n\}$. Our goal is to find a signing of the elements such that the imbalance/discrepancy across all subsets in $\mathcal{S}$ is small. Mathematically,*

$$\mathsf{disc}(\mathcal{S}) := \min_{x \in \{\pm 1\}^n} \max_{i=1}^{n} \left| \sum_{j \in S_i} x_j \right|$$

One can more generally consider the setup of a given matrix $A \in [-1, 1]^{n \times n}$ and ask for a signing $x \in \{\pm 1\}^n$ so that $\|Ax\|_\infty$ is small.

If we pick a signing at random, Chernoff bound combined with union bound over the n sets tells us that with high probability, the discrepancy is $O(\sqrt{n \log n})$. Spencer [114] showed that there exists a signing which achieves a discrepancy of $O(\sqrt{n})$. This demonstrates an improvement from $O(\sqrt{n \log n})$ from $O(\sqrt{n})$, which is tight.

As there are examples, like the Hadamard matrix, where such signings are exponentially rare, Spencer [114] actually conjectured that it should be hard to find such a signing in polynomial time. Starting from work of Bansal [15], however, a myriad of different algorithmic proofs have emerged [87, 104, 45, 85]. A feature common to all these proofs is the crucial use of the facet structure of the discrepancy constraint set, i.e., that the discrepancy set is defined by only $O(n)$ linear constraints. As we are allowed to "move" in an $O(n)$ dimensional space, one can always square off the $O(n)$ linear constraints with an $O(n)$ dimensional subspace to argue the existence of sufficient degrees of freedom to move in. We note that this intuition while exact for the random walk based approaches, is also intrinsic to another approach due to Rothvoss [104] and Eldan-Singh [45] as well. This approach appears to require the use of Sidak-Khatri Lemma/Gaussian Correlation Inequality [112, 70, 105] which again crucially uses the $O(n)$ facet structure to argue the existence of good partial signing. This reliance on the facet structure of these approaches seem to prevent its use for more general discrepancy problems defined in the space of matrices, for instance, for the so-called Matrix Spencer problem, which we state next.

Problem 2 (Matrix Spencer Problem)**.** *Given a set of n symmetric matrices $A_1, \ldots, A_n \in \mathbb{R}^{n\times n}$ such that $\|A_i\|_{\mathbb{S}_\infty} \leq 1$, find*

$$\min_{x\in\{\pm 1\}^n} \left\| \sum_{i=1}^{n} x_i A_i \right\|_{\mathbb{S}_\infty}$$

One can recover the Spencer problem by taking each column of the matrix A in the Spencer problem, and making that into a diagonal matrix A_i.

As in the Spencer problem, we can show that a random signing gives a bound of $O(\sqrt{n \log n})$ using Matrix Concentration Inequalities and it is conjectured [129, 92] that here, again, a bound of $O(\sqrt{n})$ is possible. The conjecture was known to hold for constant block-sized block diagonal matrices [85] as well as when the rank of each A_i is at most $O(\sqrt{n})$ [55]. It was observed by the author that combining a recent refined matrix concentration inequality due to [14] can be combined with Rothvoss' result to argue the conjecture being true in certain cases. This observation was also used by [20] who additionally noticed that this condition holds true if we assume A_i have rank at most $n/(\log n)^3$. It is important to remark that the rank constraint is somewhat unnatural for this problem as in the setting of diagonal matrices of rank at most $n/(\log n)^3$, a result of Banascyzk [13] already implies a bound of $\sqrt{n}/\log n$.

As mentioned, the Rothvoss [104] and Eldan-Singh [45] approaches also crucially rely on the polyhedral nature of the Spencer problem by looking at the rows of the matrix A, although [55] use a well-known connection of feasibility of SDPs and quantum communication to establish the Matrix Spencer conjecture for matrices of rank at most $\sqrt{n}$. For the Matrix Spencer problem, one would need to apply the Gaussian Correlation Inequality on $e^{O(n)}$ convex sets in order to capture the spectrahedral set defined by the operator norm constraint.

The other class of algorithmic proofs of Spencer's problem are based on random walks starting from the origin and iteratively moving towards a vertex of the discrete hypercube of dimension n, freezing when we get too close to a face of the discrete hypercube. Variants of these random walk based approaches have also been successful for other discrepancy problems like improvements to the Beck-Fiala and Komlos problems [18, 21]. The walks usually pick a subspace of dimension say $n/2$ and take a *small* Gaussian step in that chosen subspace. The crucial point is that the subspace can be adaptively chosen at every point in order to ensure that the discrepancy doesn't become too large, by blocking out directions where the discrepancy is large. The success of this approach relies on showing that one can always find a subspace to move in which keeps the increase in discrepancy to be small, per step. Levy et al. [85] present such an algorithm which is analyzed using the log-sum-exp potential that shows up in the multiplicative weights update method. The algorithm proceeds in each time step t, essentially by looking at $O(n)$ rows with the largest discrepancy and picking the subspace to move in to be orthogonal to the corresponding rows a_i and also move orthogonal to the current point x_t. This certainly ensures that we keep increasing the norm of the current point $\vec{x}_t$ (which can be at most $\sqrt{n}$ as we will stop when we hit a vertex of the hypercube) and using a decreasing rearrangement/Markov argument to say that the log-sum-exp of the discrepancy vector, which can be just viewed as a smooth proxy to the ℓ_∞ norm of the discrepancy vector, grows slowly and never crosses the target discrepancy bound of $O(\sqrt{n})$. Notice however, that this yet again exploits the polyhedral structure of the problem as it uses the fact that we can always pick $O(n)$ rows to walk orthogonal to and not change the discrepancy on those rows and such an argument is likely not possible in the spectrahedral setting of Matrix Spencer.

