Minimum Cost Flows, MDPs, and ℓ_1-Regression in Nearly Linear Time for Dense Instances

Jan van den Brand* Yin Tat Lee† Yang P. Liu‡ Thatchaphol Saranurak§
Aaron Sidford¶ Zhao Song‖ Di Wang**

January 15, 2021

Abstract

In this paper we provide new randomized algorithms with improved runtimes for solving linear programs with two-sided constraints. In the special case of the minimum cost flow problem on n-vertex m-edge graphs with integer polynomially-bounded costs and capacities we obtain a randomized method which solves the problem in $\widetilde{O}(m+n^{1.5})$ time. This improves upon the previous best runtime of $\widetilde{O}(m\sqrt{n})$ (Lee-Sidford 2014) and, in the special case of unit-capacity maximum flow, improves upon the previous best runtimes of $m^{4/3+o(1)}$ (Liu-Sidford 2020, Kathuria 2020) and $\widetilde{O}(m\sqrt{n})$ (Lee-Sidford 2014) for sufficiently dense graphs.

For ℓ_1-regression in a matrix with n-columns and m-rows we obtain a randomized method which computes an ϵ-approximate solution in $\widetilde{O}(mn+n^{2.5})$ time. This yields a randomized method which computes an ϵ-optimal policy of a discounted Markov Decision Process with S states and A actions per state in time $\widetilde{O}(S^2A+S^{2.5})$. These methods improve upon the previous best runtimes of methods which depend polylogarithmically on problem parameters, which were $\widetilde{O}(mn^{1.5})$ (Lee-Sidford 2015) and $\widetilde{O}(S^{2.5}A)$ (Lee-Sidford 2014, Sidford-Wang-Wu-Ye 2018).

To obtain this result we introduce two new algorithmic tools of independent interest. First, we design a new general interior point method for solving linear programs with two sided constraints which combines techniques from (Lee-Song-Zhang 2019, Brand et al. 2020) to obtain a robust stochastic method with iteration count nearly the square root of the smaller dimension. Second, to implement this method we provide dynamic data structures for efficiently maintaining approximations to variants of Lewis-weights, a fundamental importance measure for matrices which generalize leverage scores and effective resistances.

*janvdb@kth.se. KTH Royal Institute of Technology, Sweden.
†yintat@uw.edu. University of Washington and Microsoft Research Redmond, USA.
‡yangpliu@stanford.edu. Stanford University, USA.
§saranurak@ttic.edu. Toyota Technological Institute at Chicago, USA.
¶sidford@stanford.edu. Stanford University, USA.
‖zhaos@ias.edu. Institute for Advanced Study, USA.
**wadi@google.com. Google Research, USA.

Contents

1 Introduction

We consider solving linear programs expressed in the following primal/dual form:

$$(P) = \min_{\substack{x\in\mathbb{R}^m:\mathbf{A}^\top x=b\\ \ell_i\le x_i\le u_i\ \forall i\in[m]}} c^\top x \quad \text{and} \quad (D) = \max_{\substack{y\in\mathbb{R}^n, s\in\mathbb{R}^m\\ \mathbf{A}y+s=c}} b^\top y + \sum_{i\in[n]} \min(\ell_i s_i, u_i s_i). \tag{1}$$

where $b \in \mathbb{R}^n$, $c \in \mathbb{R}^m$, $\mathbf{A} \in \mathbb{R}^{m\times n}$ and each $\ell_i \le u_i \in \mathbb{R}$.

Equation (1) naturally encompasses prominent continuous and combinatorial optimization problems. When $\ell_i = 0$ and $u_i = \infty$, (1) corresponds to the standard primal-dual formulation of linear program. In the case when $\mathbf{A}$ is the incidence matrix of a graph, i.e. $b = \vec{0}$, and $\ell_i = 0$ for all $i \in [m]$, (1) corresponds to the problem of computing a minimum cost circulation in a directed graph with linear costs c and edge capacities given by u (Section 7). Further, in the case when each $u_i = 1$, $\ell_i = -1$, and $b = 0$, (1) corresponds to solving ℓ_1 regression (Section 8.3) and can be used to solve Markov Decision Processes (MDPs) [SWWY18] (Section 8.4).

Recent advances in interior point methods (IPMs), a prominent class of continuous optimization methods, and data structures have led to nearly linear runtimes for solving fundamental classes of (1) to high precision with only a polylogarithmic dependence on problem parameters. In [BLSS20] an $\widetilde{O}(mn+n^{2.5})$ time randomized method was obtained for solving (1) when $\ell_i = 0$ and $u_i = \infty$ for all $i \in [m]$, i.e., when the problem is in standard form. Further, in [BLN+20] a randomized method was obtained for solving minimum cost perfect matching in bipartite graphs in time $\widetilde{O}(m + n^{1.5})$, i.e. when the problem is in standard form and $\mathbf{A}$ is the incidence matrix of a bipartite graph.

Unfortunately, though it is well known that all linear programs can be written in standard form, naïvely transforming (1) to standard form can increase the dimension of the problem, i.e. turn n to $\Omega(m)$. This issue prevents the application of the recent advances in [BLN+20, BLSS20] to (1) when both u_i and ℓ_i are bounded. Consequently, despite extensive study, obtaining nearly linear running times for solving (1), maximum flow, minimum cost flow, ℓ_1 regression, and Markov Decision Processes to high-precision with a polylogarithmic dependence on problem parameters in nearly linear time in high-dimensional dense instances has been elusive.

The issue of losing density when reformulating (1) in standard form is a known difficulty in obtaining improved runtimes from continuous optimization methods. It arose in [LS14, LS19] and was addressed and leveraged to obtain improved randomized runtimes for minimum cost flow, ℓ_1-regression [LS15], and MDPs [LS14, LS15, SWWY18]. These methods provide IPMs which work directly with (1). The IPMs re-weight constraints based on variations of Lewis weights [CP15, LS19], a natural notion of row importance for matrices which generalizes leverages scores, and implement and apply the corresponding methods efficiently.

Lewis weight reweighting schemes were also key to the aforementioned nearly linear time algorithms for solving linear programs in standard form [BLSS20] and computing minimum-cost bipartite perfect matchings [BLN+20] on dense instances. Unfortunately, a key technique applied by these results is that when (1) is in standard form (i.e. $\ell_i = 0$ and $u_i = \infty$ for all $i \in [n]$) it is possible to leverage the primal-dual structure of the problem to rewrite the optimality conditions of [LS19] in terms of leverage scores, a simple special case of Lewis weights. Leveraging this structure, these papers design IPMs and data-structures for efficiently leveraging and manipulating leverage scores towards achieving their runtimes. Unfortunately, in the case that both ℓ_i and u_i are bounded, this same technique doesn't directly apply (see Section 3.1).

In this paper, we show how to overcome this difficulty and directly obtain nearly linear time algorithms for solving (1), minimum cost flow[1], ℓ_1-regression [LS15], and MDPs, in on

[1]Though Lewis weight reweighting was used to achieve state-of-the-art $\widetilde{O}(m+n^{1.5})$ runtimes for minimum-cost bipartite perfect matching on m-edge n-node graphs in [BLN+20], the same work showed that it was not needed to obtain $\widetilde{O}(n\sqrt{m})$ runtimes. Consequently, $\widetilde{O}(n\sqrt{m})$ runtimes for minimum cost flow on m-edge n-node graphs may be achievable without the full range of techniques in this paper.

moderately dense instances. First, we provide a new IPM directly tailored to solving (1) which directly works with a variant of Lewis weights. Our method applies techniques in robustifying IPMs [CLS19, LSZ19, Bra20, BLSS20, BLN+20, SY20, JSWZ20] and in particular techniques from [LSZ19] on robust IPMs for solving empirical risk minimization problems, to a variant of the Lewis-weight based optimality conditions suggested by [LS19]. Further, the method applies and generalizes sampling and analysis techniques from [BLN+20]. The combination of these techniques and interaction with Lewis weights induces a number of technical challenges. Interestingly, ultimately, our analysis leverages higher order smoothness properties of barriers for the intervals $\{x : \ell_i \leq x \leq u_i\}$ (Definition 4.1 and Lemma 4.2).

As with many recent results [CLS19, LSZ19, Bra20, BLSS20, BLN+20, SY20, JSWZ20], this new IPM reduces the challenge of solving (1) to solving a sequence of data structure problems and appropriately initializing and rounding the iterates of the IPM. One particularly challenging aspect is that our IPM requires that regularized ℓ_p-Lewis weights be maintained efficiently throughout the algorithm. Though previous work [BLSS20, BLN+20] provided various results on dynamically maintaining leverage scores, i.e. the special case when $p = 2$, and there are known algorithms for computing Lewis weights efficiently using leverage scores, the complexity of such a naïve approach is unclear. We overcome this issue by providing both a more careful reduction from Lewis weight maintenance to leverage score maintenance and a direct reduction from leverage score maintenance to detecting large rows in a dynamically changing matrix, what we call a heavy hitter data structure [BLN+20, BLSS20].

By considering specialized heavy hitter data structures for the various problems we consider, providing additional data structure as needed, and carefully applying the resulting IPM we obtain the main runtime results of this paper. In the special case of solving linear programs in standard form we provide new sampling data structure for matrices that allow us to improve upon the runtime of [BLSS20] in certain settings.

1.1 Our Results

Here we provide the main results of this paper, including new nearly linear time algorithms for solving (1) in different settings.

Linear Programming with Two-sided Constraints Our main contributions are efficient algorithms for solving primal/dual LPs with two-sided constraints (1) via IPMs. In the general case, where there is no additional structure on $\mathbf{A}$ we obtain the following result.

Theorem 1.1 (Primal solution for general LPs)**.** *Let $\mathbf{A} \in \mathbb{R}^{m \times n}$, $c, \ell, u \in \mathbb{R}^m$, and $b \in \mathbb{R}^n$. Assume that there is a point x satisfying $\mathbf{A}^\top x = b$ and $\ell_i \leq x_i \leq u_i$ for all $i \in [m]$. Let $W \stackrel{\text{def}}{=} \max(\|c\|_\infty, \|\mathbf{A}\|_\infty, \|b\|_\infty, \|u\|_\infty, \|\ell\|_\infty, \frac{\max_i(u_i - \ell_i)}{\min_i(u_i - \ell_i)})$. For any $\delta > 0$ there is an algorithm running in time $\widetilde{O}((mn + n^{2.5}) \log(W/\delta))$ that with high probability (w.h.p.) which computes a vector $x^{(\text{final})}$ satisfying*

$$\|\mathbf{A}^\top x^{(\text{final})} - b\|_\infty \leq \delta \quad \text{and} \quad \ell_i \leq x_i^{(\text{final})} \leq u_i \ \forall i \quad \text{and} \quad c^\top x^{(\text{final})} \leq \min_{\substack{\mathbf{A}^\top x = b \\ \ell_i \leq x_i \leq u_i \forall i}} c^\top x + \delta.$$

The prior best result runtimes for achieving guarantees comparable to Theorem 1.1 were $\widetilde{O}(m^{\max\{\omega, 2+1/18\}})$ [JSWZ20] ($\omega \approx 2.37286$ [Wil12, Gal14, AW21] is the exponent of current matrix multiplication) and $\widetilde{O}((\text{nnz}(\mathbf{A}) + n^2)\sqrt{n})$ [LS15]. Whenever $\mathbf{A}$ is tall and dense, i.e. $m \geq n^{1.5}$ and $\text{nnz}(\mathbf{A}) = \Omega(mn)$, this corresponds to a nearly linear time algorithm for solving (1) to high precision.

Interestingly, even in the special case when $\ell_i = 0$ and $u_i = \infty$ for all $i \in [n]$ this improves upon the previous best runtimes of $\widetilde{O}(m^\omega)$ and $\widetilde{O}((\text{nnz}(\mathbf{A}) + \sqrt{n})\sqrt{n})$ mentioned above, as well as $\widetilde{O}(mn + n^3)$ [BLSS20]. In particular, we improve upon [BLSS20] by improving the additive $\widetilde{O}(n^3)$ term to a $\widetilde{O}(n^{2.5})$. This improvement stems from IPM sampling techniques of [BLN+20] (as refined in this paper) and new data structures introduced in this paper.

ℓ_1-Regression and MDPs Theorem 1.1 immediately yields improved runtimes for additional prominent optimization problems. For instance, when $b = \vec{0}$ and $\ell_i = -1$ and $u_i = 1$ for all $i \in [n]$, the dual formulation in (1) encodes to ℓ_1-regression. Consequently, we obtain the following result.

Theorem 1.2 (Dual solution for general LPs, ℓ_1-regression). *Let $\mathbf{A} \in \mathbb{R}^{m \times n}$, $c \in \mathbb{R}^m$, and $\delta > 0$. There is an algorithm running in time $\widetilde{O}((mn + n^{2.5}) \log(W/\delta))$ for $W \stackrel{\text{def}}{=} \max(\|c\|_\infty, \|\mathbf{A}\|_\infty)$ which w.h.p. computes a vector $z \in \mathbb{R}^n$ such that*

$$\|\mathbf{A}z + c\|_1 \leq \min_{z \in \mathbb{R}^n} \|\mathbf{A}z + c\|_1 + \delta.$$

As with Theorem 1.1 this runtime is nearly linear whenever $\mathbf{A}$ is tall and dense, i.e. $\text{nnz}(\mathbf{A}) = \Omega(mn)$ and $m \geq n^{1.5}$. Further, as with Theorem 1.1, in the regime of high accuracy algorithms, it improves upon the results of [CLS19, LSZ19, Bra20, SY20, JSWZ20, LS15] mentioned above.

Further, leveraging a reduction from [SWWY18], in certain settings this yields to improved running times for solving MDPs, a fundamental mathematical model for reasoning about uncertainty. An instance of the discounted Markov Decision Process (DMDP) is specified by a tuple (S, A, P, r, γ) where S is a the state space, A is the action space, P describes state-action-state transition probabilities, r describes state-action rewards in range $[-M, M]$, and $\gamma \in (0, 1)$ is a discount factor. The goal is to compute a policy that maps each state to an action that that (approximately) maximizes the reward in a certain sense. (See Section 8.4 for more precise definition of the problem) We obtain the following result.

Theorem 1.3 (Discounted MDP). *Given a DMDP (S, A, P, r, γ), there is an algorithm that, with high probability, computes a ε-optimal policy π in $\widetilde{O}((|S|^2|A| + |S|^{2.5}) \log(\frac{M}{(1-\gamma)\epsilon}))$ time.*

Since the input size of the state-action-state transition is $\Omega(|S|^2|A|)$, this algorithms runs in nearly-linear time whenever $|A| \geq \sqrt{|S|}$. Also, this result directly improves upon the previous algorithm with running time $\widetilde{O}((|S|^{2.5}|A|) \log(\frac{M}{(1-\gamma)\epsilon}))$ [LS14, SWWY18].

Minimum Cost Flow In the minimum cost flow problem, we are given a connected directed graph $G = (V, E, u, c)$ with edges capacities $u \in \mathbb{R}^E_{\geq 0}$ and costs $c \in \mathbb{R}^E$. We call $x \in \mathbb{R}^E$ an s-t flow for $s, t \in V$ if $x_e \in [0, u_e]$ for all e in E and for each vertex $v \notin \{s, t\}$ the amount of flow entering v, i.e. $\sum_{e=(a,v)\in E} x_e$ equals the amount of flow leaving v, i.e. $\sum_{e=(v,b)\in E} x_e$. The value of s-t flow is the amount of flow leaving s (or equivalently, entering t). The maximum flow problem is to compute an s-t flow of maximum value. In the minimum cost maximum flow problem, the goal is to compute a maximum s-t flow of minimum cost, $\sum_{e\in E} c_e x_e = c^\top x$.

Such problems can be expressed in the form of (1) by taking $\mathbf{A}$ as the graph incidence matrix, letting ℓ_i and u_i denote edge capacities, and choosing b appropriately. Unfortunately, directly applying Theorem 1.1 would yield a large $\widetilde{O}((mn+n^{2.5}) \log W)$ runtime which does not improve upon previous results. However, applying the techniques developed in this paper along with Laplacian system solvers [ST04, KMP10, KMP11, KOSA13, CKM+14, PS14, LPS15, KLP+16, KS16] and data structure ideas from [BLN+20] specific to graphs we prove the following theorem.

Theorem 1.4 (Min cost flow). *There is an algorithm that, given a n-vertex, m-edge, directed graph $G = (V, E, u, c)$ integral edge capacities $u \in \mathbb{Z}^E_{\geq 0}$ and costs $c \in \mathbb{Z}^E$, with high probability, computes a minimum cost maximum flow in $\widetilde{O}(m \log(\|u\|_\infty \|c\|_\infty) + n^{1.5} \log^2(\|u\|_\infty \|c\|_\infty))$ time.*

Efficiently solving the minimum cost flow problem to high accuracy gives an algorithm for maximum flow in the same runtime on weighted graphs, and we give the reduction formally in Corollary 7.9 in Section 7.3. Using standard capacity scaling methods [AO91], we can improve the $\log^2 W$ dependence to $\log W$ on the $n^{1.5}$ term in this case, where $W = \max(\|u\|_\infty)$. The previous best runtimes for mincost flow were $m^{4/3+o(1)} \log W$ in the case of unit capacity graphs [AMV20] and $\widetilde{O}(m\sqrt{n} \log^{O(1)} W)$ [LS19]. Our algorithm improves on these for dense graphs, and in particular runs in nearly linear time for $m \geq n^{1.5}$.

Year	Authors	References	Time	
			Sparse	Dense
1970	Dinitz	[Din70]	mn	
1997	Goldberg, Rao	[GR98]	$m^{3/2}\log W$	$mn^{2/3}\log W$
2013	Madry	[Mad13, Mad16]	$m^{10/7}W^{1/7}$	
2013	Lee and Sidford	[LS14]		$m\sqrt{n}\log^{O(1)} W$
2020	Liu and Sidford	[LS20b]	$m^{11/8}W^{1/4}$	
2020	Liu and Sidford; Kathuria	[LS20a, Kat20]	$m^{4/3}W^{1/3}$	
2020		**This paper**		$(m+n^{1.5})\log W$

Table 1: The summary of the results for the **maximum flow** problem. W denotes the maximum capacity. Subpolynomial ($n^{o(1)}$) terms are hidden. For simplicity, we only list exact algorithms which yielded polynomial improvements.

Year	Authors	References	Time	
			Sparse	Dense
1972	Edmonds and Karp	[EK72]	$m^2\log W$	
1987	Goldberg and Tarjan	[GT90]		$mn\log W$
2008	Daitch and Spielman	[DS08]	$m^{3/2}\log^2 W$	
2013	Lee and Sidford	[LS14]		$m\sqrt{n}\log^{O(1)} W$
2017	* Cohen, Madry, Sankowski, Vladu	[CMSV17]	$m^{10/7}\log W$	
2020	* Axiotis, Madry, Vladu	[AMV20]	$m^{4/3}\log W$	
2020		**This paper**		$m\log W + n^{1.5}\log^2 W$

Table 2: The summary of the results for the **minimum-cost flow** problem. Subpolynomial ($n^{o(1)}$) terms are hidden. W denotes the maximum absolute value of capacities and costs. For simplicity, we only list exact algorithms which yielded polynomial improvements. Results marked with an asterisk work only on unit-capacity graph.

1.2 Related Work

The problems we consider in this paper, e.g. linear programming, dynamic data structures, minimum cost flow, ℓ_1-regression, MDPs, are all incredibly well-studied. Each has an extensive history and numerous results. Here, we provide just a brief summary of the results and tools most directly related to this paper.

Linear Programming IPMs: There has been significant work towards the design of IPMs for linear programming, starting from [Kar84, Ren88]. More recently, there have been IPM-based runtime improvements to linear programming by decreasing the number of iterations [LS14], proving that maintaining approximate primal/dual solutions suffice [CLS19, LSZ19, Bra20, BLSS20, SY20, JSWZ20, BLN+20], and using data structures to decrease iteration costs [LS15, CLS19, LSZ19, Bra20, SY20, JSWZ20, BLN+20]. There is a related line of work on strongly polynomial linear programming [VY96, MT03, MT05, DHNV20, DNV20], where the goal is to achieve exact solutions with improved parameter dependencies, instead of high accuracy solutions as we do here. In [LSZ20] it was shown how to implement IPMs in the semi-streaming model.

ℓ_1-Regression: There have been several algorithms for ℓ_1-regression in both the low accuracy ($\text{poly}(\varepsilon^{-1})$ dependence) [Cla05, Nes09, CMMP13, YCRM16, DLS18] and high accuracy ($\log(1/\varepsilon)$ dependence) [MM13, LS15, CLS19] regimes. The state of the art results in the high-accuracy regime are [LS15] which achieves a $\widetilde{O}((\text{nnz}(\mathbf{A})+n^2)\sqrt{n}\log(1/\varepsilon))$ runtime, and [JSWZ20] which achieves a $\widetilde{O}(n^{\max\{\omega,2+1/18\}})$ runtime, improving on $\widetilde{O}(n^{\max\{\omega,2+1/6\}})$ [CLS19].

Minimum Cost Flow: There has been significant work towards combinatorial algorithms for the minimum cost flow problem [EK72, Tar85, Orl84, GT88, GT90, Orl93] in both the strongly polynomial and weakly polynomial regimes. Daitch and Spielman [DS08] showed that one can use a Laplacian system solver to implement steps of an IPM to achieve a more efficient mincost flow algorithm with logarithmic capacity dependence. Since then, runtime improvements have been achieved by reducing the number of iterations of a general IPM to $\widetilde{O}(\sqrt{n})$ [LS14] and to $O(m^{1/3+o(1)})$ [AMV20] for graphs with unit-capacity, improving on $\widetilde{O}(m^{10/7}\log W)$ [CMSV17].

Markov Decision Process (MDP): We focus on solving *discounted* MDPs.[2] Previous works in the high precision regime (i.e. logarithmic dependency on error) includes [Tse90, LDK95, SWWY18] with the best running time of $\widetilde{O}((|S|^2|A| + \frac{|S||A|}{(1-\gamma)^3})\log(\frac{M}{\epsilon}))$. Strongly polynomial time exact algorithms are also known [Ye05, Ye11, Sch16].

For algorithms that depend logarithmically on one minus the discount factor γ, the algorithm by [LS14] implies a $\widetilde{O}((|S|^{2.5}|A|)\log(\frac{M}{(1-\gamma)\epsilon}))$ running time (shown in [SWWY18]). Our result in Theorem 1.3 directly improves this algorithm.

There is another line of work focusing on fast algorithms in the low precision regime with polynomial dependency on the error parameter [KS98, AMK13, Wan17, SWWY18, SWW+18, Wai19, Wan20, AKY20, LWC+20]. This setting is not directly comparable to our result.

Dynamic Data Structures: IPMs reduce the task of solving linear programs to the task of solving linear systems. Instead of solving these linear systems from scratch in each iteration, these iterative algorithms can be sped up by using data structures that efficiently maintain the matrix inverse corresponding to the linear system [Kar84, Vai89, NN91, LS15, CLS19, LSZ19, SY20, JSWZ20, BLSS20]. There also exist data structures to efficiently maintain the solution of the linear system instead of (or in addition to) the corresponding matrix [San04, BNS19, Bra20, JSWZ20, Bra21]. Recently there have also been data structures developed that are able to efficiently maintain an approximation of the primal/dual solution [LSZ19, BLSS20, BLN+20, JSWZ20]. For graph applications these algorithms are based on the dynamic expander decomposition technique [NSW17, SW19, BBN+20, GRST21]. For general LPs, the data structure for maintaining approximate primal/dual solutions are based on heavy hitters and sketching [JL84, KNPW11, Pag13, NN13, KN14, LNNT16, PSW17, CJN18, NS19, NSW19].

1.3 Organization

We give the preliminaries in Section 2. Our overview is in Section 3, split into overview of the IPM in Section 3.1 and overview of the data structures in Section 3.2. We present our IPM in Section 4, and show our regularized Lewis weight maintenance data structure in Section 5. We show how to use data structures in implement our IPM in Section 6. We analyze the runtime of our IPM in the graphical setting and show applications and mincost flow and maxflow in Section 7. Finally, we analyze the runtime of our IPM for general linear programs and show applications to Markov Decision Processes in Section 8.

Several additional pieces are deferred to the appendix. In Appendix A we give omitted proofs from Section 4. The remaining sections of the appendix give data structures based on previous methods of [BLN+20]. In Section B we give our HeavyHitter and sampling data structures, and in Section C we give our leverage score maintenance data structure. We show how to maintain the primal variable and gradient of the centrality potential in Section D, and show how to maintain the dual slack variable in Section E. Finally we state the graph specific data structures based on expander decompositions in Section F.

[2] Other variants of this problems includes deterministic MDPs (equivlent to the min-mean cycle problem) [DG98, CTCG+98, Mad02, BLN+20] and average-reward MDPs [Mah96, AO06, JOA10].

2 Preliminaries

We follow similar notation as in [BLN+20]. We let $[n] \stackrel{\text{def}}{=} \{1, 2, ..., n\}$ and $\vec{e}_i$ denote the i-th standard unit vector. We use $\widetilde{O}(\cdot)$ notation to hide $(\log \log W)^{O(1)}$, $\log \epsilon^{-1}$, and $(\log n)^{O(1)}$ factors, where W typically denotes the largest absolute value used for specifying any value in the problem (e.g. demands and edge weights) and n denotes the number of nodes. When we write *with high probability* (or w.h.p), we mean with probability $1 - n^c$ for any constant $c > 0$. We write $\mathbf{1}_{\text{condition}}$ for the indicator variable, which is 1 if the condition is true and 0 otherwise.

Diagonal Matrices Given a vector $v \in \mathbb{R}^d$ for some d, we write $\mathbf{Diag}(v)$ for the $d \times d$ diagonal matrix with $\mathbf{Diag}(v)_{i,i} = v_i$. For a vector v we also write $\mathbf{V}$ for the diagonal matrix $\mathbf{Diag}(v)$ when clear from context.

Matrix and Vector operations Given vectors $u, v \in \mathbb{R}^d$ for some d, we perform arithmetic operations $\cdot, +, -, /, \sqrt{\cdot}$ element-wise. For example $(u \cdot v)_i = u_i \cdot v_i$ or $(\sqrt{v})_i = \sqrt{v_i}$. For the inner product we write $\langle u, v \rangle$ and $u^\top v$ instead. For a vector $v \in \mathbb{R}^d$ and a scalar $\alpha \in \mathbb{R}$ we let $(\alpha v)_i = \alpha v_i$ and $(v + \alpha)_i = v_i + \alpha$.

For symmetric matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n \times n}$ we write $\mathbf{A} \preceq \mathbf{B}$ to indicate that $x^\top \mathbf{A} x \leq x^\top \mathbf{B} x$ for all $x \in \mathbb{R}^n$ and define $\succ$, $\prec$, and $\succeq$ analogously. We call any matrix (not necessarily symmetric) *non-degenerate* if its rows are all non-zero and it has full column rank.

We use $u \approx_\epsilon v$ to denote that $\exp(-\epsilon)v \leq u \leq \exp(\epsilon)v$ entrywise and $\mathbf{A} \approx_\epsilon \mathbf{B}$ to denote that $\exp(-\epsilon)\mathbf{B} \preceq \mathbf{A} \preceq \exp(\epsilon)\mathbf{B}$. Note that this notation implies $u \approx_\epsilon v \approx_\delta w \Rightarrow u \approx_{\epsilon+\delta} w$, and $u \approx_\epsilon v \Rightarrow u^\alpha \approx_{\epsilon \cdot |\alpha|} v^\alpha$ for any $\alpha \in \mathbb{R}$.

For any matrix $\mathbf{A}$ with real entries, let $\text{nnz}(\mathbf{A})$ denote the number of non-zero entries in $\mathbf{A}$ and $\text{nnz}(a_i)$ be the number of non-zero entries in the i-th row of $\mathbf{A}$.

Note that we can express the approximation error of Lemma 7.1 as some spectral approximation, i.e. there exists some $\mathbf{H} \approx_{20\epsilon} \mathbf{A}^\top \mathbf{W} \mathbf{A}$ such that $\mathbf{H}\overline{x} = b$ [BLSS20, Section 8].

Leverage Scores and Lewis-Weights For any non-degenerate matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ we let $\sigma(\mathbf{A}) \in \mathbb{R}^m$ with $\sigma(\mathbf{A})_i \stackrel{\text{def}}{=} (\mathbf{A}(\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top)_{i,i}$ denote $\mathbf{A}$'s *leverage scores*. For $p \in (0, \infty)$ and non-degenerate matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ we define the ℓ_p *Lewis weight* as the solution $w \in \mathbb{R}^m_{>0}$ to the equation $w = \sigma(\mathbf{W}^{\frac{1}{2} - \frac{1}{p}} \mathbf{A})$, where $\mathbf{W} = \mathbf{diag}(w)$. We use a regularized Lewis weight in our algorithms, and this is defined in Definition 4.5.

Norms We write $\| \cdot \|_p$ for the ℓ_p-norm, i.e. $\|v\|_p := (\sum_i |v_i|^p)^{1/p}$, $\|v\|_\infty = \max_i |v_i|$ and $\|v\|_0$ being the number of non-zero entries of v. For a positive definite matrix $\mathbf{M}$ we define $\|v\|_\mathbf{M} = \sqrt{v^\top \mathbf{M} v}$. For a vector τ we define $\|v\|_\tau := (\sum_i \tau_i v_i^2)^{1/2}$ and $\|v\|_{\tau+\infty} := \|v\|_\infty + C \log(4m/n) \|v\|_\tau$ for a large constant C, where $m \geq n$ are the dimensions of the constraint matrix of the linear program (we define $\|v\|_{\tau+\infty}$ again in Definition 4.9).

3 Overview of Approach

In this section, we give an overview of the major aspects of our algorithm, including the path following IPMs (Section 3.1), the data structures necessary for its implementation (Section 3.2), and how to combine them for our applications (Section 3.3).

3.1 IPM

As context and motivation for our method, we start by discussing the IPM of [LS19]. For matrices $\mathbf{A} \in \mathbb{R}^{m \times n}$, vectors $b \in \mathbb{R}^n$, $c \in \mathbb{R}^m$, and lower and upper bounds $\ell, u \in \mathbb{R}^m$, this IPM solves linear programs of the form

$$\min_{\substack{\mathbf{A}^\top x = b \\ \ell_i \leq x_i \leq u_i \forall i \in [m]}} c^\top x$$

to high accuracy. The runtime of this method is dominated by the runtime needed for solving $\widetilde{O}(\sqrt{n})$ linear systems of the form $\mathbf{A}^\top \mathbf{D}\mathbf{A}$ for non-negative diagonal matrices $\mathbf{D}$. To achieve this result, for all $i \in [n]$ [LS19] considers 1-self-concordant barriers $\phi_i : (\ell_i, u_i) \to \mathbb{R}$ for the intervals for i, e.g. $\phi_i(x) = -\log(u_i - x) - \log(x - \ell_i)$ (Lemma 4.2), and for a path parameter μ, considers the central path of points

$$x_\mu \stackrel{\text{def}}{=} \operatorname{argmin}_{\mathbf{A}^\top x = b} c^\top x + \mu \sum_{i=1}^{m} w_i \phi_i(x_i), \tag{2}$$

for a well-chosen *weight function* $w \in \mathbb{R}^m$. For standard IPMs such as that of Renegar [Ren88], $w_i = 1$ for all i, and in [LS19] and our algorithm w is a function of x (though it is often convenient to think of w as fixed during a step). Optimality conditions tell us that for fixed w, (2) holds if $\mathbf{A}^\top x_\mu = b$ and $c + \mathbf{A}y + \mu \mathbf{W} \nabla \Phi(x) = 0$ for some vector $y \in \mathbb{R}^n$, where $\mathbf{W} = \mathbf{diag}(w)$ is the diagonal matrix of the weights w. This optimality condition can be re-written as $s + \mu \mathbf{W} \nabla \Phi(x) = 0$ where $s = c + \mathbf{A}y$ denotes the dual slack variables.

In both [LS19] and this work we allow w to depend on x and define a *weight function* $w(x) : \mathbb{R}^m \to \mathbb{R}^m_{>0}$. We say that a triple (x, s, μ) is *central* for weight function $w(x)$ if

$$\mathbf{A}^\top x = b \quad , \quad s = \mathbf{A}y + c \ , \text{and} \quad s + \mu \mathbf{W}(x) \nabla \Phi(x) = 0. \tag{3}$$

A choice of $\mathbf{W}(x)$ made in [LS19], and which we used a regularized version of in this paper (Definitions 4.5, 4.6), is a ℓ_p Lewis weight for $p = 1 - \frac{1}{4\log(4m/n)}$. The ℓ_p Lewis weight function $w(x)$ is defined as the solution to

$$w(x) = \sigma(\mathbf{W}(x)^{\frac{1}{2} - \frac{1}{p}} (\nabla^2 \Phi(x))^{-\frac{1}{2}} \mathbf{A}), \tag{4}$$

where $\sigma(\cdot)$ denotes the leverage scores of a matrix. Using this weight function, [LS19] is able to argue that $\widetilde{O}(\sqrt{n})$ steps of an IPM suffice to solve the linear program, as opposed to $\widetilde{O}(\sqrt{m})$ from Renegar's method. Carefully implementing methods based on variants of the ℓ_p-Lewis weight function solves the maximum flow problem in time $\widetilde{O}(m\sqrt{n})$ and general LPs in time $\widetilde{O}(\sqrt{n}(\text{nnz}(A) + n^\omega))$, accounting for the runtime needed to solve linear systems.

To obtain further runtime improvements, there has been significant work towards performing less work per iteration by speeding up the linear system solve times via inverse maintenance [LS15], as well as more recent work showing that such methods can in fact be implemented even with only approximate values for the primal and dual variables x, s [CLS19, LSZ19, Bra20, BLSS20, SY20, JSWZ20, BLN+20]. To illustrate these *robust IPMs*, consider the simple case where $\ell_i = 0$ and $u_i = \infty$ for all i, i.e. the condition on x in the linear program is simply $x \geq 0$, and $\phi_i(x_i) = -\log x_i$. In this case the centrality condition, (3), reduces to $xs = w(x)\mu$ and this motivates the following *centrality potential*

$$\Psi(x, s, \mu) \stackrel{\text{def}}{=} \sum_{i=1}^{m} \cosh\left(\lambda \left(\frac{x_i s_i}{w(x)_i \mu} - 1\right)\right) \tag{5}$$

for $\lambda = \Theta(\log m/\varepsilon)$. Maintaining $\Psi(x, s, \mu) \leq \text{poly}(m)$ at all times ensures that $xs \approx_\varepsilon w(x)\mu$. Consequently, these robust IPMs take projected Newton steps that induce gradient descent steps on the potential Ψ to guarantee that it stays small in expectation throughout the algorithm.

The analysis in [BLSS20, BLN+20] critically relied on the fact that only one-sided constraints, $x \geq 0$, were imposed, instead of a two-sided constraint, $\ell \leq x \leq u$. These works leveraged that the centrality condition $xs = w(x)\mu$ for ℓ_p Lewis weights can be written in terms of leverage scores of a slightly different diagonal weighting. Specifically, if $xs = w(x)\mu$, where $w(x)$ is the ℓ_p Lewis weight, then for $\alpha = 1 - p$ we have that $xs = \sigma(\mathbf{S}^{-1/2-\alpha} \mathbf{X}^{1/2-\alpha} \mathbf{A})$. The reliance on this fact impairs extending it to the setting of two-sided constraints.

We bypass this issue by working directly with ℓ_p Lewis weights and the centrality condition (3) for general 1-self-concordant functions ϕ. Interestingly, our analysis requires a fourth derivative condition in Definition 4.1, beyond the standard third derivative condition of self-concordance. Formally, we consider the centrality potential (Definition 4.8)

$$\Psi(x, s, \mu) \stackrel{\text{def}}{=} \sum_{i=1}^{m} \cosh\left(\lambda\left(\frac{s_i + \mu\tau(x)_i\phi_i'(x_i)}{\mu\tau(x)_i\sqrt{\phi_i''(x_i)}}\right)\right), \tag{6}$$

where intuitively, the denominator arises from normalizing by the Hessian of the current point x. To analyze the progress of the Newton steps towards decreasing the potential, analysis is required to understand and bound derivatives of the ℓ_p Lewis weights. Additionally, there are several other technical challenges, including working with a regularized version of the ℓ_p Lewis weight to ensure that the Lewis weights are all $\geq n/m$, carefully maintaining approximate feasibility of the primal variable x, and using spectral sparsifiers of $\mathbf{A}^\top\mathbf{D}\mathbf{A}$ instead of the true matrix for efficient inverse maintenance. This loss of feasibility requires us to use a sampling procedure on the primal variable, and we build a general theory for valid sampling distributions that allow our algorithm to work (Definition 4.13). At a high level, we show that any sampling scheme that satisfies various properties, such as bounded variance, maximum, and mean preservation suffices to implement our IPM. The goal of showing that these generalized sampling schemes work is to handle both sampling each coordinate independently and sampling coordinates proportional to weights, so that we can handle the graphical case (as in [BLN+20]) and linear programs.

Overall, we show that we can takes steps of size $\Omega(n^{-1/2})$ while maintaining that the expected potential is polynomially bounded, and that all points x we maintain are approximately feasible. In this way, we can implement an IPM for two-sided linear programs that requires $\widetilde{O}(\sqrt{n})$ steps that only approximately maintains the primal variable x and dual slack s.

3.2 Data Structures

As outlined in Section 3.1, in contrast to [BLSS20, BLN+20], our IPM maintains approximate regularized ℓ_p Lewis weights for $p \in [1/2, 2)$. To efficiently implement the IPM we do not want to recompute the Lewis weights from scratch in every iteration. Instead we seek a data structure that maintains approximate Lewis weights. Here we describe how such a data structure can be obtained by reducing to the HeavyHitter data structure problem defined below:

Definition 3.1 (Heavy hitter)**.** *For $c \in \mathbb{R}^m$ and $P, Q \in \mathbb{R}_{>0}$ with $nP \geq \|c\|_1 \geq P$, we call a data structure with the following procedures a (P, c, Q)-HeavyHitter data structure:*

- *Initialize($\mathbf{A} \in \mathbb{R}^{m\times n}, g \in \mathbb{R}^m_{>0}$) Let $\mathbf{A}$ be a matrix with $c_i \geq \text{nnz}(a_i))$, $\forall i \in [m]$ and $P \geq \text{nnz}(\mathbf{A})$. The data structure initializes in $O(P)$ time.*
- *Scale($i \in [m], b \in \mathbb{R}$): Sets $g_i \leftarrow b$ in $O(c_i)$ time.*
- *QueryHeavy($h \in \mathbb{R}^n, \epsilon \in (0, 1)$): Returns $I \subset [m]$ containing exactly those i with $|(\mathbf{G}\mathbf{A}h)_i| \geq \epsilon$ in $O(\epsilon^{-2}\|\mathbf{G}\mathbf{A}h\|_c^2 + Q)$ time.*

A contribution of our work is to show that, if we have such a data structure for a matrix $\mathbf{A}$, then we are able to efficiently maintain the Lewis weights of $\mathbf{V}\mathbf{A}$ for a diagonal matrix $\mathbf{V}$ that changes over time (i.e. $\mathbf{V} = (\nabla^2\Phi(x))^{-1/2}$ when used inside our IPM, see (4)).

Constructing HeavyHitter-data structures was key to advances in [BLN+20, BLSS20]. For the special case where $\mathbf{A}$ is an edge-vertex incidence matrix, [BLN+20] constructed a HeavyHitter-data structure with complexities $P = \widetilde{O}(m)$, $c_i = \widetilde{O}(1)$ for all $i \in [m]$, and $Q = \widetilde{O}(n \log W)$, where W is a bound on the ratio of the largest to smallest non-zero entry in $\mathbf{G}$. This data structure will be useful for our min-cost flow application. In [BLSS20] a HeavyHitter-data structure was given for general $m \times n$ matrices, where $P = \widetilde{O}(\text{nnz}(\mathbf{A}))$, $c_i = \widetilde{O}(n)$

for all $i \in [m]$, and $Q = \widetilde{O}(n)$. This data structure can be used for general linear programs. Our algorithm for solving general LPs use this data structure from [BLSS20] and the algorithms for graph problems such as min-cost flow use the data structure from [BLN+20].

We now outline how to reduce the task of maintaining the Lewis weights $\tau(\mathbf{GA})$ under updates to $\mathbf{G}$, to the HeavyHitter problem. This reduction is done via the intermediate data structure problem of maintaining the leverage scores $\sigma(\mathbf{GA})$ under updates to $\mathbf{G}$. We show that Lewis weight maintenance can be reduced to leverage score maintenance and show that leverage score maintenance can be reduced to the HeavyHitter problem.

3.2.1 Regularized Lewis Weights

We are interested in the regularized ℓ_p-Lewis weight of a matrix $\mathbf{M}$ which is defined as the value $\tau(\mathbf{M})$ that satisfies the recursive equation $\tau = \sigma(\mathbf{T}^{1/2-1/p}\mathbf{M}) + z$ for a given vector $z \in \mathbb{R}^m_{>0}$, and $\mathbf{T} = \mathbf{diag}(\tau)$. We want a data structure that maintains an approximation $\overline{\tau} \approx_\epsilon \tau(\mathbf{VA})$ for any $p \in [1/2, 2)$ and $\epsilon > 0$ under updates to $\mathbf{V}$. Note that for the IPM we use $p = 1 - 1/(4\log(4m/n))$; consequently, $p \in [1/2, 2)$ but intuitively, may be thought of as an approximate ℓ_1-Lewis weight.

To outline our data structure, we first want to outline the algorithm of Cohen and Peng [CP15] that can be adapted to compute an approximation the regularized ℓ_p-Lewis weight $\tau(\mathbf{M})$ in the static setting (i.e. when the input matrix does not change over time). Given some matrix $\mathbf{M}$ we initialize with $w = \vec{1}_m$ and repeatedly set

$$w \leftarrow (w^{2/p-1}(\sigma(\mathbf{W}^{1/2-1/p}\mathbf{M}) + z))^{p/2}. \tag{7}$$

One can prove (see Lemma 5.3) that each iteration reduces the approximation error by a $1 - p/2$ factor, i.e. if we had $w \approx_\gamma \sigma(\mathbf{W}^{1/2-1/p}\mathbf{M}) + z$ before (7), then we have $w \approx_{\gamma(1-p/2)} \sigma(\mathbf{W}^{1/2-1/p}\mathbf{M}) + z$ after (7). Since $\vec{1}_m \approx_{O(\log(m))} \sigma(\mathbf{M}) + z$ (by $n/m \le z \le \mathrm{poly}(m)$), after $\Theta(\log((\log m)/\varepsilon))$ iterations of (7) we have $w \approx_\epsilon \tau(\mathbf{M})$.

We provide an efficient extension of this analysis to the dynamic setting. One natural idea for doing this would be to initialize $K = \Omega(\log((\log m)/\varepsilon))$ data structures $D_1, \ldots, D_K$ that maintain the following approximate leverage scores: Let $w^{(1)} = \vec{1}_m$ and define recursively

$$\overline{\sigma}^{(i)} \approx \sigma((\mathbf{W}^{(i)})^{1/2-1/p}\mathbf{VA}) + z \tag{8}$$

$$w^{(i+1)} \leftarrow ((w^{(i)})^{2/p-1}\overline{\sigma}^{(i)})^{p/2} \tag{9}$$

where $\overline{\sigma}^{(i)}$ is maintained a the leverage score data structure D_i discussed in Section 3.2.2

If the leverage score data structures are accurate enough (i.e. the approximation in (8) is good enough), then $w^{(K)}$ for $K = \Omega(\log((\log m)/\varepsilon))$ would be a good approximation of the Lewis weight. Further, this $w^{(K)}$ can be maintained under updates to $\mathbf{V}$: When $\mathbf{V}$ changes, we update all the leverage score data structures $D_1, \ldots, D_K$. Likewise, if some $\overline{\sigma}^{(i)}$ changes, then we update $w^{(i+1)}$ and the data structure D_{i+1} that maintains $\overline{\sigma}^{(i+1)} \approx \sigma((\mathbf{W}^{(i+1)})^{1/2-1/p}\mathbf{VA}) + z$.

Problems with this approach While one can show that the previously outlined approach would indeed allow us to maintain approximate regularized Lewis weights (assuming the approximate leverage scores $\overline{\sigma}^{(i)}$ are accurate enough), we are not able to analyze the time complexity of this process. This is because an update to some D_i (i.e. when $w^{(i-1)}$ changes) causes the output $\overline{\sigma}^{(i)}$ to change as well, thus changing the input to D_{i+1}. This means an update to D_i might propagate through all other $D_{i'}$ for $i' > i$. The computational cost of this propagation of the updates is difficult to analyze because updating the j-th entry of the input of some data structure D_i requires time proportional to $\overline{\sigma}^{(i)}_j$. Now, only for large i do we know that $w^{(i)} \approx \overline{\sigma}^{(i)}$ and can show that, $w^{(i)}$ is an approximation to the regularized Lewis weights. When this happens, we have bounds on how $w^{(i)}$ changes from guarantees of the IPM and this implies a small time complexity for D_i. However, for small i, $w^{(i)} \not\approx \overline{\sigma}^{(i)}$ and the same bounds and complexity analysis does not immediately apply.

One attempt to fix this issue would be to start from a moderately good approximation, i.e.

$$w^{(1)} \approx_\gamma \sigma((\mathbf{W}^{(1)})^{1/2-1/p}\mathbf{V}\mathbf{A}) + z \tag{10}$$

for $0 < \gamma = O(\epsilon)$. Then after only $K = O(1)$ recursions we have that $w^{(K)}$ is an ϵ-approximation of the regularized Lewis weights. Since here we only have $O(1)$ data structures D_i and each $\overline{\sigma}^{(i)}$ is at least an $O(\epsilon)$-approximation of the regularized Lewis weight, we are able to bound the time for propagating the updates through all D_i.