4.1 Yet Another Algorithm for Spencer's Theorem

We now present another algorithm which can also be used to recover Spencer's theorem and show how that covariance control has a direct analogue in the spectrahedral setting and also motivates how the refined matrix concentration inequalities of [14] should be helpful as was also used by [20]. The analysis, but not the algorithm, here is somewhat similar to that of [85] but will be analyzed using the Stieltjes barrier. An algorithm similar to [85] but analyzed via the Stieltjes barrier also seems to have appeared in an unpublished note of Yin Tat Lee and Mohit Singh. The use of the Stieltjes barrier to capture a smooth proxy of the operator norm in order to bypass a $\log n$ dependence stemming from the use of the exponential moment is well known and successfully implemented in the context of spectral sparsification [22] and the Kadison-Singer problem , which is implied by another matrix discrepancy problem known as Weaver's discrepancy problem [91]. We have a target discrepancy of $c\sqrt{n}$ for some $c > 1$ and will define $2n$ constraints of the form $c\sqrt{n} - a_i^\top x$ and $c\sqrt{n} + a_i^\top x$ which can be used to define a polytope. For notational convenience, we just consider $c\sqrt{n} - a_i^\top x$ and the other sided constraints are completely similar. Now, instead of exploiting the polytope structure crucially, we'll set up a self-concordant barrier for these constraints based on a moment of the resolvent. For any given point $x(t) \in (-1, 1)^n$, we will denote $s_i(t) = c\sqrt{n} - a_i^\top x(t)$ and let $S(t) = \mathsf{Diag}(s(t))$, the diagonal matrix of dimension $n \times n$ which collects all the $s_i(t)$. As we will only walk in the set of alive coordinates, denoted by $L(t) \subseteq [n]$, being those which don't have a coordinate being 1 or -1 and denote $A_{L(t)} \in \mathbb{R}^{m \times L(t)}$ as the A matrix restricted to the columns in $L(t)$. We will use $S(t)$ to "rescale" $A_{L(t)}$ and then take a step based on that, which can be seen as being inspired by affine scaling based interior point methods. We will then pick our subspace to be orthogonal to the current point $x(t)$ as well as orthogonal to the top $n/4$ right singular vectors of $S(t)^{-1}A_{L(t)}$. Our potential will be

Algorithm 3 Algorithm For Spencer's Problem

```
Require: $n \geq 10$
  Input: $A \in \mathbb{R}^{n \times n}$
  $x(0) = \vec{0}_n$
  $\delta = 1/n^2$
  $t = 0$
  $L(t) = \{1, \ldots, n\}$
  while $L(t) \neq \emptyset$ do
      $P_1(x(t)) = \{x | x_i = 0 \text{ for } i \notin L(t)\}$
      $P_2(x(t)) = \{x | x \perp x(t)\}$
      $P_3(x(t)) = \left\{x | x \perp \text{top } n/4 \text{ right singular vectors of } S(t)^{-1}A_{L(t)}\right\}$
      $P(x(t)) = P_1(x(t)) \cap P_2(x(t)) \cap P_3(x(t))$
      $x(t+\delta) = x(t) + \sqrt{\delta}\mathsf{N}(0, P(x(t)))$
      $t = t + \delta$
      $x_i(t) = \mathsf{sgn}(x_i(t))$ for $i \in L(t)$ such that $|x(t)^2 - 1| \leq 1/n$
      $L(t) = L(t) \setminus \{i \in L(t) | \, |x(t)^2 - 1| \leq 1/n\}$
  end while
```

$$\phi(x) = \sum_{i=1}^{n} (c\sqrt{n} - a_i^\top x)^{-p}$$

for some even $p \geq 2$. It is easy to see that these potentials are self-concordant. The analysis of Algorithm 3 relies on the following lemma.

Lemma 36. *Given a point $x_t \in [-1, 1]^n$ such that $s_i(t) > 0$ for all i and hence $\phi(x_t) < \infty$. Based on the update in Algorithm 3, we have, with probability $\geq 1 - e^{-n}$,*

$$\phi(x_{t+\delta}) - \phi(x_t) \leq O(n^{1-2/p})\delta\phi(x_t)^{1+2/p}$$

Proof. First note that the update, with probability $\geq 1 - e^{-n}$, satisfies $a_i^\top(x(t+\delta) - x(t))/s_i(t) < 0.1$, hence, the second order approximation to the potential is good, i.e., we have,

$$\phi(x(t+\delta)) - \phi(x(t)) \leq \langle \nabla\phi(x(t)), x(t+\delta) - x(t) \rangle + \|x(t+\delta) - x(t))\|^2_{\nabla^2\phi(x(t))}$$

We can now understand the update to the potential in expectation.

$$\begin{aligned}
\mathbb{E}\left[\phi(x(t+\delta)) - \phi(x(t))\right] &\leq \mathbb{E}\left[(x(t+\delta)-x(t))^\top \nabla^2\phi(x(t))x(t+\delta)-x(t)\right] && (4.1)\\
&\leq p(p+1)\mathsf{Tr}[A^\top S(t)^{-(p+2)}AP(x(t))]\delta && (4.2)\\
&\leq 2p^2\mathsf{Tr}[(S(t)^{-1}A)^\top S(t)^{-p}S(t)^{-1}AP(x(t))]\delta && (4.3)\\
&= 2p^2\mathsf{Tr}[S(t)^{-p}S(t)^{-1}AP(x(t))(S(t)^{-1}A)^\top]\delta && (4.4)\\
&\leq 2p^2\mathsf{Tr}[S(t)^{-p}]\|S(t)^{-1}AP(x(t))(S(t)^{-1}A)^\top\|_{\mathbb{S}_\infty} && (4.5)
\end{aligned}$$

where the last inequality follows from Holder's inequality.

As $P(x(t))$ is orthogonal to the top $n/4$ right singular vectors of $S(t)^{-1}A$, due to a standard decreasing rearrangement/Markov inequality argument, we have

$$\begin{aligned}
\|S(t)^{-1}AP(x(t))(S(t)^{-1}A)^\top\|_{\mathbb{S}_\infty} &\leq \frac{1}{n/4}\mathsf{Tr}[S(t)^{-1}A(S(t)^{-1}A)^\top]\\
&\leq \frac{4}{n}\sum_{i=1}^{n} s_i(t)^{-2}\|a_i\|_2^2\\
&\leq 4\sum_{i=1}^{n} s_i(t)^{-2}\\
&\leq 4n^{1-2/p}\left(\sum_{i=1}^{n} s_i(t)^{-p}\right)^{2/p} \quad \text{(by Holder's inequality)}
\end{aligned}$$

Plugging this back in Equation 4.5, we get,

$$\mathbb{E}\left[\phi(x(t+\delta)) - \phi(x(t))\right] \leq O(p^2n^{1-2/p})\phi(x(t))^{1+2/p}\delta \tag{4.6}$$

So we have with high probability, $\phi(x(t+\delta)) - \phi(x(t)) \leq O(p^2n^{1-2/p})\phi(x(t))^{1+2/p}$. □

We now prove our main theorem.

Theorem 16. *Algorithm 3 terminates at a vertex of the hypercube and the output signing satisfies* $\|Ax(1)\|_\infty \leq O(\sqrt{n})$

Proof. As the ℓ_2 norm squared of $x(t)$ increases by $O(\delta n)$ in every iteration, we terminate in at most $O(1/\delta)$ iterations. Now, integrating Equation 4.6, we get

$$\begin{aligned}
\phi(x(0))^{-2/p} - \phi(x(1))^{-2/p} &\leq O(pn^{1-2/p})\\
\implies \phi(x(1))^{-2/p} &\geq \phi(x(0))^{-2/p} - O(pn^{1-2/p})\\
\implies \phi(x(1))^{-2/p} &\geq (n^{1-p/2}/c^p)^{-2/p} - O(pn^{1-2/p})\\
\implies \phi(x(1))^{-2/p} &\geq c^2n^{1-2/p} - O(pn^{1-2/p})
\end{aligned}$$

Hence, taking p to be any even constant greater than or equal to 2 and taking c to be a large enough constant, we have $\phi(x(1))^{-2/p} > 0$ and hence $\phi(x(1)) < \infty$. As the potential doesn't blow up to ∞ throughout the process, we are done. □

While the algorithm proposed takes small Gaussian steps, freezing whenever we get too close to a vertex of the hypercube and thus naturally gives us a full signing at the end, the author can prove that the same "algorithm" without constraining to remain inside the hypercube can also be used to establish a sufficient condition for Rothvoss' [104] convex program to find partial signings.

Precisely, while Rothvoss' [**Rot17(Theorem 8)**] says that if there is a subspace of dimension at least $(1 - 1/1000)n$ where the Gaussian volume of the discrepancy body is at least $e^{-n/1000}$, one can actually show that instead of picking a single subspace of sufficiently large dimension, it would suffice to run a covariance controlled, mean zero Brownian motion upto time 1 such that the control on the covariance corresponds to a projection matrix of dimension $\geq (1-1/1000)n$ and argue that at the time 1, this continous martingale satisfies the discrepancy bound with high probability, then Rothvoss' algorithm finds a good partial signing.

We remark now that the control for the covariance of the Gaussian update we picked in every iteration was inspired by affine scaling the matrix A such that the current slack "looks like" the all ones point and doesn't use the facet structure unlike previous approaches to Spencer's problem. Furthermore, as we walk orthogonal to the top $O(n)$ right singular vectors of $S(t)^{-1}A$, we are ensuring that the covariance of the update to the "rescaled discrepancy vector" is kept small. Hence, it is natural to consider whether for the Matrix Spencer problem, trying to keep the covariance of the Gaussian update of the discrepancy matrix small is a good strategy. It is however not clear how to exploit such a strategy to bound the actual operator norm of the discrepancy matrix.