The assumption (10) on $w^{(1)}$ can be satisfied as follows. Assume the input $\mathbf{V}$ changes to some $\mathbf{V}'$. We know by guarantees of the IPM that $\mathbf{V} \approx_{O(\epsilon)} \mathbf{V}'$. Let $\overline{w}^{(K)}$ be the value $w^{(K)}$ we previously returned as ϵ-approximation of the regularized Lewis weight, then we have

$$\overline{w}^{(K)} \approx_\epsilon \sigma((\mathbf{W}^{(K)})^{1/2-1/p}\mathbf{V}\mathbf{A}) + z \approx_{O(\epsilon)} \sigma((\mathbf{W}^{(K)})^{1/2-1/p}\mathbf{V}'\mathbf{A}) + z.$$

Thus we can define $w^{(1)} := \overline{w}^{(K)}$ as the required moderately good approximation.

The problem with this approach is that the vector $w^{(K)}$ might change in many entries when switching from $\mathbf{V}$ to $\mathbf{V}'$. Thus we might have to perform many updates to the data structure D_1 at the start of each iteration, resulting in a larger than desired complexity.

The final algorithm Our regularized Lewis weight data structure combines these ideas with one more trick to bound the number of updates to D_1; we delay the updates a bit. We know that we have $w^{(K)} \approx_\epsilon \tau$ where τ is the exact regularized Lewis-weight. We additionally maintain some $\overline{w}^{(K)}$ where we set $\overline{w}_j^{(K)} \leftarrow w_j^{(K)}$ whenever the entry $w_j^{(K)}$ changes and $\overline{w}_j^{(K)} \not\approx_{3\epsilon} w_j^{(K)}$. This way we know $\overline{w}_j^{(K)}$ only changes, whenever τ_j must have changed by at least an $\exp(\epsilon)$ factor. By guarantees of the IPM (Lemma 4.35) we can bound how often entries of τ change by an $\exp(\epsilon)$ factor, which then in turn bounds how often entries of $\overline{w}^{(K)}$ are changed. As we set $w^{(1)} := \overline{w}^{(K)}$ whenever the input $\mathbf{V}$ changes, we can now bound the number of updates to D_1. Note that here we still satisfy the requirement (10) because $\overline{w}^{(K)} \approx_{3\epsilon} w^{(K)}$, so we can still bound the time spent on propagating the updates to D_1 through all other D_i. This way we maintain a good approximation of the Lewis-weight with a low overall complexity bound.

3.2.2 Leverage Scores

In Section 3.2.1 we outlined how to maintain approximate Lewis-weights, if we have access to a data structure that can maintain approximate leverage scores. Here we outline how to efficiently maintain an approximation $\overline{\sigma} \approx_\epsilon \sigma(\mathbf{G}\mathbf{A}) + z$ for $\mathbf{G} = \mathbf{Diag}(g)$, $g \in \mathbb{R}^m_{>0}$, $z \in \mathbb{R}^m_{>0}$. Here the vector g is allowed to change over time, while matrix $\mathbf{A}$ and vector z are fixed, and our task is to create a data structure to maintain $\overline{\sigma}$. We obtain such a data structure by reducing to the HeavyHitter problem (Definition 3.1) for the same matrix $\mathbf{A}$. Since variants of this have been considered in prior work, we first compare their results and explain why these data structures are not sufficient for our reduction to maintain regularized Lewis weights. We then describe how we obtain our data structure for leverage scores, but we do not yet optimize the complexity to highlight the general idea. At last, we outline how to speed-up the resulting data structure.

Comparison to previous work The general idea of our leverage score data structure is the same as in [BLSS20] and [BLN+20], the main difference here is how we improve the complexity. Specifically, the leverage score data structures from [BLSS20] and [BLN+20] were able to maintain an ϵ-approximation of the leverage scores $\overline{\sigma} \approx_\epsilon \sigma(\mathbf{G}\mathbf{A})$, if the input $\mathbf{G}$ was an $O(\epsilon/\log n)$-approximation of some other $\widetilde{\mathbf{G}}$ that satisfied some stability properties, i.e. the diagonal matrix $\widetilde{\mathbf{G}}$ must change very slowly over time. For previous IPMs $\mathbf{G} = \overline{\mathbf{X}}^{1/2}\overline{\mathbf{S}}^{-1/2}$, where $\overline{x} \approx_{O(\epsilon/\log n)} x$, $\overline{s} \approx_{O(\epsilon/\log n)} s$, so $\mathbf{G}$ was an $O(\epsilon/\log n)$-approximation of $\widetilde{\mathbf{G}} := \mathbf{X}^{1/2}\mathbf{S}^{-1/2}$ where both x, s are stable (i.e. they change slowly over the runtime of the algorithm) by guarantees of the IPM. Thus data structures in [BLSS20] and [BLN+20] were able to maintain leverage scores efficiently.

Unfortunately, these data structures are not usable for our Lewis-weight reduction. This is because in order for the recursion in (8) and (9) to yield an ϵ-approximation of the Lewis-weight, the leverage scores $\overline{\sigma}^{(i)}$ must be at least an ϵ-approximation as well. However, for the old leverage score data structure to return an ϵ-approximation, the input must be $O(\epsilon/\log n)$-close to some other stable sequence. We use as input $\mathbf{G} = (\mathbf{W}^{(i)})^{1/2-1/p}\mathbf{V}$ (see (9)), which does not satisfy the required property. This is because, while the exact Lewis-weight τ would satisfy the required stability properties by guarantees of the IPM, the input vector $w^{(i)}$ is at best an ϵ-approximation of the exact Lewis-weight τ. In summary, the complexity bounds of the previous leverage score data structures do not apply when we use them for our Lewis-weight reduction due to the additional precision we require. So while the idea of our leverage score data structure is the same as in [BLSS20] and [BLN+20], we must analyze and optimize the complexity in a different way.

High-level Idea for maintaining Leverage Scores Note that the output size (i.e. dimension of $\overline{\sigma}$) is m and we want to maintain the leverage scores in $o(m)$ time, so we can not afford to recompute all entries of $\overline{\sigma}$ in each iteration. Instead, the high-level idea is that in each iteration we (i) detect a set $I \subset [m]$ of indices i where $\overline{\sigma}_i \not\approx_\epsilon \sigma(\mathbf{VA})_i + z_i$, and (ii) compute a new approximation of $\sigma(\mathbf{VA})_i + z_i$ for all $i \in I$ and update $\overline{\sigma}_i$ accordingly. Thus we split the outline of the data structure into these two parts:

Computing Few Leverage Scores: We start by outlining task (ii) as that one is easier. Computing few leverage scores is standard (see e.g. Spielman-Srivastava [SS11]) and we explain it briefly as we build on it. For a matrix $\mathbf{X}$ let $\mathbf{P}(\mathbf{X}) \stackrel{\text{def}}{=} \mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top$ denote an orthogonal projection matrix. By $\mathbf{P}(\mathbf{VA}) = \mathbf{P}(\mathbf{VA})\mathbf{P}(\mathbf{VA})$ we have that $\mathbf{P}(\mathbf{VA})_{i,i} = \|\vec{e}_i^\top\mathbf{P}(\mathbf{VA})\|_2^2$, so we can reformulate maintaining approximate leverage scores as maintaining an approximation of these norms for $i = 1, \ldots, m$. Given some set $I \subset [m]$ we can compute this norm for $i \in I$ by using a JL-matrix[3] $\mathbf{J}$ and computing the matrix $\mathbf{M} := (\mathbf{A}^\top\mathbf{V}^2\mathbf{A})^{-1}\mathbf{AVJ}^\top$. Next, we obtain an approximation of the i-th leverage score by computing

$$\|\vec{e}_i^\top\mathbf{VAM}\|_2^2 \approx \|\vec{e}_i^\top\mathbf{VA}(\mathbf{A}^\top\mathbf{V}^2\mathbf{A})^{-1}\mathbf{A}^\top\mathbf{V}\|_2^2 = \|\vec{e}_i^\top\mathbf{P}(\mathbf{VA})\|_2^2. \tag{11}$$

Here the complexity will be dominated by computing $\mathbf{M}$ and computing (11) for all $i \in I$. Given that $\mathbf{J}$ needs only some $\widetilde{O}(1)$ rows to yield a good approximation of the norm, we only need to solve very few linear systems in $\mathbf{A}^\top\mathbf{V}^2\mathbf{A}$ to compute $\mathbf{M}$ and $\mathbf{M}$ has very few columns, so the norm (11) can be computed quickly.

Detecting Leverage Score Changes: We now outline how to solve task (i), i.e. how to detect when $\overline{\sigma}_i \not\approx \sigma_i(\mathbf{VA}) + z_i$. For that assume that $\mathbf{V}'$ changes to $\mathbf{V}$ and we already had $\overline{\sigma}_i \approx \sigma(\mathbf{V}'\mathbf{A})_i + z_i$ from the previous iteration. Then we must detect indices i where $\sigma(\mathbf{V}'\mathbf{A})_i + z_i \not\approx \sigma_i(\mathbf{VA}) + z_i$, because for those i the previous $\overline{\sigma}_i$ can no longer be a good approximation. As the vector z is fixed, it suffices to find indices i with $|\sigma(\mathbf{V}'\mathbf{A})_i - \sigma(\mathbf{VA})_i| > \epsilon z_i$ for some small enough $\epsilon > 0$. Using the interpretation of the leverage scores being the norm of the rows of $\mathbf{P}$, we can find such indices i by searching for indices where

$$\|\vec{e}_i^\top\left(\mathbf{P}(\mathbf{V}'\mathbf{A}) - \mathbf{P}(\mathbf{VA})\right)\|_2 \geq \|\vec{e}_i^\top\mathbf{P}(\mathbf{V}'\mathbf{A})\|_2 - \|\vec{e}_i^\top\mathbf{P}(\mathbf{VA})\|_2 > \epsilon\sqrt{z_i}$$

If we simply return all i where $v_i \neq v_i'$, then the only remaining i we must detect are those with

$$\|\vec{e}_i^\top\mathbf{V}\mathbf{Z}^{-1/2}\mathbf{A}\underbrace{\left((\mathbf{A}^\top\mathbf{V}^2\mathbf{A})^{-1}\mathbf{A}^\top\mathbf{V} - (\mathbf{A}^\top\mathbf{V}'^2\mathbf{A})^{-1}\mathbf{A}^\top\mathbf{V}'\right)\mathbf{J}^\top}_{=:\mathbf{M}'}\|_2 > \epsilon$$

where $\mathbf{J}$ is again a JL-matrix and $\mathbf{Z} = \mathbf{Diag}(z)$. Note that because $\mathbf{J}^\top$ has few columns, it suffices to look for large entries of the matrix vector products $\mathbf{V}\mathbf{Z}^{-1/2}\mathbf{A}\mathbf{M}'\vec{e}_k$ for $k = 1, ..., \widetilde{O}(1)$, which in turn can be solved by the HeavyHitter data structure.

[3] A JL-matrix $\mathbf{J}$ satisfies $\|\mathbf{J}v\|_2 \approx \|v\|_2$ for any fixed vector v. For example a random Gaussian matrix with $O(\epsilon^{-2}\log n)$ rows yields w.h.p a $(1 \pm \epsilon)$-approximation of the norm.

So far we only discussed how to detect indices i where the leverage score changed a lot within a single iteration. However, it could also happen that a leverage score changes only a little in each iteration such that after many iteration we have $\overline{\sigma}_i \not\approx \sigma(\mathbf{VA})_i + z_i$. To detect these slowly changing indices, we follow the approach appeared in [BLSS20, BLN+20]. That is, we perform the same trick used for a single iteration, but instead for each $j = 0, ..., \log \sqrt{n}$, we check whether the leverage score has changed significantly in the past 2^j iterations, i.e., the matrix $\mathbf{V}'$ now refers to the state of $\mathbf{V}$ some 2^j iterations in the past.

Improving the complexity For the data structure we outlined so far, the main bottleneck is solving linear systems in $\mathbf{A}^\top\mathbf{V}^2\mathbf{A}$ and computing the product $\mathbf{A}^\top\mathbf{V}\mathbf{J}^\top$. Both of these require $\operatorname{nnz}(\mathbf{A}) = \Omega(m)$ time, which is too slow for our purposes.[4] To speed this up, we use leverage score sampling [SS11] to construct a random sparse diagonal matrix $\mathbf{R}$ with $\widetilde{O}(n)$ nonzero entries and $\mathbf{A}^\top\mathbf{V}^2\mathbf{R}\mathbf{A} \approx \mathbf{A}^\top\mathbf{V}\mathbf{A}$. Careful analysis shows that the algorithm outlined above still works, when solving systems in $\mathbf{A}^\top\mathbf{V}^2\mathbf{R}\mathbf{A}$ and when using $\mathbf{A}^\top\mathbf{V}\mathbf{R}^{1/2}\mathbf{J}^\top$ instead of $\mathbf{A}^\top\mathbf{V}\mathbf{J}^\top$. Because of the sparsity of $\mathbf{R}$, the $\operatorname{nnz}(\mathbf{A})$ cost decreases to $\widetilde{O}(n \cdot \max_i \operatorname{nnz}(a_i))$.

However, this speed-up yields a new problem. If we use two different random matrix $\mathbf{R}$ and $\mathbf{R}'$ with $\mathbf{A}^\top\mathbf{V}^2\mathbf{R}\mathbf{A} \approx \mathbf{A}^\top\mathbf{V}\mathbf{A}$ and $\mathbf{A}^\top\mathbf{V}'^2\mathbf{R}'\mathbf{A} \approx \mathbf{A}^\top\mathbf{V}'\mathbf{A}$, then the runtime of the HeavyHitter data structure can become very large. This is because the HeavyHitter data structures must find large entries of $\mathbf{V}\mathbf{Z}^{-1/2}\mathbf{A}\mathbf{M}'\vec{e}_k$ and by Definition 3.1 the complexity of that task scales in $\|\mathbf{V}\mathbf{Z}^{-1/2}\mathbf{A}\mathbf{M}'\vec{e}_k\|_2^2$, so the total cost for all k scales in $\|\mathbf{V}\mathbf{Z}^{-1/2}\mathbf{A}\mathbf{M}'\|_F^2$. Without random sampling, this Frobenius-norm can be bounded by stability properties of the IPM. However, the Frobenius norm is very sensitive to spectral changes which causes the norm to blow-up when using two different $\mathbf{R}$ and $\mathbf{R}'$ (i.e. two different spectral approximations) for $\mathbf{A}^\top\mathbf{V}\mathbf{A}$ and $\mathbf{A}^\top\mathbf{V}'\mathbf{A}$. Thus we wish to use a single random $\mathbf{R}$ that yields a valid approximation for both.

To see how to construct such $\mathbf{R}$, consider classic leverage score sampling first. If one sets $\mathbf{R}_{i,i} = 1/p_i$ independently for each $i \in [m]$ with probability p_i (where $p_i \geq \min(1, \sigma(\mathbf{VA})_i \log n/\varepsilon^2)$) and $\mathbf{R}_{i,i} = 0$ otherwise, then $\mathbf{A}^\top\mathbf{V}^2\mathbf{R}\mathbf{A} \approx \mathbf{A}^\top\mathbf{V}^2\mathbf{A}$. To make sure that $\mathbf{R}$ also has the property $\mathbf{A}^\top\mathbf{V}'^2\mathbf{R}\mathbf{A}$, we use $p_i = 1$ for all i where $\sigma(\mathbf{VA})_i \not\approx_1 \sigma(\mathbf{V}'\mathbf{A})_i$ and $p_i = 2\overline{\sigma}_i$ otherwise. The indices i for which we have to set $p_i = 1$ are simply those where we recently had to change $\overline{\sigma}_i$. Further, we need $p_i = 1$ for all i with $v_i \neq v_i'$. This is because we can then bound $\mathbf{A}^\top\mathbf{V}^2\mathbf{R}\mathbf{A} - \mathbf{A}^\top\mathbf{V}'^2\mathbf{R}\mathbf{A} = \mathbf{A}^\top\mathbf{R}(\mathbf{V}^2 - \mathbf{V}'^2)\mathbf{A}$ more easily. If we did not choose $p_i = 1$, and $\mathbf{R}_{i,i}$ happens to be non-zero because of the sampling, then $\mathbf{R}_{i,i}(\mathbf{V}^2 - \mathbf{V}'^2)$ would blow-up the difference $(v_i - v_i')$ by an $1/p_i$ factor. So by choosing $p_i = 1$ for all i with $v_i \neq v_i'$, we are able to prove better complexity bounds.

For comparison, in [BLSS20] the leverage score data structure did not use a single $\mathbf{R}$ and instead they used two different $\mathbf{R}$ and $\mathbf{R}'$. They fixed the issue of the Frobenius-norm being large by carefully updating $\mathbf{R}$ to $\mathbf{R}'$ and performing an amortized analysis on the sum of Frobenius-norms over several iterations. However, this analysis required the random $\mathbf{R}$ to be updated over several iterations, which meant the same randomness had to be re-used in all those iterations, thus resulting in difficulties with handling adaptive adversaries. We instead use only one random $\mathbf{R}$ per iteration and this random $\mathbf{R}$ is resampled in every iteration, thus no randomness is re-used and adaptive adversaries are not an issue.

3.2.3 Further Data Structures

We now outline all the other data structures used by our algorithms. Some of these data structures were developed in [BLN+20, BLSS20] though we perform small modifications to them in Appendix D and Appendix E. Here we give a brief description of these data structures and how they are used to efficiently implement our IPM. A more detailed overview for how to implement our IPM can be found in Section 6.1 where we provide the exact statements of the involved data structures.

[4] This is true, even if we assume access to some preconditioner of $\mathbf{A}^\top\mathbf{V}\mathbf{J}^\top$.

As mentioned in Section 3.1, our IPM only requires access to approximations $\overline{x}, \overline{s}$ of the iterates x, s. The updates to these vectors are roughly of the following form

$$s^{(\text{new})} \leftarrow s + \mathbf{A}\mathbf{H}^{-1}\mathbf{A}^\top \Phi''(\overline{x})^{-1/2} g \tag{12}$$

for some gradient-vector g, matrix $\mathbf{H} \approx \mathbf{A}^\top \overline{\mathbf{T}}^{-1} \Phi''(\overline{x})^{-1} \mathbf{A}$ for diagonal matrix $\overline{\mathbf{T}} = \mathbf{Diag}(\overline{\tau})$ with the approximate Lewis weight $\overline{\tau}$ on the diagonal. Note that naive computation of (12) would require $O(\text{nnz}(\mathbf{A}) + n^\omega)$ time per iteration. This can be sped up via data structures that efficiently maintain partial solutions of this expression. The task of computing (12) can be split into three subtasks: (i) Compute $\mathbf{A}^\top \Phi''(\overline{x})^{-1/2} g$, solved by a data structure from [BLN+20] which we modify in Appendix D. (ii) Multiply the result by $\mathbf{H}^{-1}$, solved either via Laplacian solver (e.g. when considering min-cost flow) or a data structure from [BLSS20] (restated in Appendix B). We can essentially use the previous data structure, except we sample $\mathbf{H}$ (the spectral sparsifier) by the ℓ_p Lewis weights that we are already maintaining instead of by leverage score. (iii) Let $v^{(i)}$ be the vector $\mathbf{H}^{-1}\mathbf{A}^\top \Phi''(\overline{x})^{-1/2} g$ we computed with the previous data structures during iteration number i of the IPM. Then after the t-th iteration of the IPM the vector s is given by $s^{(\text{init})} + \sum_{i=1}^{t} \mathbf{A} v^{(i)}$ according to (12). Maintaining an approximation of such matrix vector products is done via a data structure from [BLN+20] which we modify in Appendix E. At last, Section 3.1 mentioned that the update to the primal solution must be sampled. For graph applications such as max-flow we use a data structure from [BLN+20] (restated in Appendix F) to perform this sampling efficiently. For general linear programs we construct a new data structure in Appendix B.

3.3 Putting Everything Together

Given our new IPM for two-sided constraints (Section 3.1) and data structures for implementing this IPM (Section 3.2), we apply them to obtain our results in a standard way.

First of all, our IPM needs to start with a *centered* initial point (i.e. an initial point with small centrality potential (6)). Given an LP instance, we modify the instance by adding an identity block to the constraints and corresponding variables so that a centered initial point can be obtained analytically. This allows us to apply the IPM which moves the initial point to a near optimal point of the modified instance in $\widetilde{O}((mn + n^{2.5}) \log W)$ time (using data structures from previous sections). The modified instance also guarantees that, at near-optimal points, the added variables must have value very close to zero. This allows us to round the near optimal point of the modified instance to a near optimal point of the original instance by a single linear system solve which takes $\widetilde{O}((\text{nnz}(\mathbf{A}) + n^\omega) \log W)$. This gives an algorithm for solving an LP with two-sided constraints in Theorem 1.1.

As our IPM maintains not only a primal solution but also a dual slack, by solving a linear system involving the dual slack at the near-optimal point, this gives a dual solution and solves the ℓ_1-regression problem as stated in Theorem 1.2. Given an ℓ_1-regression algorithm, we immediately obtain an algorithm for discounted MDPs using a known reduction by [SWWY18] and obtain Theorem 1.3.

For a given min-cost flow instance, we modify the instance with the same purpose as above by adding a star. Using graph-based data structures, the path following IPM moves the initial point to a near optimal point of the modified instance in faster $\widetilde{O}(m \log W + n^{1.5} \log^2 W)$ time. Analogously, by solving a Laplacian system in $\widetilde{O}(m \log W)$ time, we get a near-optimal flow of the original instance. Moreover, since the LP for min-cost flow is integral and the optimal solution can be assumed to be unique (using the isolation lemma as in [DS08, BLN+20]), we can round the flow on each edge to its nearest integer and obtain an exactly optimal flow. This takes time $\widetilde{O}(m \log W + n^{1.5} \log^2 W)$ as promised in Theorem 1.4. For the easier maximum flow problem, we can shave a $\log W$ factor on $n^{1.5}$ using a standard scaling technique by [AO91] and obtain Corollary 7.9.

4 IPM

Throughout this section we let $\mathbf{A} \in \mathbb{R}^{m \times n}$ denote a non-degenerate matrix, let $b \in \mathbb{R}^n$, $c \in \mathbb{R}^m$ and consider the following problem

$$\min_{\substack{x \in \mathbb{R}^m : \mathbf{A}^\top x = b \\ \ell_i \leq x_i \leq u_i \forall i \in [m]}} c^\top x.$$

In this paper, we will need for the barrier functions $\phi_i : (\ell_i, u_i) \to \mathbb{R}$ be *highly* 1*-self-concordant* barrier functions on S_i, as opposed to only 1-self-concordant. This is due to needing the fourth derivative in Lemma 4.31 later.

Definition 4.1 (Highly 1-self-concordance)**.** *For an interval (ℓ, u) we say that a function $f : (\ell, u) \to \mathbb{R}$ is a highly 1-self-concordant barrier on (ℓ, u) if for all $x \in (\ell, u)$ we have $|f'(x)| \leq f''(x)^{1/2}$, $|f'''(x)| \leq 2f''(x)^{3/2}$, $|f''''(x)| \leq 6f''(x)^2$, and $\lim_{x \to a} f(x) = \lim_{x \to b} f(x) = +\infty$.*

Lemma 4.2. *For all $\ell \leq u$ the function $\phi(x) = -\log(x - \ell) - \log(u - x)$ is highly 1-self-concordant on the interval (ℓ, u).*

We show this in Section A.1. For the remainder of the section, we fix $\phi_i(x_i) = -\log(x_i - \ell_i) - \log(u_i - x_i)$. This function satisfies a simple bound which is useful for getting the initial and final points.

Fact 4.3. *Consider the barrier function $\phi(x) = -\log(x - \ell) - \log(u - x)$ on the interval $[\ell, u]$. We have $\phi'((\ell + u)/2) = 0$ and $\phi''(x) = 1/(u - x)^2 + 1/(x - \ell)^2 \geq 1/(u - \ell)^2$ for all $x \in [\ell, u]$.*

During the IPM, we maintain tuples (x, s, μ). Given a current point (x, s, μ), we define a define a *weight function* $\tau : \mathbb{R}^m \to \mathbb{R}^m_{>0}$ that governs the central path. Intuitively, $\tau(x)_i$ is the weight on the i-th barrier function ϕ_i. The choice of weight function τ we use for this paper and the central path will be a regularized Lewis weight. It will be convenient to choose the regularizing vector v to have weight at least n/m on each coordinate, while still having low ℓ_1 norm.

Definition 4.4 (Regularized Lewis weights for a matrix)**.** *For $p = 1 - \frac{1}{4\log(4m/n)}$, vector $v \in \mathbb{R}^m_{>0}$ with $v_i \geq n/m$ for all i and $\|v\|_1 \leq 4n$, and matrix $\mathbf{A}$ define the (v-regularized) ℓ_p-Lewis weights $w(\mathbf{A}) \in \mathbb{R}^m_{>0}$ as the solution to*

$$w(\mathbf{A}) = \sigma(\mathbf{W}^{\frac{1}{2} - \frac{1}{p}} \mathbf{A}) + v \text{ where } \mathbf{W} \stackrel{\text{def}}{=} \mathbf{diag}(w(\mathbf{A})) \ .$$

When the matrix $\mathbf{A}$ is clear from context, we suppress the notation of $\mathbf{A}$ in $w(\cdot)$.

Definition 4.5 (Regularized Lewis weights for c)**.** *For $p = 1 - \frac{1}{4\log(4m/n)}$, and $c, v \in \mathbb{R}^m_{>0}$ define the (v-regularized) ℓ_p-Lewis weights $w(c) : \mathbb{R}^m_{>0} \to \mathbb{R}^m_{>0}$ as $w(c) \stackrel{\text{def}}{=} w(\mathbf{CA})$ as in Definition 4.4.*

We collect properties of regularized Lewis weights in Section 4.3, e.g. that $\|w(c)\|_1 = n + \|v\|_1$. We implicitly suppress the dependence on v, p as they are fixed throughout the algorithm.

Definition 4.6 (Central path weights)**.** *Define the central path weights $\tau(x) \stackrel{\text{def}}{=} w(\phi''(x)^{-\frac{1}{2}})$ for a fixed vector v.*

Our algoirthm maintains points (x, s, μ) satisfying the following centrality guarantee.

Definition 4.7 (ε-centered point)**.** *We say that $(x, s, \mu) \in \mathbb{R}^m \times \mathbb{R}^m \times \mathbb{R}^m_{>0}$ is ε-centered for $\varepsilon \in (0, 1/80]$ if the following properties hold, where $C_{\text{norm}} = C/(1 - p)$ for a constant $C \geq 100$.*

1. *(Approximate centrality)* $\left\|\frac{s+\mu\tau(x)\phi'(x)}{\mu\tau(x)\sqrt{\phi''(x)}}\right\|_\infty \leq \varepsilon$.
2. *(Dual Feasibility) There exists a vector* $z \in \mathbb{R}^n$ *with* $\mathbf{A}z + s = c$.
3. *(Approximate Feasibility)* $\|\mathbf{A}^\top x - b\|_{(\mathbf{A}^\top(\mathbf{T}(x)\Phi''(x))^{-1}\mathbf{A})^{-1}} \leq \varepsilon\gamma/C_{\text{norm}}$.

To maintain approximate centrality in Definition 4.7, we will track a centrality potential.

Definition 4.8 (Centrality potential)**.** *We track the following centrality potential.*

$$\Psi(x, s, \mu) \stackrel{\text{def}}{=} \sum_{i=1}^{m} \psi\left(\frac{s_i + \mu\tau(x)_i\phi_i'(x_i)}{\mu\tau(x)_i\sqrt{\phi_i''(x_i)}}\right)$$

for $\psi(x) \stackrel{\text{def}}{=} \cosh(\lambda x)$*, where* $\lambda = \Theta(\log(m)/\varepsilon)$.

At a high level, this potential is derived from noting that $s + \mu\tau(x)\phi'(x) = 0$ for exactly central points x, s. The denominators are the Hessians of x with respect to the barriers, and thus capture changes of x, s within a small stable region.

Finally, we need to control both the τ and ∞ norms of the steps in the algorithm, which leads to the following definition.

Definition 4.9 ($\tau + \infty$ norm)**.** *Let* $\|g\|_{\tau+\infty} \stackrel{\text{def}}{=} \|g\|_\infty + C_{\text{norm}}\|g\|_\tau$ *for* $C_{\text{norm}} = C/(1-p)$ *and*

$$g^{\flat(\tau)} \stackrel{\text{def}}{=} \text{argmax}_{\|h\|_{\tau+\infty}=1} h^\top g.$$

Let the dual norm be $\|g\|_{\tau+\infty}^* \stackrel{\text{def}}{=} g^\top g^{\flat(\tau)}$.

As described, our algorithm will not maintain exact points $(x, s, \mu) \in \mathbb{R}^m \times \mathbb{R}^m \times \mathbb{R}^m_{>0}$ and weights $\tau \in \mathbb{R}^m$, instead they will be *approximate* in the following sense. Precisely, the algorithm maintains the following condition throughout.

Invariant 4.10. *We maintain the following approximations* $\overline{x}, \overline{\tau}$ *of* $x, \tau \in \mathbb{R}^m$ *at the start and end of each call to* ShortStep *(Algorithm 1).*

- $\|\Phi''(x)^{\frac{1}{2}}(\overline{x} - x)\|_\infty \leq \varepsilon$.
- $\|\mathbf{T}(\overline{x})^{-1}(\overline{\tau} - \tau(\overline{x}))\|_\infty \leq \varepsilon$.

Constants and approximation notation. We will use C to denote a large constant, chosen later. It is used in the definition of C_{norm} and for the parameters $\varepsilon, \gamma, \lambda$ in Algorithm 1. For quantities f, g we write $f \lesssim g$ or $f = O(g)$ if there is a universal constant Z (independent of the constant C) such that $|f| \leq Z|g|$. We assume that C is chosen large enough in Algorithm 1 so that for any quantity f written in the analysis satisfying $f \lesssim \varepsilon$ in fact satisfies $f \leq 1/1000$. Also, if $f \lesssim \gamma$ then, because $\gamma = \varepsilon/(C\lambda)$, we will assume similarly that in fact $f \leq \frac{1}{1000\lambda}$.

We are ready to state the IPM. Algorithm 1 takes a single step, and Algorithm 2 takes a sequence of short steps to solve a linear program. Taking a sequence of short steps using Algorithm 1 allows us to solve LPs, assuming we have an initial point. The initial point construction is done formally in Section A.3, and final point is computed in Lemma 4.11, proven in the appendix in Section A.3.

Lemma 4.11 (Final point)**.** *Given an* ε*-centered point* (x, s, μ) *where* $\varepsilon \leq 1/80$*, we can compute a point* $(x^{(\text{final})}, s^{(\text{final})})$ *satisfying*

1. $\mathbf{A}^\top x^{(\text{final})} = b$, $s^{(\text{final})} = \mathbf{A}y + c$ *for some* y.
2. $c^\top x^{(\text{final})} - \min_{\substack{\mathbf{A}^\top x = b \\ \ell_i \leq x_i \leq u_i \forall i}} c^\top x \lesssim n\mu$.

Algorithm 1: Short Step (Lee Sidford Barrier)

procedure *ShortStep*$(x, s, \mu, \mu^{(\text{new})})$

Fix $\tau(x) = w(\phi''(x)^{-\frac{1}{2}})$, where v and w are defined in Definition 4.4 and 4.5.

Let $\alpha = \frac{1}{4\log(4m/n)}, \varepsilon = \frac{\alpha}{C}, \lambda = \frac{C\log(Cm/\varepsilon^2)}{\varepsilon}, \gamma = \frac{\varepsilon}{C\lambda}, r = \frac{\varepsilon\gamma}{C_{\text{norm}}\sqrt{n}}$.

Assume that (x, s, μ) is ε-centered and $\delta_\mu \stackrel{\text{def}}{=} \mu^{(\text{new})} - \mu$ satisfies $|\delta_\mu| \le r\mu$.

Pick $(\overline{x}, \overline{\tau})$ to satisfy Invariant 4.10 with respect to $(x, \tau(x))$.

Let $y = \frac{s_i + \mu\tau(x)_i\phi_i'(x_i)}{\mu\tau(x)_i\sqrt{\phi_i''(x_i)}}$ and let $\|\overline{y} - y\|_\infty \le \gamma/20$.

Let $g = -\gamma\nabla\Psi(\overline{y})^{\flat(\overline{\tau})}$, where $h^{\flat(\overline{\tau})}$ is defined in Definition 4.9.

Let $\mathbf{H} \approx_\gamma \overline{\mathbf{A}}^\top\overline{\mathbf{A}} = \mathbf{A}^\top\overline{\mathbf{T}}^{-1}\Phi''(\overline{x})^{-1}\mathbf{A}$, where $\overline{\mathbf{A}} = \overline{\mathbf{T}}^{-\frac{1}{2}}\Phi''(\overline{x})^{-\frac{1}{2}}\mathbf{A}$.

Let $\delta_1 = \overline{\mathbf{T}}^{-1}\Phi''(\overline{x})^{-\frac{1}{2}}\mathbf{A}\mathbf{H}^{-1}\mathbf{A}^\top\Phi''(\overline{x})^{-\frac{1}{2}}g$ and $\delta_2 = \overline{\mathbf{T}}^{-1}\Phi''(\overline{x})^{-\frac{1}{2}}\mathbf{A}\mathbf{H}^{-1}(\mathbf{A}^\top x - b)$.

Let $\delta_r = \delta_1 + \delta_2$.

Let $\mathbf{R} \in \mathbb{R}^{m\times m}$ be a C_{valid}-valid random diagonal matrix for large C_{valid} chosen later. `// Definition 4.13`

$\overline{\delta}_x \leftarrow \Phi''(\overline{x})^{-\frac{1}{2}}(g - \mathbf{R}\delta_r)$.

$\overline{\delta}_s \leftarrow \mu\overline{\mathbf{T}}\Phi''(\overline{x})^{\frac{1}{2}}\delta_1$.

$x^{(\text{new})} \leftarrow x + \overline{\delta}_x$ and $s^{(\text{new})} \leftarrow s + \overline{\delta}_s$.

return $(x^{(\text{new})}, s^{(\text{new})})$.

Algorithm 2: Path Following Meta-Algorithm for solving $\min_{\mathbf{A}^\top x = b, \ell_i \le x_i \le u_i \forall i} c^\top x$, given an initial point $\varepsilon/C_{\text{start}}$-centered point $(x^{(\text{init})}, s^{(\text{init})}, \mu)$ for large C_{start}.

procedure *PathFollowing*$(\mathbf{A}, \ell, u, \mu, \mu^{(\text{final})})$

Define r as in Algorithm 1.

while $\mu > \mu^{(\text{final})}$ **do**

  $(x^{(\text{new})}, s^{(\text{new})}) \leftarrow$ ShortStep$(x, s, \mu, (1-r)\mu)$.

  $x \leftarrow x^{(\text{new})}, s \leftarrow s^{(\text{new})}, \mu \leftarrow (1-r)\mu$.

Use Lemma 4.11 to return a point $(x^{(\text{final})}, s^{(\text{final})})$.

The algorithm takes $O(\text{nnz}(\mathbf{A}))$ time plus the time for solving a linear system on $\mathbf{A}^\top\mathbf{D}\mathbf{A}$ where $\mathbf{D}$ is a diagonal matrix.

The main goal of Sections 4.1 to 4.5 is to show the following, proven formally at the end of Section 4.5.

Lemma 4.12. *Algorithm PathFollowing$(\mathbf{A}, b, \ell, u, c, \mu, \mu^{(\text{final})})$ makes $\widetilde{O}(\sqrt{n}\log(\mu/\mu^{(\text{final})}))$ calls to ShortStep$(\cdot)$, and with probability at least $1 - m^{-5}$ satisfies the following conditions at the start and end of each call to ShortStep (Algorithm 1).*

1. *(Slack feasibility) $s = \mathbf{A}z + c$ for some vector $z \in \mathbb{R}^n$*
2. *(Approximate feasibility) $\|\mathbf{A}^\top x - b\|_{(\mathbf{A}^\top(\mathbf{T}(x)\Phi''(x))^{-1}\mathbf{A})^{-1}} \le \varepsilon\gamma/C_{\text{norm}}$.*
3. *(Potential function) $\mathbb{E}[\Psi(x, s, \mu)] \le m^2$, where the expectation is over the randomness of x, s.*
4. *(ε-centered) (x, s, μ) is ε-centered.*

For $\mu^{(\text{final})} \le \delta/(Cn)$, we have that $\mathbf{A}x^{(\text{final})} = b$ and $c^\top x^{(\text{final})} \le \min_{\substack{\mathbf{A}^\top x = b \\ \ell_i \le x_i \le u_i \forall i}} c^\top x + \delta$.

We sample a random diagonal scaling $\mathbf{R}$ in our algorithm, and will require some properties of this random matrix to guarantee progress of the IPM. We summarize the necessary properties

here. This definition captures distributions such as sampling each coordinate independently as a Bernoulli with probabilities p_i, or taking the sum of multiple samples proportional to p_i.

Definition 4.13 (Valid sampling distribution)**.** *Given vector $\delta_r, \mathbf{A}, \overline{x}, \overline{\tau}$ as in ShortStep (Algorithm 1), we say that a random diagonal matrix $\mathbf{R} \in \mathbb{R}^{m \times m}$ is C_{valid}-*valid *if it satisfies the following properties, for $\overline{\mathbf{A}} = \overline{\mathbf{T}}^{-\frac{1}{2}} \Phi''(\overline{x})^{-\frac{1}{2}} \mathbf{A}$. We assume that $C_{\text{valid}} \geq C_{\text{norm}}$.*

- *(Expectation) We have that $\mathbb{E}[\mathbf{R}] = \mathbf{I}$.*
- *(Variance) For all $i \in [m]$, we have that $\text{Var}[\mathbf{R}_{ii}(\delta_r)_i] \leq \frac{\gamma |(\delta_r)_i|}{C_{\text{valid}}^2}$. and $\mathbb{E}[\mathbf{R}_{ii}^2] \leq 2\sigma(\overline{\mathbf{A}})_i^{-1}$.*
- *(Covariance) For all $i \neq j$, we have that $\mathbb{E}[\mathbf{R}_{ii}\mathbf{R}_{jj}] \leq 2$.*
- *(Maximum) With probability at least $1 - n^{-10}$ we have that $\|\mathbf{R}\delta_r - \delta_r\|_\infty \leq \frac{\gamma}{C_{\text{valid}}^2}$.*
- *(Matrix approximation) We have that $\overline{\mathbf{A}}^\top \mathbf{R} \overline{\mathbf{A}} \approx_\gamma \overline{\mathbf{A}}^\top \overline{\mathbf{A}}$ with probability at least $1 - n^{-10}$.*

4.1 Overview of Analysis

Our proof will show that the expected value of the potential function in Definition 4.8 is bounded by $\text{poly}(m)$ with high probability throughout the algorithm. This will imply that it is ε-centered at the start and end of each call to ShortStep (Algorithm 1). The main pieces of the analysis are as follows.

Potential function analysis. In Section 4.2 we set up the analysis of the change in the potential function. Specifically, in Lemma 4.15 we show that bounding the change in the potential function reduces to bounding first and second order changes of the numerator $y = s + \mu\tau(x)\phi'(x)$ and denominator terms $\mu, \tau(x), \phi''(x)$ in Definition 4.8. These are done in Section 4.5 in Lemmas 4.27 (for μ), 4.31 (for $\mathbf{C} \stackrel{\text{def}}{=} \Phi''(x)^{-1/2}$), 4.35 (for τ), 4.37 (for y).

The analysis of change in τ requires several facts of derivatives of regularized Lewis weights with respect to diagonal scalings, and these are done in Section 4.3, culminating in Lemmas 4.25, 4.26 which bound the change in τ under small changes in the diagonal scaling, which we refer to as γ-bounded changes (Definition 4.23).

Feasibility. To guarantee efficiency, our algorithm sparsifies matrices $\mathbf{A}^\top \mathbf{D} \mathbf{A}$ to solve systems, which results in potentially infeasible points x during the algorithm, i.e. $\mathbf{A}^\top x = b$ may fail. However, our algorithm maintains approximate feasibility as discussed in Definition 4.7. The analysis is done in Lemma 4.38.

Additional properties. In Section A.3 we first show that computing an ε-centered point for small path parameter μ guarantees small objective error. We then show how to construct an initial ε-centered point for a perturbed linear program, and show that the modified linear program still gives approximate solutions to the original. In Section 4.6 we show that two sampling schemes are both valid distributions. The one in Lemma 4.41 is used for the graphical setting, and Lemma 4.42 is used for the general linear program setting. Finally, the data structures for maintaining approximate solutions for x, s, τ require a stability bound of the true x, s which may not hold. However, we can show that there are nearby points that satisfy stronger stability bounds in Lemma 4.44.

Comparison to [BLN+20]. The main difference from the analysis of [BLN+20] is our use of general self-concordant functions to handle two-sided barrier constraints, while [BLN+20] used logarithmic barriers. This leads to the following differences in the IPM: the gradient optimality for our barrier takes the form in Definition 4.7 (Approximate centrality), while in [BLN+20] they simply use the form $xs \approx w\mu$. Additionally, we require our weights w to be Lewis weights, while [BLN+20] was able to use leverage scores due to the structure of the centrality condition $xs \approx w\mu$. Finally, our analysis deals more generally with valid distributions in Definition 4.13 which allows us to handle both sampling coordinate independently for the graphical setting, and proportional to sampling probability for general linear programs.

4.2 Analysis Tools and Setup

In this section, we set up the analysis of our IPM, and all omitted proofs are given in Section A.1. First, we collect some basic properties of ψ to help in the analysis.

Lemma 4.14 (Basic properties of ψ)**.** *We have for $\lambda \geq 1$ that*

- $\psi''(x) = \lambda^2 \psi(x)$.
- $\psi(x') \leq 2\psi(x)$ *for* $|x' - x| \leq \frac{1}{20\lambda}$.
- $|\psi'(x)| \leq \lambda^{-1}\psi''(x)$.

We now state a helper lemma that shows that we can analyze the change in the centrality potential by analyzing second order changes of each contributing piece. This differs from the corresponding [BLN+20, Lemma 4.34] in that our errors are not strictly multiplicative errors.

Lemma 4.15 (Potential change bound)**.** *Define for $u_i^{(j)} \geq 0$ and y_i*

$$w_i = \prod_{j \in [k]} (u_i^{(j)})^{c_j} \quad \text{and} \quad w_i^{(\text{new})} = \prod_{j \in [k]} (u_i^{(j)} + \delta_i^{(j)})^{c_j}$$

and

$$v_i = y_i w_i \quad \text{and} \quad v_i^{(\text{new})} = (y_i + \eta_i) w_i^{(\text{new})}$$

where $\|(\mathbf{U}^{(j)})^{-1}\delta^{(j)}\|_\infty \leq \frac{1}{50(1+\|c\|_1)}$ for all $j \in [k]$, $\|v\|_\infty \leq 1/50$, $\|\mathbf{W}\eta\|_\infty \leq \frac{1}{50\lambda(1+\|c\|_1)}$, and

$$\|v\|_\infty \sum_{j \in [k]} |c_j| \|(\mathbf{U}^{(j)})^{-1}\delta^{(j)}\|_\infty \leq \frac{1}{100\lambda}.$$

Then we have that $\|v^{(\text{new})} - v\|_\infty \leq \frac{1}{20\lambda}$

$$\Psi(v^{(\text{new})}) \leq \Psi(v) + \psi'(v)^\top \left(\mathbf{W}\eta + \sum_{j \in [k]} c_j \mathbf{V}(\mathbf{U}^{(j)})^{-1}\delta^{(j)} \right) \tag{13}$$

$$+ 8\|\mathbf{W}\eta\|^2_{\psi''(v)} + 8(1 + \|c\|_1)\|v\|^2_\infty \sum_{j \in [k]} |c_j| \|(\mathbf{U}^{(j)})^{-1}\delta^{(j)}\|^2_{\psi''(v)} \tag{14}$$

$$+ 8\|\mathbf{W}\eta\|_{|\psi'(v)|} \sum_{j \in [k]} |c_j| \|(\mathbf{U}^{(j)})^{-1}\delta^{(j)}\|_{|\psi'(v)|} + 8(1 + \|c\|_1)\|v\|_\infty \sum_{j \in [k]} |c_j| \|(\mathbf{U}^{(j)})^{-1}\delta^{(j)}\|^2_{|\psi'(v)|}. \tag{15}$$

It is known that the Hessian $\Phi''(x)$ is maintained under small perturbations to x.

Lemma 4.16. *$\|\Phi''(x)^{\frac{1}{2}}(\overline{x} - x)\|_\infty \leq \varepsilon$ for $\varepsilon \in [0, 1/100]$ then $\Phi''(x)^{\frac{1}{2}} \approx_{\varepsilon + 2\varepsilon^2} \Phi''(\overline{x})^{\frac{1}{2}}$.*

Proof. Follows directly from $\left| \frac{\mathrm{d}}{\mathrm{d}x_i} \phi_i''(x_i)^{-\frac{1}{2}} \right| \leq 1$. □

In order the analyze the potential Ψ under changes, we define the quantities $\tau_i \stackrel{\text{def}}{=} \tau(x)_i$, $\mu_i = \mu$, $c_i = \phi_i''(x_i)^{-\frac{1}{2}}$, and $y_i \stackrel{\text{def}}{=} s_i + \mu\tau(x)_i \phi_i'(x_i)$. This allows us to write the potential as

$$\Psi(x, s, \mu) = \sum_{i=1}^m \psi(y_i \mu_i^{-1} \tau_i^{-1} c_i).$$

Let $\tau^{(\text{new})}, \mu^{(\text{new})}, c^{(\text{new})}, y^{(\text{new})}$ be the corresponding vectors after a step. We will use Lemma 4.15 to analyze the change in Ψ. Define

$$\delta_\mu \stackrel{\text{def}}{=} \mu^{(\text{new})} - \mu, \delta_\tau \stackrel{\text{def}}{=} \tau^{(\text{new})} - \tau, \delta_c \stackrel{\text{def}}{=} c^{(\text{new})} - c, \delta_y \stackrel{\text{def}}{=} y^{(\text{new})} - y.$$

The goal of the remainder of the section will be to analyze the changes $\delta_\mu, \delta_\tau, \delta_c, \delta_y$ and apply Lemma 4.15 to analyze the change in the centrality potential. This will complete the correctness proof of the IPM.