This is where the refined matrix concentration inequalities of Bandiera-Boedihardjo-van Handel [14] come into the picture. We state their main result next.

Theorem 17. *Given matrices $A_1, \dots, A_n \in \mathbb{R}^{d\times d}$, we have*

$$\mathbb{E}\left[\left\|\sum_{i=1}^{n} g_i A_i\right\|_{\mathbb{S}_\infty}\right] \leq \left\|\sum_{i=1}^{n} A_i^2\right\|_{\mathbb{S}_\infty}^{1/2} + O\left(\left\|\sum_{i=1}^{n} \mathsf{vec}(A_i)\mathsf{vec}(A_i)^\top\right\|_{\mathbb{S}_\infty}^{1/2}\right)(\log d)^{3/2}$$

where $g_1, \dots, g_n$ are independent standard Gaussian random variables.

Note that the term multiplying the $(\log d)^{3/2}$ is exactly the norm of the $n^2 \times n^2$ sized covariance matrix of the entries of the Gaussian matrix $\sum_i g_i A_i$ and is also equivalent to the norm of $\mathcal{A}\mathcal{A}^\top$ where $\mathcal{A} \in \mathbb{R}^{n^2\times n}$ is the matrix formed by stacking $\mathsf{vec}(A_i)$ column-wise. Hence, if we can find a subspace of dimension $n/1000$ to truncate such that the Gaussian discrepancy matrix satisfies a norm bound of $O(\sqrt{n})$, then Rothvoss' algorithm will find a good partial signing. Certainly based on Theorem 17, one can ask whether a similar decreasing rearrangement/Markov's inequality trick to truncate the top $n/1000$ right singular vectors of $\mathcal{A} \in \mathbb{R}^{n^2\times n}$, the matrix with $\mathsf{vec}(A_i)$ stacked together, which is just a factorization of the covariance matrix of the Gaussian discrepancy matrix, leads to the norm of this covariance matrix to be bounded by $n/(\log n)^3$, then Rothvoss' algorithm would succeed in finding good partial signings. This observation was used by the authors of [20], who additionally realised that such a condition holds if one assumes that the rank of each matrix A_i is bounded by $n/(\log n)^3$.

Essentially, now it seems that if we take a large Gaussian step, i.e., of Euclidean norm $\Theta(\sqrt{n})$, we incur $\log n$ factors due to requiring a $\log n$ moment of the random matrix or of the resolvent of the random matrix like in [14] however, if we take many small Gaussian steps, each of which has the subspace to truncate chosen adaptively, then we can use a constant moment of the resolvent to avoid $\log n$ factors in the norm, at least in the diagonal case. It is then natural to ask whether there is an analogue of the algorithm in the usual Spencer setting, for the matrix Spencer setting. We next propose one such algorithm which is again inspired by affine scaling style algorithms for semi-definite programming. We conjecture that exactly the algorithm as in Algorithm 4 should also suffice to establish the Matrix Spencer bound of $O(\sqrt{n})$. Here $\mathcal{A}(\mathcal{S})_{L(t)} \in \mathbb{R}^{n^2 \times L(t)}$ is the matrix formed by taking $\mathsf{vec}(S^{-1/2}A_iS^{-1/2})$ corresponding to the active elements $L(t)$ and stacking them column-wise. We remark that $\mathcal{A}(\mathcal{S})$ also plays a crucial role in Chapter 3 for Interior Point Methods for Semi-definite programming.

Algorithm 4 Conjectured Algorithm For Matrix Spencer

```
Require: $n \geq 10$
  Input: $A_1, \ldots, A_n \in \mathbb{R}^{n \times n}$
  $x(0) = \vec{0}_n$
  $\delta = 1/n$
  $t = 0$
  $L(t) = \{1, \ldots, n\}$
  while $L(t) \neq \emptyset$ do
      $P_1(x(t)) = \{x | x_i = 0 \text{ for } i \notin L(t)\}$
      $P_2(x(t)) = \{x | x \perp x(t)\}$
      $P_3(x(t)) = \left\{x | x \perp \text{top } n/4 \text{ right singular vectors of } \mathcal{A}(\mathcal{S})_{L(t)}\right\}$
      $P(x(t)) = P_1(x(t)) \cap P_2(x(t)) \cap P_3(x(t))$
      $x(t+\delta) = x(t) + \sqrt{\delta}\mathsf{N}(0, P(x(t)))$
      $t = t + \delta$
      $x_i(t) = \mathsf{sgn}(x_i(t))$ for $i \in L(t)$ such that $|x(t)^2 - 1| \leq 1/n$
      $L(t) = L(t) \setminus \{i \in L(t) | \, |x(t)^2 - 1| \leq 1/n\}$
  end while
```

The analysis of such an algorithm likely requires tools similar but more general than that which were used in [14] and we leave that for future work.

Bibliography

[1] Deeksha Adil and Sushant Sachdeva. "Faster p-norm minimizing flows, via smoothed q-norm problems". In: *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*. 2020.