4.3 Regularized Lewis Weights

Statement	Term	Comment
Definition 4.17	$\mathbf{W}_c, \mathbf{P}_c, \mathbf{\Sigma}_c, \mathbf{\Lambda}_c, \mathbf{J}_c$	Fundamental matrices
Lemma 4.18	$w(c), f(w,c)$	Alternative definition of regularized Lewis weights
Lemma 4.19	$\mathbf{J}_c$	Jacobian of Regularized Lewis weight
Lemma 4.20	$\mathbf{D}_c, \mathbf{K}_c$	Decomposition of $\mathbf{J}_c$
Lemma 4.21	$\mathbf{D}'_c, \mathbf{K}'_c$	Alternate decomposition
Lemma 4.22	$\mathbf{W}_c, \mathbf{T}$	Lewis weight approximation
Definition 4.23	$\mathbf{C}$	γ-boundedness
Lemma 4.24	$\mathbf{T}^{-1}\delta_{\tau_t}$	$\|\cdot\|_\infty$, $\|\cdot\|_{\tau+\infty}$, $\|\cdot\|_{\mathbf{P}_i^{(2)}}$; infinitesimal bound
Lemma 4.25	$\mathbf{T}^{-1}\delta_\tau$	$\|\cdot\|_\infty$, $\|\cdot\|_{\tau+\infty}$ norm
Lemma 4.26	$\mathbf{T}^{-1}(\mathbb{E}[\delta_\tau] - \mathbf{J}\mathbb{E}[\delta_c])$	$\|\cdot\|_{\tau+\infty}$

Table 3: Summary of Section 4.3

Parameters	Definition	Range
C	Large constant, chosen later	$[200, \infty)$
α	$1/(4\log(4m/n))$	$< 1/2$
p	$1-\alpha$	$[2/3, 1)$
ε	α/C	$[0, 1/100]$
λ	$C\log(Cm/\varepsilon^2)/\varepsilon$	$\geq \varepsilon^{-1}$
γ	$\varepsilon/(C\lambda)$	$(0, \varepsilon)$
C_{norm}	C/α	≥ 100
C_{valid}	Constant in Definition 4.13	$\geq C_{\text{norm}}$
r	$\varepsilon\gamma/(C_{\text{norm}}\sqrt{n})$	$\leq n^{-1/2}$

In this section we collect several facts about the regularized Lewis weights defined in Definition 4.5, many of which are variations of those in [LS19]. All omitted proofs are provided in Section A.2. Before starting, we set up notation for important matrices throughout.

Definition 4.17 (Fundamental matrices)**.** *We define the matrices*

- $\mathbf{W}_c \stackrel{\text{def}}{=} \mathbf{diag}(w(c))$.
- *For any matrix* $\mathbf{M}$*, orthogonal projection matrix* $\mathbf{P}(\mathbf{M}) \stackrel{\text{def}}{=} \mathbf{M}(\mathbf{M}^\top\mathbf{M})^{-1}\mathbf{M}^\top$.
- *The projection matrix* $\mathbf{P}_c \stackrel{\text{def}}{=} \mathbf{P}(\mathbf{W}_c^{\frac{1}{2}-\frac{1}{p}}\mathbf{CA})$.
- $\sigma_c \stackrel{\text{def}}{=} \sigma(\mathbf{W}_c^{\frac{1}{2}-\frac{1}{p}}\mathbf{CA})$ *and* $\mathbf{\Sigma}_c \stackrel{\text{def}}{=} \mathbf{diag}(\sigma_c)$.
- $\mathbf{\Lambda}_c \stackrel{\text{def}}{=} \mathbf{\Sigma}_c - \mathbf{P}_c^{(2)}$ *and* $\overline{\mathbf{\Lambda}}_c = \mathbf{W}_c^{-\frac{1}{2}}\mathbf{\Lambda}_c\mathbf{W}_c^{-\frac{1}{2}}$.
- $\mathbf{J}_c$ *as the Jacobian of* $w(c)$ *with respect to* c.

We now describe the regularized Lewis weights as the solution to a convex program.

Lemma 4.18 (Alternate definition of regularized Lewis weights)**.** *For all non-negative* c

$$w(c) = \text{argmin}_{w\in\mathbb{R}^m_{>0}} f(w,c)$$

where

$$f(w,c) \stackrel{\text{def}}{=} -\frac{1}{1-\frac{2}{p}}\log\det(\mathbf{A}^\top\mathbf{C}\mathbf{W}^{1-\frac{2}{p}}\mathbf{CA}) + \sum_{i=1}^m w_i - \sum_{i=1}^m v_i\log w_i.$$

Using the convex program in Lemma 4.18 we can compute the Jacobian of changes in the ℓ_p regularized Lewis weights with respect to the diagonal scaling.

Lemma 4.19 (Jacobian of Regularized Lewis weight)**.** *For a fixed vector v, we have that*

$$\mathbf{J}_c = 2\mathbf{W}_c \left(\mathbf{W}_c - \left(1 - \frac{2}{p}\right)\mathbf{\Lambda}_c\right)^{-1} \mathbf{\Lambda}_c \mathbf{C}^{-1}.$$

It will be useful to decompose the matrices relating to $\overline{\mathbf{\Lambda}}_c$ into two parts – one of which is diagonal, and one of which is bounded by $\mathbf{P}_c^{(2)}$ essentially.

Lemma 4.20 (Decomposition of $\mathbf{J}_c$)**.** *For any vector $c \in \mathbb{R}^m_{>0}$, there is a diagonal matrix $\mathbf{0} \preceq \mathbf{D}_c \preceq \mathbf{I}$ such that for $\mathbf{K}_c = \mathbf{W}_c^{-1}\mathbf{J}_c\mathbf{C} - \mathbf{D}_c$, we have for all vectors h that*

- $\|\mathbf{K}_c h\|_\infty \lesssim \|h\|_\infty$.
- $\|\mathbf{K}_c h\|_{w(c)} \lesssim \|\|h\|\|_{\mathbf{P}_c^{(2)}}$.

We will need a similar decomposition for a related matrix.

Lemma 4.21 (Alternate decomposition)**.** *In the notation of Lemma 4.20, there is a diagonal matrix $\mathbf{D}'_c$ and matrix $\mathbf{K}'_c$ such that $\mathbf{0} \preceq \mathbf{D}'_c \preceq \mathbf{I}$,*

$$\mathbf{W}_c^{-\frac{1}{2}} \left(\mathbf{I} - \left(1 - \frac{2}{p}\right)\overline{\mathbf{\Lambda}}_c\right)^{-1} \mathbf{W}_c^{\frac{1}{2}} = \mathbf{D}'_c + \mathbf{K}'_c,$$

and for all vectors h,

- $\|\mathbf{K}'_c h\|_{w(c)} \lesssim \|\|h\|\|_{\mathbf{P}_c^{(2)}}$
- $\|\mathbf{K}'_c h\|_\infty \lesssim \|h\|_\infty$.

We note that small perturbations to the diagonal scaling also have a small effect on our ℓ_p Lewis weights. This is useful because our algorithm only maintains approximate x, s, τ.

Lemma 4.22 (Lewis weight approximation)**.** *Let $p \in (0, 4)$. If $\overline{\mathbf{C}} \approx_\varepsilon \mathbf{C}$ then $w_p(\mathbf{CA}) \approx_{4\varepsilon} w_p(\overline{\mathbf{C}}\mathbf{A})$.*

Notation for analysis of δ_τ. The remainder of the section is devoted to understanding the changes in the regularized Lewis weights under (random) changes to c. The notation used is as follows. We will consider the definition $\delta_c = c^{(\text{new})} - c$ and $c_t = c + t\delta_c$, and let $\mathbf{\Sigma}_t, \mathbf{P}_t, \mathbf{\Lambda}_t, \mathbf{J}_t, \overline{\mathbf{\Lambda}}_t$ denote the corresponding fundamental matrices defined in Definition 4.17 for $c := c_t$. Additionally, we write $\mathbf{T}_t$ for $\mathbf{W}_{c_t}$, and let $\mathbf{K}_t, \mathbf{D}_t, \mathbf{K}'_t, \mathbf{D}'_t$ be the matrices resulting in Lemmas 4.20 and 4.21.

To give good bounds on δ_τ we will to assume bounds on δ_c. Our necessary conditions are summarized as follows.

Definition 4.23 (γ-boundedness)**.** *Let $c \in \mathbb{R}^m_{>0}$ be a deterministic vector, and $c^{(\text{new})} \in \mathbb{R}^m_{>0}$ be a stochastic vector. Let $\delta_c = c^{(\text{new})} - c$. We say that the δ_c is* γ-bounded *if the following conditions hold.*

1. *With probability* 1 *we have*

$$\|\mathbf{C}^{-1}\delta_c\|_\infty \lesssim \gamma. \tag{16}$$

2.

$$\left\|\mathbb{E}[(\mathbf{C}^{-1}\delta_c)^2]\right\|_{\tau+\infty} \lesssim \gamma^2. \tag{17}$$

3. For all $t \in [0,1]$, we have

$$\mathbb{E}[\|\mathbf{C}^{-1}|\delta_c|\|_{\mathbf{P}_t^{(2)}}] \lesssim \gamma / C_{\text{norm}}. \tag{18}$$

Note that if δ_c is γ-bounded, then for $c_t = c + t\delta_c$, we have $\mathbf{C}_t \approx_{1.1} \mathbf{C}$ for all $0 \le t \le 1$.

Before moving to bounds on δ_τ, we first show some preliminary bounds on the norms of derivatives of τ locally.

Lemma 4.24 (δ_τ infinitesimal bound)**.** *Let $\delta_c = c^{(\text{new})} - c$ be a γ-bounded change, $c_t = c + t\delta_c$, and $\tau_t = w_p(c_t)$. Let $\delta_{\tau_t} = \frac{\mathrm{d}}{\mathrm{d}t}\tau_t$. Then*

- $\|\mathbf{T}^{-1}\delta_{\tau_t}\|_\infty \lesssim \gamma$ *with probability* 1.
- $\|\mathbb{E}[(\mathbf{T}^{-1}\delta_{\tau_t})^2]\|_{\tau+\infty} \lesssim \gamma^2$.
- $\mathbb{E}[\|\mathbf{T}^{-1}|\delta_{\tau_t}|\|_{\mathbf{P}_t^{(2)}}] \lesssim \gamma / C_{\text{norm}}$.

Proof. Before beginning the proof, we note that $\tau_t \approx \tau$ for all $t \in [0,1]$ by Lemma 4.22, and because δ_c is a γ-bounded change (Definition 4.23 (16)), so $c_t \approx_{0.1} c$.

For the first claim we write

$$\begin{aligned}
\|\mathbf{T}^{-1}\delta_{\tau_t}\|_\infty &\lesssim \|\mathbf{T}_t^{-1}\mathbf{J}_t\delta_c\|_\infty \\
&= \|\mathbf{T}_t^{-1}\mathbf{J}_t\mathbf{C}_t\mathbf{C}_t^{-1}\delta_c\|_\infty \\
&\lesssim \|\mathbf{D}_t\mathbf{C}_t^{-1}\delta_c\|_\infty + \|\mathbf{K}_t\mathbf{C}_t^{-1}\delta_c\|_\infty \\
&\le (\|\mathbf{D}_t\|_\infty + \|\mathbf{K}_t\|_\infty)\|\mathbf{C}_t^{-1}\delta_c\|_\infty \lesssim \gamma,
\end{aligned}$$

where the second step follows from $\mathbf{I} = \mathbf{C}_t\mathbf{C}_t^{-1}$, the third step follows from triangle inequality and $\mathbf{T}_t^{-1}\mathbf{J}_t\mathbf{C}_t = \mathbf{D}_t + \mathbf{K}_t$ as in Lemma 4.20, the fourth step follows from $\|\mathbf{D}_t\mathbf{C}_t^{-1}\delta_c\|_\infty \le \|\mathbf{D}_t\|_\infty \cdot \|\mathbf{C}_t^{-1}\delta_c\|_\infty$, and the last step follows from $\|\mathbf{D}_t\|_\infty, \|\mathbf{K}_t\|_\infty \lesssim 1$ and $\|\mathbf{C}_t^{-1}\delta_c\|_\infty \lesssim \gamma$ (Definition 4.23 (16)).

Because $\mathbf{T}_t^{-1}\mathbf{J}_t\mathbf{C}_t = \mathbf{D}_t + \mathbf{K}_t$ as in Lemma 4.20, we can write

$$\begin{aligned}
\mathbb{E}[(\mathbf{T}^{-1}\delta_{\tau_t})_i^2] &\lesssim \mathbb{E}[(\mathbf{T}_t^{-1}\mathbf{J}_t\delta_c)_i^2] \\
&\le \mathbb{E}[(\mathbf{D}_t\mathbf{C}_t^{-1}\delta_c + \mathbf{K}_t\mathbf{C}_t^{-1}\delta_c)_i^2] \\
&\lesssim \mathbb{E}[(\mathbf{D}_t\mathbf{C}_t^{-1}\delta_c)_i^2] + \mathbb{E}[(\mathbf{K}_t\mathbf{C}_t^{-1}\delta_c)_i^2].
\end{aligned}$$

For the first term, we bound

$$\left\|\mathbb{E}[(\mathbf{D}_t\mathbf{C}_t^{-1}\delta_c)^2]\right\|_{\tau+\infty} \lesssim \left\|\mathbb{E}[(\mathbf{C}^{-1}\delta_c)^2]\right\|_{\tau+\infty} \lesssim \gamma^2$$

by Lemma 4.20 ($\mathbf{D}_t$ is diagonal and $\lesssim \mathbf{I}$) and γ-boundedness (Definition 4.23 (17)). For the second term, we use Lemma 4.20 item 1 and γ-boundedness (Definition 4.23 (16)) to first bound

$$|(\mathbf{K}_t\mathbf{C}_t^{-1}\delta_c)_i| \lesssim \|\mathbf{C}_t^{-1}\delta_c\|_\infty \lesssim \gamma.$$

This handles the ∞-norm part directly. For the τ-norm, we compute

$$\begin{aligned}
\left\|\mathbb{E}[(\mathbf{K}_t\mathbf{C}_t^{-1}\delta_c)^2]\right\|_\tau &\lesssim \gamma\mathbb{E}[\|\mathbf{K}_t\mathbf{C}_t^{-1}\delta_c\|_\tau] \\
&\lesssim \gamma\mathbb{E}[\|\mathbf{C}_t^{-1}|\delta_c|\|_{\mathbf{P}_t^{(2)}}] \\
&\lesssim \gamma\mathbb{E}[\|\mathbf{C}^{-1}|\delta_c|\|_{\mathbf{P}_t^{(2)}}] \lesssim \gamma^2 / C_{\text{norm}}
\end{aligned}$$

where the first step follows from $\|\mathbf{K}_t\mathbf{C}_t^{-1}\delta_c\|_\infty \lesssim \gamma$, the second step follows from Lemma 4.20 item 2, the third step follows from $c \approx_1 c_t$, the last step follows from γ-boundedness (Definition 4.23 (18)).

Summing the contributions gives the desired.
For the third claim, we use $\mathbf{T}_t^{-1}\mathbf{J}_t\mathbf{C}_t = \mathbf{D}_t + \mathbf{K}_t$ as in Lemma 4.20 to get

$$\begin{aligned}\mathbb{E}[\||\mathbf{T}^{-1}|\delta_{\tau_t}|\|_{\mathbf{P}_t^{(2)}}] &\lesssim \mathbb{E}[\||\mathbf{T}_t^{-1}\mathbf{J}_t\mathbf{C}_t\mathbf{C}_t^{-1}\delta_c|\|_{\mathbf{P}_t^{(2)}} \\ &\le \mathbb{E}[\||\mathbf{D}_t|\mathbf{C}_t^{-1}\delta_c|\|_{\mathbf{P}_t^{(2)}}] + \mathbb{E}[\||\mathbf{K}_t\mathbf{C}_t^{-1}\delta_c|\|_{\mathbf{P}_t^{(2)}}] \\ &\lesssim \gamma/C_{\text{norm}} + \mathbb{E}[\|\mathbf{K}_t\mathbf{C}_t^{-1}\delta_c\|_{\tau_t}] \\ &\lesssim \gamma/C_{\text{norm}} + \mathbb{E}[\|\mathbf{C}_t^{-1}|\delta_c|\|_{\mathbf{P}_t^{(2)}}] \lesssim \gamma/C_{\text{norm}},\end{aligned}$$

where the first step follows from $\tau \approx_1 \tau_t$, the second step follows from the triangle inequality, the third step follows from $\mathbf{D}_t \lesssim \mathbf{I}$ and γ-boundedness (Definition 4.23 (18)) and $\mathbf{P}_t^{(2)} \lesssim \mathbf{T}_t$, the fourth step follows from Lemma 4.20 item 2, and the last step follows from γ-boundedness (Definition 4.23 (18)) again. □

Finally, we give our full analysis of δ_τ. We first control the infinity norm and square of the change, which have shorter analyses.

Lemma 4.25 (Sharper bound on changes in τ part 1)**.** *Let $\delta_c = c^{(\text{new})} - c$ be a γ-bounded change, and let $\tau^{(\text{new})} = \tau_1$, and $\delta_\tau = \tau^{(\text{new})} - \tau$. We have*

- $\|\mathbf{T}^{-1}\delta_\tau\|_\infty \lesssim \gamma$ *with probability* 1.
- $\left\|\mathbb{E}[(\mathbf{T}^{-1}\delta_\tau)^2]\right\|_{\tau+\infty} \lesssim \gamma^2$.

Proof. To start, we define $c_t = c + t\delta_c$, $\tau_t = w_p(c_t)$. Let $\delta_{\tau_t} = \frac{\mathrm{d}}{\mathrm{d}t}\tau_t$.

Bound on $\|\mathbf{T}^{-1}\delta_\tau\|_\infty$. Follows from γ-boundedness (Definition 4.23 (16)) and Lemma 4.22.

Bound on $\|\mathbb{E}[(\mathbf{T}^{-1}\delta_\tau)^2]\|_{\tau+\infty}$. We have

$$\begin{aligned}\|\mathbb{E}[(\mathbf{T}^{-1}\delta_\tau)^2]\|_{\tau+\infty} &= \left\|\mathbb{E}\left[\left(\int_0^1 \mathbf{T}^{-1}\delta_{\tau_t}\mathrm{d}t\right)^2\right]\right\|_{\tau+\infty} \\ &\le \int_0^1 \mathbb{E}[(\mathbf{T}^{-1}\delta_{\tau_t})^2]\mathrm{d}t \lesssim \gamma^2.\end{aligned}$$

where the first step follows from $\delta_\tau = \int_0^1 \delta_{\tau_t} dt$ and Cauchy-Schwarz, the second step follows from Cauchy-Schwarz, and the last step follows from Lemma 4.24 item 2. □

We now analyze the first order change in δ_τ. We defer the proof to the appendix due to its length.

Lemma 4.26 (Sharper bound on changes in τ part 2)**.** *Let $\delta_c = c^{(\text{new})} - c$ be a γ-bounded change, and let $\tau^{(\text{new})} = \tau_1$, and $\delta_\tau = \tau^{(\text{new})} - \tau$. For $\mathbf{J}$ as defined in Lemma 4.19, we have*

$$\left\|\mathbf{T}^{-1}(\mathbb{E}[\delta_\tau] - \mathbf{J}\mathbb{E}[\delta_c])\right\|_{\tau+\infty} \lesssim \gamma^2.$$

4.4 Bounding $\delta_\mu, \delta_\tau, \delta_c, \delta_y$

In this section, we bound $\delta_\mu, \delta_\tau, \delta_c, \delta_y$. First we bound δ_μ. Since the change in μ is a deterministic change, bounding δ_μ is straightforward.

Lemma 4.27 (Bounds on δ_μ)**.** *Let $\delta_\mu = \mu^{(\text{new})} - \mu$. We have that $\|\mu^{-1}(\mu^{(\text{new})} - \mu)\|_{\tau+\infty} \lesssim \varepsilon\gamma$.*

Statement	Term	Comment
Lemma 4.27	δ_μ	Bounds on δ_μ
Lemma 4.29	$\overline{\delta}_x$	Bounds on $\overline{\delta}_x$
Lemma 4.31	δ_c	Bounds on δ_c
Lemma 4.35	δ_τ	Bounds on δ_τ
Lemma 4.37	δ_y	Bounds on δ_y
Lemma 4.33	δ_x, δ_c	Bounds on δ_x and δ_c with respect to $\mathbf{P}_t^{(2)}$-norm
Lemma 4.28	$\Phi''(\overline{x}), \overline{\mathbf{T}}, \mathbf{H}$	Matrix norm bounds
Corollary 4.30	$\overline{\delta}_s, \delta_r$	An application of Lemma 4.29
Lemma 4.32	$\mathbf{R}v$	$\mathbf{P}^{(2)}$ norm bound
Lemma 4.36	g	Approximation to direction g

Table 4: Summary of Section 4.4

Proof. Note that $\|\mu^{-1}(\mu^{(\text{new})} - \mu)\|_\infty \leq r$ by definition. Therefore

$$\begin{aligned}\|\mu^{-1}(\mu^{(\text{new})} - \mu)\|_{\tau+\infty} &\leq r\|\vec{1}\|_{\tau+\infty} \\ &= r(1 + C_{\text{norm}}(n + \|v\|_1)^{1/2}) \\ &\lesssim C_{\text{norm}} r n^{1/2} \leq \varepsilon\gamma\end{aligned}$$

where the first step follows from $\|\mu^{-1}(\mu^{(\text{new})}-\mu)\|_\infty \leq r$, the third step follows from $C_{\text{norm}} \geq 100$ and $\|v\|_1 \lesssim \sqrt{n}$, and the last step follows from $C_{\text{norm}} r n^{1/2} \leq \varepsilon\gamma$ (by the choice of r). □

The following bounds allow us to relate changes in x to $\|g\|_{\tau+\infty}$.

Lemma 4.28 (Matrix norm bounds)**.** *Let $x, \tau, \overline{x}, \overline{\tau}$ satisfy Invariant 4.10. For any $g \in \mathbb{R}^m$ we have that*

- $\|\Phi''(\overline{x})^{-\frac{1}{2}}\overline{\mathbf{T}}^{-1}\mathbf{A}\mathbf{H}^{-1}\mathbf{A}^\top\Phi''(\overline{x})^{-\frac{1}{2}}g\|_\infty \lesssim \|g\|_\tau$.
- $\|\Phi''(\overline{x})^{-\frac{1}{2}}\overline{\mathbf{T}}^{-1}\mathbf{A}\mathbf{H}^{-1}g\|_\tau \lesssim \|g\|_{(\mathbf{A}^\top\mathbf{T}^{-1}\Phi''(x)^{-1}\mathbf{A})^{-1}}$.
- $\|\Phi''(\overline{x})^{-\frac{1}{2}}\overline{\mathbf{T}}^{-1}\mathbf{A}\mathbf{H}^{-1}g\|_\infty \lesssim \|g\|_{(\mathbf{A}^\top\mathbf{T}^{-1}\Phi''(x)^{-1}\mathbf{A})^{-1}}$.