[2] Deeksha Adil et al. "Iterative Refinement for lp-norm Regression". In: *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*. 2019.

[3] 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.

[4] 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*. 2017.

[5] Kurt Anstreicher. "Potential Reduction Algorithms". In: (1996).

[6] Kurt M Anstreicher. "The volumetric barrier for semidefinite programming". In: *Mathematics of Operations Research* (2000).

[7] Sanjeev Arora, Elad Hazan, and Satyen Kale. "The multiplicative weights update method: a meta-algorithm and applications". In: *Theory computing (8)*. 2012, pp. 121–164.

[8] Sanjeev Arora and Satyen Kale. "A combinatorial, primal-dual approach to semidefinite programs". In: *STOC*. 2007, pp. 227–236.

[9] 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.

[10] Sanjeev Arora, Satish Rao, and Umesh Vazirani. "Expander flows, geometric embeddings and graph partitioning". In: *Journal of the ACM (JACM)* (2009).

[11] Sanjeev Arora, Satish Rao, and Umesh Vazirani. "Expander flows, geometric embeddings and graph partitioning". In: *J. ACM*. Vol. 56(2). 2009.

[12] David S Atkinson and Pravin M Vaidya. “A cutting plane algorithm for convex programming that uses analytic centers”. In: *Mathematical Programming* 69.1-3 (1995), pp. 1–43.

[13] Wojciech Banaszczyk. “Balancing vectors and gaussian measures of n-dimensional convex bodies”. In: *Random Structures and Algorithms*. Vol. 12 (4). 1998, pp. 351–360.

[14] Afonso S. Bandeira, March T. Boedihardjo, and Ramon van Handel. “Matrix concentration inequalities and free probability”. In: *Inventiones Mathematicae*. 2023.

[15] Nikhil Bansal. “Constructive algorithms for discrepancy minimization”. In: *FOCS*. 2010, pp. 3–10.

[16] Nikhil Bansal. “On a generalization of iterated and randomized rounding”. In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC)*. 2019.

[17] 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)*. 2016.

[18] Nikhil Bansal, Daniel Dadush, and Shashwat Garg. “An algorithm for komlos conjecture matching Banaszczyk’s bound”. In: *FOCS*. 2016.

[19] Nikhil Bansal and Shashwat Garg. “Algorithmic discrepancy beyond partial coloring”. In: *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*. 2017.

[20] Nikhil Bansal, Haotian Jiang, and Raghu Meka. “Resolving Matrix Spencer Conjecture Up to Poly-logarithmic Rank”. In: *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023*. 2023, pp. 1814–1819.

[21] Nikhil Bansal et al. “The Gram-Schmidt Walk: A Cure for the Banaszczyk Blues”. In: *STOC*. 2017.

[22] Joshua D. Batson, Daniel A. Spielman, and Nikhil Srivastava. “Twice-Ramanujan sparsifiers”. In: *SIAM Review*. Vol. 56(2). 2014, pp. 315–334.

[23] Dimitri P. Bertsekas, Angelia Nedi´c, and Asuman E. Ozdaglar. *Convex Analysis and Optimization*. 2003.

[24] 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)*. ACM. 2002, pp. 109–115.

[25] Markus Bläser. “Fast matrix multiplication”. In: *Theory of Computing* (2013), pp. 1–60.

[26] Jan van den Brand et al. “Faster maxflow via improved dynamic spectral vertex sparsifiers”. In: *STOC ’22: 54th Annual ACM SIGACT Symposium on Theory of Computing,* ed. by Stefano Leonardi and Anupam Gupta. ACM, 2022, pp. 543–556.

[27] Jan van den Brand et al. "Solving Tall Dense Linear Programs in Nearly Linear Time". In: *STOC* (2020).

[28] Jan van den Brand. "A Deterministic Linear Program Solver in Current Matrix Multiplication Time". In: *ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2020.

[29] Jan van den Brand et al. "Solving Tall Dense Linear Programs in Nearly Linear Time". In: *52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*. 2020.

[30] Sébastien Bubeck et al. "An homotopy method for $\mathrm{l_p}$ regression provably beyond self-concordance and in input-sparsity time". In: *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC*. 2018.

[31] Peter Bürgisser, Michael Clausen, and Mohammad A Shokrollahi. *Algebraic complexity theory*. Vol. 315. Springer Science & Business Media, 1997.

[32] Yair Carmon et al. "A Rank-1 Sketch for Matrix Multiplicative Weights". In: *Conference on Learning Theory, COLT 2019, 25-28 June 2019, Phoenix, AZ, USA*. 2019, pp. 589–623.

[33] Li Chen et al. "Maximum Flow and Minimum-Cost Flow in Almost-Linear Time". In: *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS*. 2022.

[34] 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)*. SIAM. 2019.

[35] Yu Cheng and Rong Ge. "Non-Convex Matrix Completion Against a Semi-Random Adversary". In: *Conference On Learning Theory (COLT)*. 2018.

[36] Yu Cheng et al. "Faster algorithms for high-dimensional robust covariance estimation". In: *Conference on Learning Theory (COLT)*. 2019.

[37] Paul Christiano et al. "Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs". In: *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC*. 2011.

[38] Michael B Cohen, Yin Tat Lee, and Zhao Song. "Solving Linear Programs in the Current Matrix Multiplication Time". In: *Proceedings of the 51st Annual ACM Symposium on Theory of Computing (STOC)*. 2019.

[39] 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 SIGACT Symposium on Theory of Computing, STOC*. 2019.

[40] Michael B. Cohen et al. "Negative-Weight Shortest Paths and Unit Capacity Minimum Cost Flow in $\tilde{\mathrm{O}}$ $(m^{10/7} \log W)$ Time". In: *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*. 2017.

[41] Michael B. Cohen et al. "Solving SDD linear systems in nearly $m\log^{1/2} n$ time". In: *Symposium on Theory of Computing, STOC*. 2014.

[42] Samuel I. Daitch and Daniel A. Spielman. “Faster approximate lossy generalized flow via interior point algorithms”. In: *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*. Ed. by Cynthia Dwork. 2008.

[43] George B Dantzig. “Maximization of a linear function of variables subject to linear inequalities”. In: *Activity analysis of production and allocation* 13 (1947), pp. 339–347.

[44] Ronen Eldan. “Thin shell implies spectral gap up to polylog via a stochastic localization scheme”. In: *Geometric and Functional Analysis* (2013).

[45] Ronen Eldan and Mohit Singh. “Efficient algorithms for discrepancy minimization in convex sets”. In: *Random Struct. Algorithms*. Vol. 53 (2). 2018, pp. 289–307.

[46] 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.

[47] Yu Gao, Yang P. Liu, and Richard Peng. “Fully Dynamic Electrical Flows: Sparse Maxflow Faster Than Goldberg-Rao”. In: *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS*. IEEE, 2021, pp. 516–527.

[48] Dan Garber and Elad Hazan. “Sublinear time algorithms for approximate semidefinite programming”. In: *Mathematical Programming* 158.1-2 (2016), pp. 329–361.

[49] Michel X Goemans and David P Williamson. “Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming”. In: *Journal of the ACM (JACM)* (1995).

[50] Michel X. Goemans and David P. Williamson. “Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming”. In: *J. ACM*. Vol. 42(6). 1995, pp. 1115–1145.

[51] Jean-Louis Goffin and Jean-Philippe Vial. “Convex nondifferentiable optimization: A survey focused on the analytic center cutting plane method”. In: *Optimization methods and software* (2002).

[52] Andrew V. Goldberg and Satish Rao. “Beyond the Flow Decomposition Barrier”. In: *J. ACM* 45.5 (1998), pp. 783–797.

[53] Martin Grötschel, László Lovász, and Alexander Schrijver. “The ellipsoid method and its consequences in combinatorial optimization”. In: *Combinatorica* (1981).

[54] A. J. Hoffman and H. W. Wielandt. “The variation of the spectrum of a normal matrix”. In: *Duke Math. J.* 20.1 (Mar. 1953), pp. 37–39. DOI: `10.1215/S0012-7094-53-02004-3`. URL: `https://doi.org/10.1215/S0012-7094-53-02004-3`.

[55] Samuel B. Hopkins, Prasad Raghavendra, and Abhishek Shetty. “Matrix Discrepancy from Quantum Information”. In: *STOC*. 2022.

[56] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. 2nd. New York, NY, USA: Cambridge University Press, 2012. ISBN: 0521548233, 9780521548236.

[57] Baihe Huang et al. “Solving SDP Faster: A Robust IPM Framework and Efficient Implementation”. In: *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS*. IEEE, 2022, pp. 233–244.

[58] 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.

[59] Rahul Jain et al. “QIP = PSPACE”. In: *Journal of the ACM (JACM)* (2011).

[60] Arun Jambulapati et al. “Positive semidefinite programming: mixed, parallel, and width-independent”. In: *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*. ACM, 2020.

[61] Haotian Jiang et al. “A Faster Interior Point Method for Semidefinite Programming”. In: *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS*. Ed. by Sandy Irani. IEEE, 2020, pp. 910–918.

[62] Haotian Jiang et al. “An Improved Cutting Plane Method for Convex Optimization, Convex-Concave Games and its Applications”. In: *STOC*. 2020.

[63] David Karger, Rajeev Motwani, and Madhu Sudan. “Approximate graph coloring by semidefinite programming”. In: *Proceedings 35th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 1994.

[64] Narendra Karmarkar. “A new polynomial-time algorithm for linear programming”. In: *Proceedings of the sixteenth annual ACM symposium on Theory of computing (STOC)*. 1984.

[65] Tarun Kathuria, Yang P. Liu, and Aaron Sidford. “Unit Capacity Maxflow in Almost $O(m$
$\{4/3\})Time$”. In: *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS*. 2020.

[66] Jonathan A. Kelner et al. “A simple, combinatorial algorithm for solving SDD systems in nearly-linear time”. In: *Symposium on Theory of Computing Conference, STOC’13*. 2013.

[67] Jonathan A. Kelner et al. “An Almost-Linear-Time Algorithm for Approximate Max Flow in Undirected Graphs, and its Multicommodity Generalizations”. In: *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*. 2014.