Proof. Define

$$\mathbf{Q} = \overline{\mathbf{T}}^{-\frac{1}{2}}\Phi''(\overline{x})^{-\frac{1}{2}}\mathbf{A}\mathbf{H}^{-1}\mathbf{A}^\top\overline{\mathbf{T}}^{-\frac{1}{2}}\Phi''(\overline{x})^{-\frac{1}{2}}.$$

For the first point, we use the Cauchy-Schwarz inequality to get that

$$\begin{aligned}\|\Phi''(\overline{x})^{-\frac{1}{2}}\overline{\mathbf{T}}^{-1}\mathbf{A}\mathbf{H}^{-1}\mathbf{A}^\top\Phi''(\overline{x})^{-\frac{1}{2}}g\|_\infty &= \max_{i\in[m]}\left|e_i^\top\Phi''(\overline{x})^{-\frac{1}{2}}\overline{\mathbf{T}}^{-1}\mathbf{A}\mathbf{H}^{-1}\mathbf{A}^\top\Phi''(\overline{x})^{-\frac{1}{2}}g\right| \\ &= \max_{i\in[m]}\left|e_i^\top\overline{\mathbf{T}}^{-\frac{1}{2}}\mathbf{Q}\overline{\mathbf{T}}^{\frac{1}{2}}g\right| \\ &\leq \left|g^\top\overline{\mathbf{T}}^{\frac{1}{2}}\mathbf{Q}\overline{\mathbf{T}}^{\frac{1}{2}}g\right|^{\frac{1}{2}}\max_{i\in[m]}\left|e_i^\top\overline{\mathbf{T}}^{-\frac{1}{2}}\mathbf{Q}\overline{\mathbf{T}}^{-\frac{1}{2}}e_i\right|^{\frac{1}{2}} \\ &\lesssim \|g\|_{\overline{\tau}}\max_{i\in[m]}\overline{\tau}_i^{-1}\sigma\left(\overline{\mathbf{T}}^{-\frac{1}{2}}\Phi''(\overline{x})^{\frac{1}{2}}\right) \\ &\lesssim \|g\|_\tau\max_{i\in[m]}\tau(\overline{x})_i^{-1}\sigma\left(\tau(\overline{x})^{\frac{1}{2}-\frac{1}{p}}\Phi''(\overline{x})^{\frac{1}{2}}\right) \lesssim \|g\|_\tau,\end{aligned}$$

where the third step follows from $|a^\top b| \leq \|a\|_2 \cdot \|b\|_2$, where the fifth step follows from the stability of leverage scores and that $p = 1 - \frac{1}{4\log(4m/n)}$.

Let $\overline{\mathbf{H}} \stackrel{\text{def}}{=} \mathbf{A}^\top \overline{\mathbf{T}}^{-1} \overline{\Phi}''(x)^{-1}\mathbf{A}$, so that $\mathbf{H} \approx_\varepsilon \overline{\mathbf{H}}$ and $\overline{\mathbf{H}} \approx_{2\varepsilon} \mathbf{A}^\top \mathbf{T}^{-1}\Phi''(x)^{-1}\mathbf{A}$.

$$\begin{aligned}\|\Phi''(\overline{x})^{-\frac{1}{2}}\overline{\mathbf{T}}^{-1}\mathbf{A}\mathbf{H}^{-1}g\|_\tau &= (g^\top \mathbf{H}^{-1}\overline{\mathbf{H}}\mathbf{H}^{-1}g)^{\frac{1}{2}} \\ &\lesssim (g^\top \overline{\mathbf{H}}^{-1}g)^{\frac{1}{2}} \\ &= \|g\|_{\overline{\mathbf{H}}^{-1}} \lesssim \|g\|_{(\mathbf{A}^\top \mathbf{T}^{-1}\Phi''(x)^{-1}\mathbf{A})^{-1}},\end{aligned}$$

where the second step follows from $\mathbf{H} \approx_\varepsilon \overline{\mathbf{H}}$, and the last step follows from $\overline{\mathbf{H}} \approx_{2\varepsilon} \mathbf{A}^\top \mathbf{T}^{-1}\Phi''(x)^{-1}\mathbf{A}$.

For the third point, we use the Cauchy-Schwarz inequality to get that

$$\begin{aligned}\|\Phi''(\overline{x})^{-\frac{1}{2}}\overline{\mathbf{T}}^{-1}\mathbf{A}\mathbf{H}^{-1}g\|_\infty &= \max_{i\in[m]} |e_i^\top \Phi''(\overline{x})^{-\frac{1}{2}}\overline{\mathbf{T}}^{-1}\mathbf{A}\mathbf{H}^{-1}g| \\ &\le \max_{i\in[m]} |e_i^\top \overline{\mathbf{T}}^{-\frac{1}{2}}\mathbf{Q}\overline{\mathbf{T}}^{-1}e_i|^{\frac{1}{2}} |g^\top \mathbf{H}^{-1}g|^{\frac{1}{2}} \\ &= \|g\|_{\mathbf{H}^{-1}} \max_{i\in[m]} |e_i^\top \overline{\mathbf{T}}^{-\frac{1}{2}}\mathbf{Q}\overline{\mathbf{T}}^{-1}e_i|^{\frac{1}{2}} \\ &\lesssim \|g\|_{(\mathbf{A}^\top \mathbf{T}^{-1}\Phi''(x)^{-1}\mathbf{A})^{-1}} \max_{i\in[m]} \left|e_i^\top \overline{\mathbf{T}}^{-\frac{1}{2}}\mathbf{Q}\overline{\mathbf{T}}^{-\frac{1}{2}}e_i\right|^{\frac{1}{2}} \\ &\lesssim \|g\|_{(\mathbf{A}^\top \mathbf{T}^{-1}\Phi''(x)^{-1}\mathbf{A})^{-1}},\end{aligned}$$

where the second step follows from $|a^\top b| \le \|a\|_2 \cdot \|b\|_2$, the fourth step follows from $\mathbf{H} \approx_{3\varepsilon} \mathbf{A}^\top \mathbf{T}^{-1}\Phi''(x)^{-1}\mathbf{A}$, and the last step follows from $\max_{i\in[m]} |e_i^\top \overline{\mathbf{T}}^{-\frac{1}{2}}\mathbf{Q}\overline{\mathbf{T}}^{-\frac{1}{2}}e_i|^{\frac{1}{2}} \lesssim 1$. □

We now show that the change in x, i.e. $\overline{\delta}_x$, is small.

Lemma 4.29 (Bounds on $\overline{\delta}_x$)**.** *Let $x, \tau, \overline{x}, \overline{\tau}$ satisfy Invariant 4.10. If $\overline{\delta}_x$ are defined as in Algorithm 1, then*

- $\|\Phi''(x)^{\frac{1}{2}}\mathbb{E}[\overline{\delta}_x]\|_{\tau+\infty} \le \gamma + O((\varepsilon + 1/C_{\text{norm}})\gamma)$.
- $\|\Phi''(x)^{\frac{1}{2}}\overline{\delta}_x\|_\infty \lesssim \gamma$ *with probability* 1.
- $\|\mathbb{E}[\Phi''(x)\overline{\delta}_x^2]\|_{\tau+\infty} \lesssim \gamma^2$.

Proof. We break the proof into the three claims.

Bound on $\|\Phi''(x)^{\frac{1}{2}}\mathbb{E}[\overline{\delta}_x]\|_{\tau+\infty}$. Because $\mathbb{E}[\mathbf{R}] = \mathbf{I}$ (Expectation) we have that

$$\mathbb{E}[\overline{\delta}_x] = \Phi''(\overline{x})^{-\frac{1}{2}}g - \Phi''(\overline{x})^{-\frac{1}{2}}\overline{\mathbf{T}}^{-1}\Phi''(\overline{x})^{-\frac{1}{2}}\mathbf{A}\mathbf{H}^{-1}\mathbf{A}^\top\Phi''(\overline{x})^{-\frac{1}{2}}g \tag{19}$$

$$- \Phi''(\overline{x})^{-\frac{1}{2}}\overline{\mathbf{T}}^{-1}\Phi''(\overline{x})^{-\frac{1}{2}}\mathbf{A}\mathbf{H}^{-1}(\mathbf{A}^\top x - b). \tag{20}$$

We start by bounding the $\tau + \infty$ norm of the expression in (20). We calculate

$$\begin{aligned}&\|\Phi''(x)^{\frac{1}{2}}\Phi''(\overline{x})^{-\frac{1}{2}}\overline{\mathbf{T}}^{-1}\Phi''(\overline{x})^{-\frac{1}{2}}\mathbf{A}\mathbf{H}^{-1}(\mathbf{A}^\top x - b)\|_{\tau+\infty} \\ &\lesssim C_{\text{norm}}\|\mathbf{A}^\top x - b\|_{(\mathbf{A}^\top \mathbf{T}^{-1}\Phi''(x)^{-1}\mathbf{A})^{-1}} \\ &\le \varepsilon\gamma\end{aligned}$$

where the first step follows from combining Lemma 4.28 items 2 and 3, and the last step follows from the fact that (x, s, μ) is ε-centered (Definition 4.7).

Now we bound the τ-norm of the two terms in (19). Define

$$\mathbf{Q} = \overline{\mathbf{T}}^{-\frac{1}{2}}\Phi''(\overline{x})^{-\frac{1}{2}}\mathbf{A}\mathbf{H}^{-1}\mathbf{A}^\top\overline{\mathbf{T}}^{-\frac{1}{2}}\Phi''(\overline{x})^{-\frac{1}{2}},$$

and note that $\mathbf{Q} \approx_\gamma \widetilde{\mathbf{Q}}$ for orthogonal projection matrix

$$\widetilde{\mathbf{Q}} \stackrel{\text{def}}{=} \overline{\mathbf{T}}^{-\frac{1}{2}}\Phi''(\overline{x})^{-\frac{1}{2}}\mathbf{A}(\mathbf{A}^\top\overline{\mathbf{T}}^{-1}\Phi''(\overline{x})^{-1}\mathbf{A})^{-1}\mathbf{A}^\top\overline{\mathbf{T}}^{-\frac{1}{2}}\Phi''(\overline{x})^{-\frac{1}{2}}$$

by the condition in line 8 of Algorithm 1. Hence all eigenvalues of $\mathbf{Q}$ are either 0 or in $[1-\gamma, 1+\gamma]$, so all eigenvalues of $\mathbf{I}-\mathbf{Q}$ are either 1 or in $[-\gamma,\gamma]$, so $\|\mathbf{I}-\mathbf{Q}\|_2 \le 1$.

Using Lemma 4.16

$$\begin{aligned}
&\|\Phi''(x)^{\frac{1}{2}}(\Phi''(\overline{x})^{-\frac{1}{2}}g - \Phi''(\overline{x})^{-\frac{1}{2}}\overline{\mathbf{T}}^{-1}\Phi''(\overline{x})^{-\frac{1}{2}}\mathbf{A}\mathbf{H}^{-1}\mathbf{A}^\top\Phi''(\overline{x})^{-\frac{1}{2}}g)\|_\tau \\
&\le e^{O(\varepsilon)}\|g - \overline{\mathbf{T}}^{-1}\Phi''(\overline{x})^{-\frac{1}{2}}\mathbf{A}\mathbf{H}^{-1}\mathbf{A}^\top\Phi''(\overline{x})^{-\frac{1}{2}}g\|_{\overline{\tau}} \\
&= e^{O(\varepsilon)}\|(\mathbf{I}-\mathbf{Q})\overline{\mathbf{T}}^{\frac{1}{2}}g\|_2 \\
&\le e^{O(\varepsilon)}\|\overline{\mathbf{T}}^{\frac{1}{2}}g\|_2 = e^{O(\varepsilon)}\|g\|_{\overline{\tau}} \le e^{O(\varepsilon)}\|g\|_\tau \le (1+O(\varepsilon))\|g\|_\tau
\end{aligned} \tag{21}$$

where the first step follows from $\Phi''(x)^{\frac{1}{2}} \approx_{O(\varepsilon)} \Phi''(\overline{x})^{\frac{1}{2}}$ by Invariant 4.10 and Lemma 4.16, the second step follows from the definition of $\mathbf{Q}$, the third step follows from $\|\mathbf{I}-\mathbf{Q}\|_2 \le 1$, and the last step follows from $e^{O(\epsilon)} \le 1+O(\epsilon)$, $\forall \epsilon \in (0,1)$.

Now we bound the ∞ norm. To handle the first term of (19), we can use Lemma 4.16 to get

$$\|\Phi''(x)^{\frac{1}{2}}\Phi''(\overline{x})^{-\frac{1}{2}}g\|_\infty \le e^{O(\varepsilon)}\|g\|_\infty \le (1+O(\varepsilon))\|g\|_\infty,$$

where the last step follows from $e^{O(\epsilon)} \le 1+O(\epsilon)$, $\forall \epsilon \in (0,1)$.

For the second term, we have

$$\begin{aligned}
\|\Phi''(x)^{\frac{1}{2}}\Phi''(\overline{x})^{-1}\overline{\mathbf{T}}^{-1}\mathbf{A}\mathbf{H}^{-1}\mathbf{A}^\top\Phi''(\overline{x})^{-\frac{1}{2}}g\|_\infty &\lesssim \|\Phi''(\overline{x})^{-\frac{1}{2}}\overline{\mathbf{T}}^{-1}\mathbf{A}\mathbf{H}^{-1}\mathbf{A}^\top\Phi''(\overline{x})^{-\frac{1}{2}}g\|_\infty \\
&\lesssim \|g\|_\tau.
\end{aligned}$$

where the first step follows from Lemma 4.16, and the last step follows from Part 1 of Lemma 4.28.

Finally

$$\begin{aligned}
\|\Phi''(x)^{\frac{1}{2}}\mathbb{E}[\overline{\delta}_x]\|_{\tau+\infty} &\le (1+O(\varepsilon))\|g\|_\infty + O(\|g\|_\tau) + C_{\text{norm}}(1+O(\varepsilon))\|g\|_\tau + O(\varepsilon\gamma) \\
&\le (1+O(\varepsilon+1/C_{\text{norm}}))\|g\|_{\tau+\infty} + O(\varepsilon\gamma) \\
&\le \gamma + O(\varepsilon+1/C_{\text{norm}})\gamma
\end{aligned}$$

where we have used that $\|g\|_{\tau+\infty} \le \gamma$.

Bound on $\|\Phi''(x)^{\frac{1}{2}}\overline{\delta}_x\|_\infty$. First, by Lemma 4.16 we have

$$\|\Phi''(x)^{\frac{1}{2}}\overline{\delta}_x\|_\infty = \|\Phi''(x)^{\frac{1}{2}}\Phi''(\overline{x})^{-\frac{1}{2}}(g-\mathbf{R}\delta_r)\|_\infty \lesssim \|g-\mathbf{R}\delta_r\|_\infty \le \|g\|_\infty + \|\mathbf{R}\delta_r\|_\infty.$$

Note that $\|g\|_\infty \le \|g\|_{\tau+\infty} \le \gamma$, and $\|\mathbf{R}\delta_r\|_\infty \lesssim \gamma$ by the (Maximum) condition. Therefore, $\|g-\mathbf{R}\delta_r\|_\infty \lesssim \gamma$ as desired.

Bound on $\|\mathbb{E}[\Phi''(x)\overline{\delta}_x^2]\|_{\tau+\infty}$. First we use Lemma 4.16 to get

$$\begin{aligned}
\|\mathbb{E}[\Phi''(x)\overline{\delta}_x^2]\|_{\tau+\infty} &\lesssim \|\Phi''(\overline{x})\mathbb{E}[\overline{\delta}_x^2]\|_{\tau+\infty} \\
&\lesssim \|\mathbb{E}[(g-\mathbf{R}\delta_r)]^2\|_{\tau+\infty} \\
&\lesssim \|g^2\|_{\tau+\infty} + \|\mathbb{E}[\mathbf{R}^2\delta_r^2]\|_{\tau+\infty} \\
&\lesssim \gamma^2 + \|\mathbb{E}[\mathbf{R}^2\delta_r^2]\|_{\tau+\infty}.
\end{aligned}$$

We bound $\mathbb{E}[(\mathbf{R}_{ii}(\delta_r)_i)^2]$ by the (Variance) condition of Definition 4.13. This gives

$$\|\mathbb{E}[\mathbf{R}^2\delta_r^2]\|_{\tau+\infty} \le \|\delta_r^2\|_{\tau+\infty} + \frac{\gamma}{C_{\text{valid}}^2}\|\delta_r\|_{\tau+\infty} \lesssim \gamma^2$$

as $\|\delta_r\|_\tau \lesssim \gamma$ by the above, and $C_{\text{valid}} \ge C_{\text{norm}} \ge 1$. Summing these gives the result. ∎

We show a corollary which is straightforward application of Part 1 of Lemma 4.29.

Corollary 4.30. *Let $x, \tau, \overline{x}, \overline{\tau}$ satisfy Invariant 4.10. If $\overline{\delta}_x$ are defined as in Algorithm 1, then*

- $\mu^{-1}\|\mathbf{T}^{-1}\Phi''(x)^{-\frac{1}{2}}\overline{\delta}_s\|_{\tau+\infty} \lesssim \gamma$.
- $\|\delta_r\|_{\tau+\infty} \leq \gamma + O((\varepsilon + 1/C_{\text{norm}})\gamma)$.

Proof. Recall the proof of Part 1 of Lemma 4.29. The bounds on $\mu^{-1}\|\mathbf{T}^{-1}\Phi''(x)^{-\frac{1}{2}}\overline{\delta}_s\|_{\tau+\infty}$ and $\|\delta_r\|_{\tau+\infty}$ follow analogously, by replacing $\mathbf{I} - \mathbf{Q}$ with $\mathbf{Q}$ in (21). □

We can use highly 1-self-concordance to bound δ_c in terms of $\overline{\delta}_x$.

Lemma 4.31 (Bounds on δ_c)**.** *Let $x, \tau, \overline{x}, \overline{\tau}$ satisfy Invariant 4.10, and $\|g\|_{\tau+\infty} \leq \gamma$. Let $c^{(\text{new})} = \Phi''(x^{(\text{new})})^{-\frac{1}{2}}$ and $\delta_c = c^{(\text{new})} - c$. Then*

- $\|\mathbf{C}^{-1}\delta_c\|_\infty \lesssim \gamma$ *with probability* 1.
- $\|\mathbf{C}^{-1}\mathbb{E}[\delta_c]\|_{\tau+\infty} \leq \gamma + O((\varepsilon + 1/C_{\text{norm}})\gamma)$.
- $\|\mathbb{E}[\mathbf{C}^{-2}\delta_c^2]\|_{\tau+\infty} \lesssim \gamma^2$.

Proof. For the first point, simply note that $\mathbf{C}^{-1} = \Phi''(x)^{\frac{1}{2}}$ and $|\delta_c| = |c^{(\text{new})} - c| \leq |\overline{\delta}_x|$ coordinate-wise by 1-self-concordance. Therefore, the result follows from Lemma 4.29. Now, we get that $c^{(\text{new})} \approx_{O(\gamma)} c$ by Lemma 4.16.

For the second point, we integrate and use highly 1-self-concordance. Specifically, define $x_t = x + t\overline{\delta}_x$ and $c_t = \Phi''(x_t)^{-\frac{1}{2}}$. Then

$$\frac{\mathrm{d}}{\mathrm{d}t}c_t = -\frac{1}{2}\frac{\Phi'''(x_t)}{\Phi''(x_t)^{\frac{3}{2}}}\overline{\delta}_x \quad \text{and} \quad \frac{\mathrm{d}^2}{\mathrm{d}t^2}c_t = \left(-\frac{1}{2}\frac{\Phi''''(x_t)}{\Phi''(x_t)^{\frac{3}{2}}} + \frac{3}{4}\frac{\Phi'''(x_t)^2}{\Phi''(x_t)^{\frac{5}{2}}}\right)\overline{\delta}_x^2.$$

Now, we have by second order expansion, highly 1-self-concordance, and Lemma 4.29

$$\begin{aligned}
\|\mathbf{C}^{-1}\mathbb{E}[\delta_c]\|_{\tau+\infty} &\leq \left\|\Phi''(x)^{\frac{1}{2}} \cdot -\frac{1}{2}\frac{\Phi'''(x)}{\Phi''(x)^{\frac{3}{2}}}\mathbb{E}[\overline{\delta}_x]\right\|_{\tau+\infty} \\
&\quad + \frac{1}{2}\int_0^1 \left\|\mathbb{E}\left[\Phi''(x)^{\frac{1}{2}}\left(-\frac{1}{2}\frac{\Phi''''(x_t)}{\Phi''(x_t)^{\frac{3}{2}}} + \frac{3}{4}\frac{\Phi'''(x_t)^2}{\Phi''(x_t)^{\frac{5}{2}}}\right)\overline{\delta}_x^2\right]\right\|_{\tau+\infty} \mathrm{d}t \\
&\leq \left\|\Phi''(x)^{\frac{1}{2}}\mathbb{E}[\overline{\delta}_x]\right\|_{\tau+\infty} + O\left(\int_0^1 \left\|\mathbb{E}\left[\Phi''(x)\overline{\delta}_x^2\right]\right\|_{\tau+\infty} \mathrm{d}t\right) \\
&\leq \gamma + O(\varepsilon + 1/C_{\text{norm}})\gamma + O(\gamma^2) \\
&\leq \gamma + O(\varepsilon + 1/C_{\text{norm}})\gamma,
\end{aligned}$$

where the last step follows from $\gamma \in (0, 1)$.

For the final point, use again that $|\delta_c| \leq |\overline{\delta}_x|$ pointwise, so by Lemma 4.29

$$\|\mathbb{E}[\mathbf{C}^{-2}\delta_c^2]\|_{\tau+\infty} \leq \|\mathbb{E}[\Phi''(x)\overline{\delta}_x^2]\|_{\tau+\infty} \lesssim \gamma^2.$$

□

The next few lemmas show that δ_c is γ-bounded, as in Definition 4.23. We need a variant of [BLSS20, Lemma 48].

Lemma 4.32 ($\mathbf{P}^{(2)}$ norm bound)**.** *If $\mathbf{R}$ is valid (Definition 4.13) then for any vector $v \in \mathbb{R}^m$ we have that*

$$\mathbb{E}[\||\mathbf{R}v|\|_{\mathbf{P}^{(2)}}^2] \lesssim \|\mathbb{E}[|v|]\|_\tau^2 \text{ and } \mathbb{E}[\||\mathbf{R}v|\|_{\mathbf{P}^{(2)}}] \lesssim \|\mathbb{E}[|v|]\|_\tau.$$

Proof. The second claim follows from the first and Cauchy-Schwarz. For the first claim, we compute using (Variance) and (Covariance) of Definition 4.13 that

$$\begin{aligned}\mathbb{E}[\||\mathbf{R}v|\|_{\mathbf{P}^{(2)}}^2] &= \sum_{i,j}\mathbb{E}[\mathbf{R}_{ii}\mathbf{R}_{jj}]|v_i||v_j|\mathbf{P}_{ij}^2 \\ &\le 2\sum_{i\in[m]} v_i^2\sigma_i^{-1}\mathbf{P}_{ii}^2 + 2\sum_{i\neq j}|v_i||v_j|\mathbf{P}_{ij}^2 \\ &\le 2\sum_{i\in[m]} v_i^2\sigma_i + 2\||v|\|_{\mathbf{P}^{(2)}}^2 \lesssim \|v\|_\tau^2.\end{aligned}$$

□

We need the following fact to handle terms involving $\mathbf{P}^{(2)}$.