[68] Leonid G Khachiyan. “Polynomial algorithms in linear programming”. In: *USSR Computational Mathematics and Mathematical Physics* 20.1 (1980), pp. 53–72.

[69] Leonid G Khachiyan, Sergei Pavlovich Tarasov, and I. I. Erlikh. “The method of inscribed ellipsoids”. In: *Soviet Math. Dokl.* Vol. 37. 1. 1988, pp. 226–230.

[70] C.G. Khatri. “On certain inequalities for normal distributions and their applications to simultaneous confidence bounds”. In: *Ann. Math. Statist.* Vol. 38. 1967, pp. 1853–1867.

[71] Ioannis Koutis, Gary L. Miller, and Richard Peng. “Approaching Optimality for Solving SDD Linear Systems”. In: *SIAM J. Comput.* 43.1 (2014), pp. 337–354.

[72] Kartik Krishnan and John E Mitchell. “Properties of a cutting plane method for semidefinite programming”. In: *submitted for publication* (2003).

[73] Rasmus Kyng and Sushant Sachdeva. “Approximate Gaussian Elimination for Laplacians - Fast, Sparse, and Simple”. In: *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS*. Ed. by Irit Dinur. 2016.

[74] Rasmus Kyng et al. “Flows in almost linear time via adaptive preconditioning”. In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC*. 2019.

[75] Rasmus Kyng et al. “Sparsified Cholesky and multigrid solvers for connection laplacians”. In: *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC*. Ed. by Daniel Wichs and Yishay Mansour. 2016.

[76] Jean B. Lasserre. “Global optimization with polynomials and the problem of moments”. In: *SIAM J. Optim.* Vol. 11(3). 2000, pp. 796–817.

[77] Yin Tat Lee. “Faster algorithms for convex and combinatorial optimization”. PhD thesis. Massachusetts Institute of Technology, 2016.

[78] Yin Tat Lee and Swati Padmanabhan. “An $\mathcal{O}(m/\varepsilon^{3.5})$-Cost Algorithm for Semidefinite Programs with Diagonal Constraints”. In: *Conference on Learning Theory, COLT 2020, 9-12 July 2020, Virtual Event [Graz, Austria]*. Ed. by Jacob D. Abernethy and Shivani Agarwal. Proceedings of Machine Learning Research. PMLR, 2020.

[79] Yin Tat Lee, Satish Rao, and Nikhil Srivastava. “A new approach to computing maximum flows using electrical flows”. In: *Symposium on Theory of Computing Conference, STOC*. 2013.

[80] 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)*. 2015.

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

[82] Yin Tat Lee and Aaron Sidford. “Path Finding Methods for Linear Programming: Solving Linear Programs in Õ(vrank) Iterations and Faster Algorithms for Maximum Flow”. In: *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS*. 2014.

[83] 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)*. 2015.

[84] Yin Tat Lee, Zhao Song, and Qiuyi Zhang. “Solving Empirical Risk Minimization in the Current Matrix Multiplication Time”. In: *Conference on Learning Theory, COLT 2019, 25-28 June*. 2019.

[85] Avi Levy, Harishchandra Ramadas, and Thomas Rothvoss. “Deterministic Discrepancy Minimization via the Multiplicative Weight Update Method”. In: *Integer Programming and Combinatorial Optimization - 19th International Conference, IPCO*. Vol. 10328. Lecture Notes in Computer Science. 2017, pp. 380–391.

[86] Yang P. Liu and Aaron Sidford. “Faster Energy Maximization for Faster Maximum Flow”. In: *STOC* (2020).

[87] Shachar Lovett and Raghu Meka. “Constructive discrepancy minimization by walking on the edges”. In: *FOCS*. 2012, pp. 61–67.

[88] Haihao Lu, Robert M. Freund, and Yurii E. Nesterov. “Relatively Smooth Convex Optimization by First-Order Methods, and Applications”. In: *SIAM Journal on Optimization* 28.1 (2018), pp. 333–354.

[89] Aleksander Madry. “Computing Maximum Flow with Augmenting Electrical Flows”. In: *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS*. 2016.

[90] Aleksander Madry. “Navigating Central Path with Electrical Flows: From Flows to Matchings, and Back”. In: *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS*. 2013.

[91] Adam W. Marcus, Daniel A. Spielman, and Nikhil Srivastava. “Interlacing Families II: Mixed Characteristic Polynomials and the Kadison-Singer Problem”. In: *Annals of Mathematics*. Vol. 182. 2015, pp. 327–350.

[92] Raghu Meka. “Discrepancy and Beating the Union Bound”. In: *Windows on Theory Blog Post: Discrepancy and Beating The Union Bound*. 2014.

[93] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*. Kluwer Academic Publishers, 2004.

[94] Yurii Nesterov and Arkadi Nemirovski. “Conic formulation of a convex programming problem and duality”. In: *Optimization Methods and Software* 1.2 (1992), pp. 95–115.

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

[96] Yurii Nesterov and Arkadi Nemirovski. “Self-concordant functions and polynomial time methods in convex programming. preprint, Central Economic & Mathematical Institute, USSR Acad”. In: *Sci. Moscow, USSR* (1989).

[97] Yurii E. Nesterov and Arkadii Nemirovskii. *Interior-point polynomial algorithms in convex programming*. Vol. 13. Siam studies in applied mathematics. SIAM, 1994.

[98] Pablo Parrilo. “Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization”. In: *PhD thesis, California Institute of Technology.* 2000.

[99] Richard Peng. “Approximate Undirected Maximum Flows in $O(m\text{polylog}(n))$ Time”. In: *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016.* Ed. by Robert Krauthgamer. SIAM, 2016, pp. 1862–1867. DOI: `10.1137/1.9781611974331.ch130`. URL: `https://doi.org/10.1137/1.9781611974331.ch130`.

[100] Richard Peng and Daniel A. Spielman. “An efficient parallel solver for SDD linear systems”. In: *Symposium on Theory of Computing, STOC.* 2014.

[101] 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).* 2012, pp. 101–108.

[102] James Renegar. *A Mathematical View of Interior-point Methods in Convex Optimization.* Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2001.

[103] James Renegar. *A mathematical view of interior-point methods in convex optimization.* MPS-SIAM series on optimization. SIAM, 2001.

[104] Thomas Rothvoss. “Constructive discrepancy minimization for convex sets”. In: *SIAM J. Comput.* Vol. 46 (1). 2017, pp. 224–234.

[105] Thomas Royen. “A simple proof of the Gaussian correlation conjecture extended to multivariate gamma distributions”. In: *arXiv preprint arXiv:1408.1028.* 2014.

[106] Hanif D. Sherali and Warren P. Adams. “A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems”. In: *SIAM J. Discrete Math.* Vol. 3(3). 1990, pp. 411–430.

[107] Jonah Sherman. “Area-convexity, l_∞ regularization, and undirected multicommodity flow”. In: *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC.* 2017.

[108] Jonah Sherman. “Breaking the multicommodity flow barrier for O(vlog n)-approximations to sparsest cut”. In: *50th Annual IEEE Symposium on Foundations of Computer Science (FOCS).* 2009, pp. 363–372.

[109] Jonah Sherman. “Generalized Preconditioning and Undirected Minimum-Cost Flow”. In: *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA.* 2017.

[110] Jonah Sherman. “Nearly Maximum Flows in Nearly Linear Time”. In: *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS.* 2013.

[111] Naum Z Shor. “Cut-off method with space extension in convex programming problems”. In: *Cybernetics and systems analysis* 13.1 (1977), pp. 94–96.

[112] Zbynek Sidak. “Rectangular confidence regions for the means of multivariate normal distributions”. In: *J. Amer. Statist. Assoc.* Vol. 62. 1967, pp. 626–633.

[113] Aaron Daniel Sidford. “Iterative methods, combinatorial optimization, and linear programming beyond the universal barrier”. PhD thesis. Massachusetts Institute of Technology, 2015.

[114] Joel Spencer. “Six standard deviations suffice”. In: *Transactions of the American Mathematical Society*. Vol. 289 (2). 1985, pp. 679–706.

[115] Daniel A. Spielman and Shang-Hua Teng. “Nearly Linear Time Algorithms for Preconditioning and Solving Symmetric, Diagonally Dominant Linear Systems”. In: *SIAM J. Matrix Analysis Applications* 35.3 (2014), pp. 835–885.

[116] Volker Strassen. “Degeneration and complexity of bilinear maps: some asymptotic spectra”. In: *J. reine angew. Math* 413 (1991), pp. 127–180.

[117] Michael J. Todd. “Potential-reduction methods in mathematical programming”. In: *Math. Program.* 76 (1996), pp. 3–45.

[118] Levent Tunçel. “Constant potential primal-dual algorithms: A framework”. In: *Math. Program.* 66 (1994), pp. 145–159.

[119] Levent Tunçel. “On the convergence of primal-dual interior-point methods with wide neighborhoods”. In: *Comp. Opt. and Appl.* 4.2 (1995), pp. 139–158.

[120] Pravin M Vaidya. “A new algorithm for minimizing convex functions over convex sets”. In: *30th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 1989, pp. 338–343.

[121] 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.

[122] Pravin M Vaidya. “Speeding-up linear programming using fast matrix multiplication”. In: *30th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 1989, pp. 332–337.

[123] Lieven Vandenberghe and Stephen P. Boyd. “Semidefinite Programming”. In: *SIAM Review* (1996).

[124] Adrian Vladu. “Interior Point Methods with a Gradient Oracle”. In: *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC*. Ed. by Barna Saha and Rocco A. Servedio. ACM, 2023, pp. 1876–1889.

[125] Max A Woodbury. “Inverting modified matrices”. In: (1950).

[126] Max A Woodbury. “The stability of out-input matrices”. In: *Chicago, IL* 9 (1949).

[127] David B Yudin and Arkadi S Nemirovski. “Evaluation of the information complexity of mathematical programming problems”. In: *Ekonomika i Matematicheskie Metody* 12 (1976), pp. 128–142.

[128] Alp Yurtsever et al. *Scalable Semidefinite Programming*. 2019. arXiv: `1912.02949`.

[129] Anastasios Zouzias. "A matrix hyperbolic cosine algorithm and applications". In: *ICALP*. 2012, pp. 846–858.