Lemma 4.33 (Bounds on δ_x and δ_c with respect to $\mathbf{P}_t^{(2)}$ norm)**.** *We have that* $\mathbb{E}[\|\mathbf{C}^{-1}|\delta_x|\|_{\mathbf{P}_t^{(2)}}] \lesssim \gamma/C_{\text{norm}}$ *and* $\mathbb{E}[\|\mathbf{C}^{-1}|\delta_c|\|_{\mathbf{P}_t^{(2)}}] \lesssim \gamma/C_{\text{norm}}$.

Proof. Note that

$$\mathbf{C}^{-1}|\delta_x| = |g - \mathbf{R}\delta_r| \le |g| + |\mathbf{R}\delta_r|.$$

Note that $\mathbf{T}_t^{1/2-1/p}\mathbf{C}_t \approx_2 \mathbf{T}^{1/2-1/p}\mathbf{C}$. Therefore, we can use $\mathbf{P}_t^{(2)} \preceq \mathbf{\Sigma}_t$, $\mathbf{\Sigma}_t \approx_2 \mathbf{\Sigma}$, Lemma 4.29, [BLN+20, Lemma 4.23], and Lemma 4.32 to get

$$\begin{aligned}\mathbb{E}[\|\mathbf{C}^{-1}|\delta_x|\|_{\mathbf{P}_t^{(2)}}] &\le \|g\|_{\mathbf{P}_t^{(2)}} + \mathbb{E}[\||\mathbf{R}\delta_r|\|_{\mathbf{P}_t^{(2)}}] \\ &\lesssim \gamma/C_{\text{norm}} + \mathbb{E}[\||\mathbf{R}\delta_r|\|_{\mathbf{P}^{(2)}}] \\ &\le \gamma/C_{\text{norm}} + \|\delta_r\|_\tau \lesssim \gamma/C_{\text{norm}},\end{aligned}$$

where the second step follows from $\|g\|_{\mathbf{P}_t^{(2)}} \lesssim \gamma/C_{\text{norm}}$, the third step follows from $\mathbb{E}[\||\mathbf{R}\delta_r|\|_{\mathbf{P}^{(2)}}] \le \|\delta_r\|_\tau$.

The second claim follows directly from 1-self-concordance and the first claim. □

Lemma 4.34 (γ-boundedness of δ_c)**.** *For* $c^{(\text{new})} = \Phi''(x^{(\text{new})})^{-1/2}$ *and* $\delta_c = c^{(\text{new})} - c$*, we have that* δ_c *is* γ*-bounded.*

Proof. To show Definition 4.23 (16) and (17), use Lemma 4.31. Definition 4.23 (18) follows from Lemma 4.33. □

By Lemmas 4.25 and 4.26, γ-boundedness of δ_c allows us to control the change to τ.

Lemma 4.35 (Bounds on δ_τ)**.** *Let* $\tau^{(\text{new})} = \tau(x^{(\text{new})})$*, and* $\delta_\tau = \tau^{(\text{new})} - \tau$*. Then we have that*

- $\|\mathbf{T}^{-1}\delta_\tau\|_\infty \lesssim \gamma$ *with probability* 1.
- $\left\|\mathbb{E}[(\mathbf{T}^{-1}\delta_\tau)^2]\right\|_{\tau+\infty} \lesssim \gamma^2$.
- $\left\|\mathbf{T}^{-1}(\mathbb{E}[\delta_\tau] - \mathbf{J}\mathbb{E}[\delta_c])\right\|_{\tau+\infty} \lesssim \gamma^2$.

Proof. Follows directly from the fact that δ_c is γ-bounded (Lemma 4.34), along with Lemmas 4.25 and 4.26. □

Now, we show that despite the approximations to x, s, τ, the algorithm still take a step in a direction very close to the desired direction g.

Lemma 4.36 (Approximation to direction g)**.** *Let $x, \tau, \overline{x}, \overline{\tau}$ be as in Algorithm 1. Then*

$$\|\mu^{-1}\Phi''(x)^{-\frac{1}{2}}\mathbf{T}^{-1}(\overline{\delta}_s + \mu\tau\phi''(x)\mathbb{E}[\overline{\delta}_x]) - g\|_{\tau+\infty} \lesssim \varepsilon\gamma.$$

Proof. At a high level, our proof will gradually change x to $\overline{x}$ and τ to $\overline{\tau}$ and track the incurred error. Here, $\tau = \tau(x)$ and $\overline{\tau}$ satisfies Invariant 4.10. We let $\delta_x = \mathbb{E}[\overline{\delta}_x]$. First, we write

$$\begin{aligned}\overline{\delta}_s + \mu\tau\phi''(x)\mathbb{E}[\overline{\delta}_x] &= \overline{\delta}_s + \mu\overline{\tau}\phi''(x)\delta_x + \mu(\tau-\overline{\tau})\phi''(x)\delta_x \\ &= \overline{\delta}_s + \mu\overline{\tau}\phi''(\overline{x})\delta_x + \mu(\tau-\overline{\tau})\phi''(x)\delta_x + \mu\overline{\tau}(\phi''(x)-\phi''(\overline{x}))\delta_x \\ &= \mu\overline{\mathbf{T}}\Phi''(\overline{x})^{\frac{1}{2}}(g-\delta_2) + \mu(\tau-\overline{\tau})\phi''(x)\delta_x + \mu\overline{\tau}(\phi''(x)-\phi''(\overline{x}))\delta_x.\end{aligned}$$

Therefore, we have that

$$\begin{aligned}&\mu^{-1}\Phi''(x)^{-\frac{1}{2}}\mathbf{T}^{-1}(\overline{\delta}_s + \mu\tau\phi''(x)\delta_x) - g \\ &= (\Phi''(x)^{-\frac{1}{2}}\mathbf{T}^{-1}\Phi''(\overline{x})^{\frac{1}{2}}\overline{\mathbf{T}} - \mathbf{I})g + \Phi''(x)^{-\frac{1}{2}}\mathbf{T}^{-1}(\tau-\overline{\tau})\phi''(x)\delta_x \\ &+ \Phi''(x)^{-\frac{1}{2}}\mathbf{T}^{-1}\overline{\tau}(\phi''(x)-\phi''(\overline{x}))\delta_x - \Phi''(x)^{-\frac{1}{2}}\mathbf{T}^{-1}\Phi''(\overline{x})^{\frac{1}{2}}\overline{\mathbf{T}}\delta_2.\end{aligned}$$

We now bound the terms in the above sum. For the second term, we use the approximation condition of Invariant 4.10 and Lemma 4.22 to get that

$$\|\Phi''(x)^{-\frac{1}{2}}\mathbf{T}^{-1}(\tau-\overline{\tau})\phi''(x)\delta_x\|_{\tau+\infty} \lesssim \varepsilon\|\Phi''(x)^{\frac{1}{2}}\overline{\delta}_x\|_{\tau+\infty} \lesssim \varepsilon\|g\|_{\tau+\infty} \lesssim \varepsilon\gamma$$

by Lemma 4.29. For the third term, we use the approximation condition of Invariant 4.10, Lemma 4.16 to get that

$$\|\Phi''(x)^{-1}(\phi''(x)-\phi''(\overline{x}))\|_\infty \lesssim \varepsilon.$$

Applying this and $\|\mathbf{T}^{-1}\overline{\tau}\|_\infty \lesssim 1$ using Lemma 4.22, we get that

$$\|\Phi''(x)^{-\frac{1}{2}}\mathbf{T}^{-1}\overline{\tau}(\phi''(x)-\phi''(\overline{x}))\delta_x\|_{\tau+\infty} \lesssim \varepsilon\|\Phi''(x)^{\frac{1}{2}}\delta_x\|_{\tau+\infty} \lesssim \varepsilon\|g\|_{\tau+\infty} \lesssim \varepsilon\gamma$$

by Lemma 4.29. For the fourth/last term, we use Lemma 4.28 to get

$$\|\Phi''(x)^{-\frac{1}{2}}\mathbf{T}^{-1}\Phi''(\overline{x})^{\frac{1}{2}}\overline{\mathbf{T}}\delta_2\|_{\tau+\infty} \lesssim \|\delta_2\|_{\tau+\infty} \lesssim C_{\text{norm}}\|\mathbf{A}^\top x - b\|_{(\mathbf{A}^\top\mathbf{T}^{-1}\Phi''(x)^{-1}\mathbf{A})^{-1}} \lesssim \varepsilon\gamma,$$

where we have used that our point is ε-centered.

For the first term, we write

$$\begin{aligned}&(\mathbf{I} - \Phi''(x)^{-\frac{1}{2}}\mathbf{T}^{-1}\Phi''(\overline{x})^{\frac{1}{2}}\overline{\mathbf{T}})g \\ &= (\mathbf{I} - \Phi''(x)^{-\frac{1}{2}}\Phi''(\overline{x})^{\frac{1}{2}} + \Phi''(x)^{-\frac{1}{2}}\mathbf{T}^{-1}\Phi''(\overline{x})^{\frac{1}{2}}(\mathbf{T}-\overline{\mathbf{T}}))g \\ &= (\Phi''(x)^{-\frac{1}{2}}(\Phi''(\overline{x})^{\frac{1}{2}} - \Phi''(x)^{\frac{1}{2}}) + \Phi''(x)^{-\frac{1}{2}}\mathbf{T}^{-1}\Phi''(\overline{x})^{\frac{1}{2}}(\mathbf{T}-\overline{\mathbf{T}}))g.\end{aligned}$$

For the first term in the previous expression, we can use the approximation condition of Invariant 4.10, and Lemma 4.16 to get

$$\|\Phi''(x)^{-\frac{1}{2}}(\Phi''(\overline{x})^{\frac{1}{2}} - \Phi''(x)^{\frac{1}{2}})g\|_{\tau+\infty} \lesssim \varepsilon\|g\|_{\tau+\infty} \le \varepsilon\gamma.$$

For the second term, we can use the approximation condition of Invariant 4.10, Lemma 4.16, and Lemma 4.22 to get

$$\|\Phi''(x)^{-\frac{1}{2}}\mathbf{T}^{-1}\Phi''(\overline{x})^{\frac{1}{2}}(\overline{\mathbf{T}}-\mathbf{T}))g\|_{\tau+\infty} \lesssim \varepsilon\|g\|_{\tau+\infty} \le \varepsilon\gamma.$$

Summing over these bounds gives the desired result. ∎

Combining the above analyses allows us to analyze δ_y.

Lemma 4.37 (Bounds on δ_y)**.** *Let $x, \tau, \overline{x}, \overline{\tau}$ be as in Algorithm 1, and let*

$$y^{(\text{new})} = s^{(\text{new})} + \mu^{(\text{new})}\tau(x^{(\text{new})})\phi'(x^{(\text{new})}).$$

Then we have that

$$\|\mu^{-1}\Phi''(x)^{-\frac{1}{2}}\mathbf{T}^{-1}\mathbb{E}[\delta_y] - g\|_{\tau+\infty} \leq p\gamma + O((\varepsilon + 1/C_{\text{norm}})\gamma)$$

and

$$\|\mathbb{E}[(\mu^{-1}\Phi''(x)^{-\frac{1}{2}}\mathbf{T}^{-1}\delta_y)^2]\|_{\tau+\infty} \lesssim \gamma^2.$$

Proof. We start with the first claim. Let $\delta_{\phi'} = \Phi'(x^{(\text{new})}) - \Phi'(x)$.

$$\begin{aligned}
y^{(\text{new})} &= s^{(\text{new})} + \mu^{(\text{new})}\tau^{(\text{new})}\phi'(x^{(\text{new})}) \\
&= s + \overline{\delta}_s + \mu^{(\text{new})}\tau\phi'(x^{(\text{new})}) + \mu^{(\text{new})}\delta_\tau\phi'(x^{(\text{new})}) \\
&= s + \overline{\delta}_s + \mu\tau\phi'(x^{(\text{new})}) + \mu^{(\text{new})}\delta_\tau\phi'(x^{(\text{new})}) + \delta_\mu\tau\phi'(x^{(\text{new})}) \\
&= s + \overline{\delta}_s + \mu\tau\phi'(x) + \mu^{(\text{new})}\delta_\tau\phi'(x^{(\text{new})}) + \delta_\mu\tau\phi'(x^{(\text{new})}) + \mu\tau\delta_{\phi'} \\
&= y + (\overline{\delta}_s + \mu\tau\delta_{\phi'}) + \mu^{(\text{new})}\delta_\tau\phi'(x^{(\text{new})}) + \delta_\mu\tau\phi'(x^{(\text{new})}),
\end{aligned}$$

where for simplicity of notation we have defined vector multiplication coordinate-wise. Also for $x_t = x + t\overline{\delta}_x$ we have that

$$\delta_{\phi'} = \phi''(x)\overline{\delta}_x + \int_0^1 (1-t)\phi'''(x_t)\overline{\delta}_x^2 \mathrm{d}t.$$

Therefore, we have that

$$\delta_y = (\overline{\delta}_s + \mu\tau\phi''(x)\overline{\delta}_x) + \mu^{(\text{new})}\delta_\tau\phi'(x^{(\text{new})}) + \delta_\mu\tau\phi'(x^{(\text{new})}) + \mu\tau\int_0^1 (1-t)\phi'''(x_t)\overline{\delta}_x^2 \mathrm{d}t.$$

We analyze the four terms in the above term one by one. The first term can be handled by using Lemma 4.36. Precisely, we have that

$$\|\mu^{-1}\Phi''(x)^{-\frac{1}{2}}\mathbf{T}^{-1}(\overline{\delta}_s + \mu\tau\phi''(x)\mathbb{E}[\overline{\delta}_x]) - g\|_{\tau+\infty} \lesssim \varepsilon\gamma.$$

For the second term, we first rewrite

$$\begin{aligned}
&\|\mu^{-1}\Phi''(x)^{-\frac{1}{2}}\mathbb{E}[\phi'(x^{(\text{new})})\mathbf{T}^{-1}\mu^{(\text{new})}\delta_\tau]\|_{\tau+\infty} \\
\leq\ &(1+r)\|\Phi''(x)^{-\frac{1}{2}}\mathbb{E}[\phi'(x^{(\text{new})})\mathbf{T}^{-1}\delta_\tau]\|_{\tau+\infty} \\
\leq\ &(1+r)(\|\Phi''(x)^{-\frac{1}{2}}\Phi'(x)\mathbb{E}[\mathbf{T}^{-1}\delta_\tau]\|_{\tau+\infty} + \|\mathbb{E}[\Phi''(x)^{-\frac{1}{2}}\delta_{\phi'}\mathbf{T}^{-1}\delta_\tau]\|_{\tau+\infty}).
\end{aligned}$$

For the first of these, we can write

$$\begin{aligned}
(1+r)\|\Phi''(x)^{-\frac{1}{2}}\Phi'(x)\mathbb{E}[\mathbf{T}^{-1}\delta_\tau]\|_{\tau+\infty} &\leq (1+r)\|\mathbb{E}[\mathbf{T}^{-1}\delta_\tau]\|_{\tau+\infty} \\
&\leq (1+r)(\|\mathbb{E}[\mathbf{T}^{-1}\mathbf{J}\delta_c]\|_{\tau+\infty} + \|\mathbb{E}[\mathbf{T}^{-1}\delta_\tau] - \mathbb{E}[\mathbf{T}^{-1}\mathbf{J}\delta_c]\|_{\tau+\infty}) \\
&\leq (1+r)(\|\mathbb{E}[\mathbf{T}^{-1}\mathbf{J}\delta_c]\|_{\tau+\infty} + O(\gamma^2)) \\
&= (1+r)(\|\mathbf{T}^{-1}\mathbf{J}\mathbf{C}\mathbf{C}^{-1}\mathbb{E}[\delta_c]\|_{\tau+\infty} + O(\gamma^2)) \\
&\leq (1+O(r+1/C_{\text{norm}}))p\|\mathbf{C}^{-1}\mathbb{E}[\delta_c]\|_{\tau+\infty} + O(\gamma^2) \\
&\leq (1+O(r+\varepsilon+1/C_{\text{norm}}))p\gamma + O(\gamma^2)
\end{aligned}$$

$$\leq p\gamma + O(\varepsilon + 1/C_{\text{norm}})\gamma,$$

where the first step follows from 1-self-concordance (Definition 4.1), the second step follows from triangle inequality, the third step follows from Part 3 of Lemma 4.35, the fifth step follows from Lemma A.2 Part 3, the sixth step follows from Part 2 of Lemma 4.31, and the last step follows from the choice of parameters.

For the second, we can write

$$\begin{aligned}
\|\mathbb{E}[\Phi''(x)^{-\frac{1}{2}}\delta_{\phi'}\mathbf{T}^{-1}\delta_\tau]\|_{\tau+\infty} &\lesssim \|\mathbb{E}[(\mathbf{T}^{-1}\delta_\tau)^2]\|_{\tau+\infty} + \|\mathbb{E}[\Phi''(x)^{-1}\delta_{\phi'}^2]\|_{\tau+\infty} \qquad (22)\\
&\lesssim \gamma^2 + \|\mathbb{E}[\Phi''(x)^{-1}\delta_{\phi'}^2]\|_{\tau+\infty}\\
&= \gamma^2 + \left\|\mathbb{E}\left[\Phi''(x)^{-1}\left(\int_0^1 \Phi''(x_t)\overline{\delta}_x \mathrm{d}t\right)^2\right]\right\|_{\tau+\infty}\\
&\lesssim \gamma^2 + \int_0^1 \left\|\mathbb{E}[\Phi''(x)^{-1}\Phi''(x_t)^2\overline{\delta}_x^2]\right\|_{\tau+\infty}\mathrm{d}t\\
&\lesssim \gamma^2 + \|\mathbb{E}[\Phi''(x)\overline{\delta}_x^2]\|_{\tau+\infty} \lesssim \gamma^2, \qquad (23)
\end{aligned}$$

where the first step follows from AM-GM and the triangle inequality, the second step follows from Lemma 4.35 Part 2, the fourth step follows by Cauchy-Schwarz, the fifth step follows from $\Phi''(x) \approx_1 \Phi''(x_t)$, and the final step follows from Lemma 4.29 Part 3.

Therefore, the total for the second term is at most $p\gamma + O(\varepsilon + 1/C_{\text{norm}})\gamma$, as $\gamma \leq \varepsilon$.

For the third term, we can bound

$$\begin{aligned}
\|\mu^{-1}\Phi''(x)^{-\frac{1}{2}}\mathbf{T}^{-1}\delta_\mu\tau\phi'(x^{(\text{new})})\|_{\tau+\infty} &= \mu^{-1}\delta_\mu\|\Phi''(x^{(\text{new})})^{-\frac{1}{2}}\phi'(x^{(\text{new})})\|_{\tau+\infty}\\
&\lesssim r\|\vec{1}\|_{\tau+\infty}\\
&\lesssim rC_{\text{norm}}n^{\frac{1}{2}} \lesssim \varepsilon\gamma.
\end{aligned}$$

where the second step uses 1-self-concordance (Definition 4.1), the third step uses $|\mu^{-1}\delta_\mu| \lesssim r$ by the choice of r, and the final step uses $\|\vec{1}\|_{\tau+\infty} = 1 + C_{\text{norm}}\sqrt{\|v\|_1} \lesssim C_{\text{norm}}n^{\frac{1}{2}}$ by the choice of parameters.

For the fourth term, we have

$$\begin{aligned}
\left\|\mu^{-1}\Phi''(x)^{-\frac{1}{2}}\mathbf{T}^{-1}\mu\tau\int_0^1(1-t)\mathbb{E}[\phi'''(x_t)\overline{\delta}_x^2]\mathrm{d}t\right\|_{\tau+\infty} &\lesssim \int_0^1 \|\mathbb{E}[\Phi''(x_t)^{-\frac{1}{2}}|\Phi'''(x_t)|\overline{\delta}_x^2]\|_{\tau+\infty}\\
&\lesssim \int_0^1 \|\mathbb{E}[\Phi''(x_t)\overline{\delta}_x^2]\|_{\tau+\infty} \lesssim \gamma^2
\end{aligned}$$

where the first step follows from $\Phi''(x_t) \approx_1 \Phi''(x)$ and the triangle inequality, the second step follows from 1-self-concordance (Definition 4.1) and the final step follows from Lemma 4.29 Part 3.

Combining everything gives us that

$$\|\mu^{-1}\Phi''(x)^{-\frac{1}{2}}\mathbf{T}^{-1}\delta_y - g\|_{\tau+\infty} - p\gamma \lesssim (\varepsilon + 1/C_{\text{norm}})\gamma.$$

Now, we move on to the second claim. Once again, we write

$$\delta_y = \overline{\delta}_s + \mu\tau\delta_{\phi'} + \mu^{(\text{new})}\delta_\tau\phi'(x^{(\text{new})}) + \delta_\mu\tau\phi'(x^{(\text{new})})$$

and note that

$$\delta_y^2 \lesssim \overline{\delta}_s^2 + (\mu\tau\delta_{\phi'})^2 + (\mu^{(\text{new})}\delta_\tau\phi'(x^{(\text{new})}))^2 + (\delta_\mu\tau\phi'(x^{(\text{new})}))^2.$$

We analyze it term by term. Lemma 4.29 we have

$$\|(\mu^{-1}\Phi''(x)^{-\frac{1}{2}}\mathbf{T}^{-1}\overline{\delta}_s)^2\|_{\tau+\infty} \leq \|\mu^{-1}\Phi''(x)^{-\frac{1}{2}}\mathbf{T}^{-1}\overline{\delta}_s^2\|_{\tau+\infty}^2 \lesssim \gamma^2.$$

By (22) and (23) above we have

$$\|\mathbb{E}(\mu^{-1}\Phi''(x)^{-\frac{1}{2}}\mathbf{T}^{-1}\mu\tau\delta_{\phi'})^2\|_{\tau+\infty} \leq \|\mathbb{E}[\Phi''(x)^{-1}\delta_{\phi'}^2]\|_{\tau+\infty} \lesssim \gamma^2.$$

By Lemma 4.31, 1-self-concordance, and Lemma 4.35 we have

$$\|\mathbb{E}(\mu^{-1}\Phi''(x)^{-\frac{1}{2}}\mathbf{T}^{-1}\mu^{(\text{new})}\delta_\tau\phi'(x^{(\text{new})}))^2\|_{\tau+\infty} \lesssim \|\mathbb{E}(\mathbf{T}^{-1}\delta_\tau)^2\|_{\tau+\infty} \lesssim \gamma^2.$$

By Lemma 4.27, 1-self-concordance, and Lemma 4.31 we have

$$\|\mathbb{E}(\mu^{-1}\Phi''(x)^{-\frac{1}{2}}\mathbf{T}^{-1}\delta_\mu\tau\phi'(x^{(\text{new})}))^2\|_{\tau+\infty} \lesssim r^2\|\vec{1}\|_{\tau+\infty} \lesssim r^2 C_{\text{norm}} n^{\frac{1}{2}} = r\varepsilon\gamma \lesssim \gamma^2$$

by the choice of r. Combining these gives the desired result. □

4.5 Feasibility and potential function analysis

Section	Statement	Comment
Section 4.5	Lemma 4.38	Feasibility bound
Section 4.5	Lemma 4.39	First potential drop
Section 4.5	Corollary 4.40	Final potential drop
Section 4.6	Lemma 4.41	Independent sampling
Section 4.6	Lemma 4.42	Proportional sampling
Section 4.6	Corollary 4.43	Sampling by a mixture of ℓ_2 and uniform.
Section 4.7	Lemma 4.44	Nearby stability of x
Section 4.7	Lemma 4.45	Nearby stability of ϕ'' and τ
Section 4.7	Lemma 4.46	Parameter changes along the central path

Table 5: Summary of Section 4.5, 4.6, 4.7.

In this section, we analyze the change in feasibility and centrality potential. We start with the feasibility.

Lemma 4.38 (Feasibility bound)**.** *For sufficiently large constants C in Algorithm 1 we have the following. Let $x^{(\text{new})}, s^{(\text{new})}, \mu$ be as in Algorithm 1, where $\|g\|_{\tau+\infty} \leq \gamma$ and (x, s, μ) is ε-centered. Then with probability at least $1 - m^{-10}$ we have*

$$\|\mathbf{A}^\top x^{(\text{new})} - b\|_{(\mathbf{A}^\top(\mathbf{T}(x^{(\text{new})})\Phi''(x^{(\text{new})}))^{-1}\mathbf{A})^{-1}} \leq .5\varepsilon\gamma/C_{\text{norm}}.$$

Proof. Define $\overline{\mathbf{A}} = \overline{\mathbf{T}}^{-\frac{1}{2}}\Phi''(\overline{x})^{-\frac{1}{2}}\mathbf{A}$, and note that

$$\mathbf{A}^\top(\mathbf{T}(x^{(\text{new})})\Phi''(x^{(\text{new})}))^{-1}\mathbf{A} \approx_2 \overline{\mathbf{A}}^\top\overline{\mathbf{A}}$$

by Lemma 4.31, Lemma 4.35, and $\mathbf{H} \approx_\gamma \overline{\mathbf{A}}^\top\overline{\mathbf{A}}$ by definition. Define the vector $v = \mathbf{A}^\top\Phi''(\overline{x})^{-\frac{1}{2}}g + (\mathbf{A}^\top x - b)$. A direct calculation shows that

$$\mathbf{A}^\top x^{(\text{new})} - b = (\mathbf{I} - \mathbf{A}^\top\Phi''(\overline{x})^{-\frac{1}{2}}\overline{\mathbf{T}}^{-\frac{1}{2}}\mathbf{R}\overline{\mathbf{T}}^{-\frac{1}{2}}\Phi''(\overline{x})^{-\frac{1}{2}}\mathbf{A}\mathbf{H}^{-1})v = (\mathbf{I} - \overline{\mathbf{A}}^\top\mathbf{R}\overline{\mathbf{A}}\mathbf{H}^{-1})v.$$

Therefore, we have by Lemma 4.28 that

$$\|\mathbf{A}^\top x^{(\text{new})} - b\|_{(\mathbf{A}^\top(\mathbf{T}(x^{(\text{new})})\Phi''(x^{(\text{new})}))^{-1}\mathbf{A})^{-1}} \lesssim \|\mathbf{H}^{-\frac{1}{2}}(\mathbf{I} - \overline{\mathbf{A}}^\top\mathbf{R}\overline{\mathbf{A}}\mathbf{H}^{-1})\mathbf{H}^{\frac{1}{2}}(\mathbf{H}^{-\frac{1}{2}}v)\|_2$$

$$\leq \|\mathbf{H}^{-\frac{1}{2}}(\mathbf{H} - \overline{\mathbf{A}}^\top \mathbf{R}\overline{\mathbf{A}})\mathbf{H}^{-\frac{1}{2}}\|_2 \|\mathbf{H}^{-\frac{1}{2}} v\|_2.$$

We have that $\overline{\mathbf{A}}^\top \mathbf{R}\overline{\mathbf{A}} \approx_\gamma \overline{\mathbf{A}}^\top \overline{\mathbf{A}} \approx_\gamma \mathbf{H}$ with probability $1 - m^{-10}$ by the (Matrix approximation) condition of Definition 4.13 and line 8 of Algorithm 1. Therefore, by [BLN+20, Lemma 4.30] we have that

$$\|\mathbf{H}^{-\frac{1}{2}}(\mathbf{H} - \overline{\mathbf{A}}^\top \mathbf{R}\overline{\mathbf{A}})\mathbf{H}^{-\frac{1}{2}}\|_2 \lesssim \gamma.$$

Also

$$\begin{aligned}\|\mathbf{H}^{-\frac{1}{2}} v\|_2 &\leq \|\mathbf{H}^{-\frac{1}{2}}\mathbf{A}^\top \Phi''(\overline{x})^{-\frac{1}{2}} g\|_2 + \|\mathbf{H}^{-\frac{1}{2}}(\mathbf{A}^\top x - b)\|_2 \\ &\lesssim \|g\|_\tau + \varepsilon\gamma / C_{\text{norm}} \lesssim \gamma / C_{\text{norm}}.\end{aligned}$$

Therefore with probability $1 - m^{-10}$ we have

$$\|\mathbf{A}^\top x^{(\text{new})} - b\|_{(\mathbf{A}^\top(\mathbf{T}(x^{(\text{new})})\Phi''(x^{(\text{new})}))^{-1}\mathbf{A})^{-1}} \lesssim \gamma \cdot \gamma / C_{\text{norm}} \leq \gamma^2 / C_{\text{norm}}.$$

Let C_2 be the universal constant such that

$$\|\mathbf{A}^\top x^{(\text{new})} - b\|_{(\mathbf{A}^\top(\mathbf{T}(x^{(\text{new})})\Phi''(x^{(\text{new})}))^{-1}\mathbf{A})^{-1}} \leq C_2\gamma^2 / C_{\text{norm}}.$$

Now, we can choose $C \geq 2C_2$, so that $\gamma \leq \varepsilon / C \leq \varepsilon/(2C_2)$. Then

$$\|\mathbf{A}^\top x^{(\text{new})} - b\|_{(\mathbf{A}^\top(\mathbf{T}(x^{(\text{new})})\Phi''(x^{(\text{new})}))^{-1}\mathbf{A})^{-1}} \leq C_2\gamma^2 / C_{\text{norm}} \leq .5\varepsilon\gamma / C_{\text{norm}}$$

as desired. □

We first bound the effect of the step in Algorithm 1 on the centrality potential.

Lemma 4.39 (First potential drop)**.** *Let $x^{(\text{new})}, s^{(\text{new})}, \mu$ be as in Algorithm 1. For sufficiently large choice of C, we have that for $v = \frac{s_i + \mu\tau(x)_i \phi_i'(x_i)}{\mu\tau(x)_i \sqrt{\phi_i''(x_i)}}$ and $\alpha = 1 - p$ we have*

$$\begin{aligned}\mathbb{E}[\Psi(x^{(\text{new})}, s^{(\text{new})}, \mu^{(\text{new})})] &\leq \Psi(x, s, \mu) + \psi'(v)^\top g \\ &\quad + (1 - \alpha/4)\|\psi'(v)\|^*_{\tau+\infty}\gamma + O(\|\psi''(v)\|^*_{\tau+\infty}\gamma^2).\end{aligned}$$

Proof. We carefully apply Lemma 4.15 for the choices $u^{(j)} = \mu, \tau, p$ respectively and y as itself. Note that

$$\|\mathbf{C}^{-1}\mathbb{E}[\delta_c]\|_{\tau+\infty} \lesssim \gamma \quad \text{and} \quad \|\mathbf{T}^{-1}\mathbb{E}[\delta_\tau]\|_{\tau+\infty} \lesssim \gamma$$

by Lemma 4.31 and Lemma 4.35 respectively. Also, by Lemma 4.27

$$\|\mu^{-1}\delta_\mu\|_{\tau+\infty} \lesssim \varepsilon\gamma.$$

We now bound each term in the conclusion of Lemma 4.15. We start with (13). For the $\psi'(v)^\top \mathbf{W}\eta$ term we bound

$$\begin{aligned}\mathbb{E}[\psi'(v)^\top \mu^{-1}\Phi''(x)^{-\frac{1}{2}}\mathbf{T}^{-1}\delta_y] &= \psi'(v)^\top g + \psi'(v)^\top \left(\mu^{-1}\Phi''(x)^{-\frac{1}{2}}\mathbf{T}^{-1}\mathbb{E}[\delta_y] - g\right) \\ &\leq \psi'(v)^\top g + \|\psi'(v)\|^*_{\tau+\infty}\|\mu^{-1}\Phi''(x)^{-\frac{1}{2}}\mathbf{T}^{-1}\mathbb{E}[\delta_y] - g\|_{\tau+\infty} \\ &\leq \psi'(v)^\top g + \|\psi'(v)\|^*_{\tau+\infty}(p\gamma + O(\varepsilon + 1/C_{\text{norm}})\gamma) \qquad (24) \\ &\leq \psi'(v)^\top g + (1 - \alpha/2)\|\psi'(v)\|^*_{\tau+\infty}\gamma\end{aligned}$$

for sufficiently large C. Indeed, let C_3 be the universal constant such that the bound in (24) is

$$\psi'(v)^\top g + \|\psi'(v)\|^*_{\tau+\infty}(p\gamma + C_3(\varepsilon + 1/C_{\text{norm}})\gamma).$$

We can then choose $C \geq 10C_3$ so that $\varepsilon = \alpha/C = 1/C_{\text{norm}}$.

For the terms $\psi'(v)^\top \mathbf{V}(\mathbf{U}^{(j)})^{-1}\delta^{(j)}$ we can bound them as for example

$$\begin{aligned}\left|\psi'(v)^\top \mathbf{V}\mathbf{T}^{-1}\mathbb{E}[\delta_\tau]\right| &\lesssim \|\mathbf{V}\psi'(v)\|^*_{\tau+\infty}\|\mathbf{T}^{-1}\mathbb{E}[\delta_\tau]\|_{\tau+\infty} \\ &\lesssim \varepsilon\|\psi'(v)\|^*_{\tau+\infty}\gamma \\ &\leq \frac{\alpha}{12}\|\psi'(v)\|^*_{\tau+\infty}\gamma\end{aligned}$$

because $\|v\|_\infty \leq \varepsilon$, and that $\varepsilon = \alpha/C$ for a sufficiently large constant C. Similar bounds hold for the contributions from δ_c and δ_μ.

Now we bound (14). First, the contribution of the $16\|\mathbf{W}\eta\|^2_{\psi''(v)}$ term is at most

$$\begin{aligned}16\mathbb{E}\left[\|\mu^{-1}\Phi''(x)^{-\frac{1}{2}}\mathbf{T}^{-1}\delta_y\|^2_{\psi''(v)}\right] &\lesssim \|\psi''(v)\|^*_{\tau+\infty}\|\mathbb{E}[(\mu^{-1}\Phi''(x)^{-\frac{1}{2}}\mathbf{T}^{-1}\delta_y)^2]\|_{\tau+\infty} \\ &\lesssim \|\psi''(v)\|^*_{\tau+\infty}\gamma^2\end{aligned}$$

by Lemma 4.37. In (14) we know that $1 + \|c\|_1 = 4$. For the second term in (14) we have

$$\mathbb{E}[\|\mathbf{T}^{-1}\delta_\tau\|^2_{\psi''(v)}] \leq \|\psi''(v)\|^*_{\tau+\infty}\|\mathbb{E}[(\mathbf{T}^{-1}\delta_\tau)^2]\|_{\tau+\infty} \lesssim \|\psi''(v)\|^*_{\tau+\infty}\gamma^2$$

by Lemma 4.35, and achieve bounds of

$$\mathbb{E}[\|\mathbf{C}^{-1}\delta_c\|^2_{\psi''(v)}] \leq \|\psi''(v)\|^*_{\tau+\infty}\|\mathbb{E}[(\mathbf{C}^{-1}\delta_c)^2]\|_{\tau+\infty} \lesssim \|\psi''(v)\|^*_{\tau+\infty}\gamma^2$$

and

$$\mathbb{E}[\|\mathbf{M}^{-1}\delta_\mu\|^2_{\psi''(v)}] \lesssim \|\psi''(v)\|^*_{\tau+\infty}\gamma^2$$

similarly, using Lemma 4.31 and Lemma 4.27. Therefore, the total contribution from (14) and (15) is $\lesssim \|\psi''(v)\|^*_{\tau+\infty}\gamma^2$, where we have used that $|\psi'(v)| \leq \psi''(v)$ on all coordinates. Summing all the previous bounds gives the desired result. □

Applying Lemma 4.39 allows us to show that the potential is bounded in expectation.

Corollary 4.40. *In the notation of Lemma 4.39 we have for sufficiently large C that*

$$\mathbb{E}[\Psi(x^{(\text{new})}, s^{(\text{new})}, \mu^{(\text{new})})] \leq \left(1 - \frac{\alpha^2\lambda\gamma}{32C\sqrt{n}}\right)\Psi(x, s, \mu) + m.$$

Proof. We use [BLN+20, Lemma 4.36] and verify that the guarantees of Lemma 4.39 satisfy the hypotheses. In that notation, we have that $(1 - c_1) = \alpha/4$, $\delta = \gamma/10$, and c_2 as the implicit constant in the $O(\|\psi''(v)\|^*_{\tau+\infty}\gamma^2)$ of Lemma 4.39. Note that $\gamma = \frac{\varepsilon}{C\lambda} = \frac{\alpha}{C^2\gamma}$. Therefore, for sufficiently large C we have

$$2\lambda\delta + c_2\lambda\gamma \leq \alpha/8 = .5(1 - c_1).$$

Now, $u = \frac{1}{4C_{\text{norm}}\sqrt{n}} = \frac{\alpha}{4C\sqrt{n}}$, as $\|\vec{1}\|_{\tau+\infty} \leq 2C_{\text{norm}}(n + \|v\|_1)^{\frac{1}{2}} \leq 4C_{\text{norm}}\sqrt{n}$. Therefore,

$$.5(1 - c_1)\lambda\gamma u = \frac{\alpha^2\lambda\gamma}{32C\sqrt{n}}.$$

as desired. □

We have the necessary lemmas to show Lemma 4.12.

Proof of Lemma 4.12. The iteration complexity is clear by the definition of r in Algorithm 1. The conditions on $x^{(\text{final})}$ follow from Lemma 4.11.

We proceed by induction. (Slack feasibility) follows by the fact that $\mathbf{A}^\top \overline{d}_s = 0$ in Algorithm 1. (Approximate feasibility) follows by induction and Lemma 4.38 with probability at least $1 - m^{-10}$ per step. To show the (Potential function) bound, we first verify the base case. Indeed, because $(x^{(\text{init})}, s^{(\text{init})}, \mu)$ is $\varepsilon/C_{\text{start}}$-centered, we know that

$$\Psi(x, s, \mu) \leq m \exp(\lambda\varepsilon/C_{\text{start}}) \leq m^2$$

for sufficiently large C_{start} compared to C. The inductive step follows from Corollary 4.40, and that $\frac{32Cm\sqrt{n}}{\alpha^2\lambda\gamma} \leq m^2$ for sufficiently large m, n. Therefore, by Markov's inequality, with probability $1 - m^{-7}$ we have that $\mathbb{E}[\Psi(x, s, \mu)] \leq m^{10}$ for all steps. Now, as $\exp(\lambda\varepsilon) > m^{10}$, we know that (x, s, μ) will be ε-centered. ☐

4.6 Sampling Schemes

In this section, we analyze two sampling schemes. All proofs are deferred to Appendix A.4. The first samples each coordinate independently, and is efficiently implementable in the graphical setting.

Lemma 4.41 (Independent sampling)**.** *Let vector $q \in \mathbb{R}^m_{\geq 0}$ satisfy*

$$q_i \geq C_{\text{valid}}^2 \gamma^{-1} |(\delta_r)_i| + C_{\text{sample}} \sigma (\overline{\mathbf{T}}^{-\frac{1}{2}} \Phi''(\overline{x})^{-\frac{1}{2}} \mathbf{A})_i \log(m) \gamma^{-2}$$

for sufficiently large C_{sample}. Then picking $\mathbf{R}_{ii} = 1/\min(q_i, 1)$ with probability $\min(q_i, 1)$ and 0 otherwise is a C_{valid}-valid (Definition 4.13).

A different sampling scheme is sampling proportional to weights q_i – this is useful for general linear programs, and is implemented in Appendix B.

Lemma 4.42 (Proportional sampling)**.** *Let vector $q \in \mathbb{R}^m_{\geq 0}$ satisfy*

$$q_i \geq |(\delta_r)_i| + \sigma (\overline{\mathbf{T}}^{-\frac{1}{2}} \Phi''(\overline{x})^{-\frac{1}{2}} \mathbf{A})_i.$$

Let $S \geq \sum_i q_i$. Let X be a random variable which equals $q_i^{-1} e_i$ (e_i is the standard basis vector) with probability q_i/S for all i, and $\vec{0}$ otherwise. For $C_0 = 100 C_{\text{valid}}^4 \gamma^{-2} \log(m)$ let $\mathbf{R} = C_0^{-1} \sum_{j=1}^{C_0 S} X_j$, where X_j are i.i.d. copies of X. Then $\mathbf{R}$ is a C_{valid}-valid distribution (Definition 4.13).

It is convenient for our sampling data structures to sample by the ℓ_2 norm in δ_r, instead of the ℓ_1 norm as described in Lemma 4.41 and 4.42. We can achieve this by the following observation.

Corollary 4.43 (Sampling by a mixture of ℓ_2 and uniform)**.** *Let C_1, C_2, C_3 be constants such that $C_3 \geq 4C_{\text{sample}}$ and $C_1 C_2 \geq C_{\text{valid}}^4 \gamma^{-2}$ and*

$$p_i = C_1 \sqrt{n} (\delta_r)_i^2 + C_2/\sqrt{n} + C_3 \tau_i \gamma^{-2} \log m.$$

Then $p_i \geq q_i$ in each of Lemma 4.41 and 4.42. Hence replacing q_i with p_i in Lemma 4.41 and 4.42 and sampling acoordingly gives a valid distribution (Definition 4.13). Additionally

$$\sum_{i \in [m]} p_i \leq \left((C_1 + C_2) \frac{m}{\sqrt{n}} + C_3 n \gamma^{-2} \log m \right). \tag{25}$$

4.7 Additional Properties of the IPM

Even though random sampling may make the sequence of points x and weights τ change more rapidly, we can still argue that there is a nearby sequence of points $\widehat{x}, \widehat{\tau}$ that is more stable. We start with the stability of $\widehat{x}$. The proofs of the following lemmas are given in Appendix A.5.

Lemma 4.44 (Nearby stability of x)**.** *Suppose that* $\mathbf{R}$ *is sampled from a* C_{valid}*-valid distribution for* $C_{\text{valid}} \geq \beta^{-2}\log(mT)$ *where* $\beta \in (0,\gamma)$*. Let* $(x^{(k)}, s^{(k)})$ *for* $k \in [T]$ *be the sequence of points found by Algorithm 1. With probability* $1 - m^{-10}$*, there is a sequence of points* $\widehat{x}^{(k)}$ *from* $1 \leq k \leq T$ *such that*

- $\|\Phi''(x^{(k)})^{\frac{1}{2}}(\widehat{x}^{(k)} - x^{(k)})\|_\infty \leq \beta/2.$
- $\|\Phi''(\widehat{x}^{(k)})^{\frac{1}{2}}(\widehat{x}^{(k)} - x^{(k)})\|_\infty \leq \beta.$
- $\|\Phi''(x^{(k)})^{\frac{1}{2}}(\widehat{x}^{(k+1)} - \widehat{x}^{(k)})\|_{\tau(\widehat{x}^{(k)})+\infty} \leq 2\gamma.$

The stable sequence $\widehat{x}$ induces corresponding stable sequences for ϕ'' and τ.

Lemma 4.45 (Nearby stability of ϕ'' and τ)**.** *In the setup of Lemma 4.44, and* $\widehat{w}^{(k)} = \Phi''(\widehat{x}^{(k)})^{-\frac{1}{2}}\tau(\widehat{x}^{(k)})^{\frac{1}{2}-\frac{1}{p}}$*, we have the following.*

- $\|\Phi''(\widehat{x}^{(k)})^{\frac{1}{2}}(\phi''(\widehat{x}^{(k)})^{-\frac{1}{2}} - \phi''(x^{(k)})^{-\frac{1}{2}})\|_\infty \leq \beta.$
- $\|\Phi''(\widehat{x}^{(k)})^{\frac{1}{2}}(\phi''(\widehat{x}^{(k+1)})^{-\frac{1}{2}} - \phi''(\widehat{x}^{(k)})^{-\frac{1}{2}})\|_{\tau(\widehat{x}^{(k)})+\infty} \lesssim \gamma.$
- $\|\mathbf{T}(\widehat{x}^{(k)})^{-1}(\tau(\widehat{x}^{(k+1)}) - \tau(\widehat{x}^{(k)}))\|_{\tau(\widehat{x}^{(k)})+\infty} \lesssim \gamma.$
- $\|(\widehat{\mathbf{W}}^{(k)})^{-1}(\widehat{w}^{(k+1)} - \widehat{w}^{(k)})\|_{\tau(\widehat{x}^{(k)})+\infty} \lesssim \gamma.$

To bound the bit complexity that our algorithms must maintain throughout, we show that the Hessian of points x encountered along the central path is bounded by polynomial factors in n, m, and $\mu^{(\text{final})}/\mu^{(\text{init})}$.

Lemma 4.46 (Parameter changes along central path)**.** *For* $\mathbf{A} \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^n, c \in \mathbb{R}^m$ *and* $\ell, u \in \mathbb{R}^m$ *assume that the point* $x^{(\text{init})} = (\ell + u)/2$ *is feasible, i.e.* $\mathbf{A}^\top x^{(\text{init})} = b$*. Let* W *be the ratio of the largest to smallest entry of* $\phi''(x^{(\text{init})})^{1/2}$*, and let* W' *be the ratio of the largest to smallest entry of* $\phi''(x)^{1/2}$ *encountered in Algorithm 2. Then*

$$\log W' = \widetilde{O}\left(\log W + \log(1/\mu^{(\text{final})}) + \log\|c\|_\infty\right).$$

5 Maintaining Regularized Lewis-Weights

In this section we show how to efficiently maintain an approximation of the regularized ℓ_p-Lewis weight of $\mathbf{GA}$ under updates to $\mathbf{G} = \mathbf{Diag}(g)$ for $p \in [1/2, 2)$. Lewis weights are a generalization of leverage scores, i.e. for $p = 2$ the two concepts are identical. Correspondingly, we obtain our regularized Lewis weight data structure by reducing to a regularized leverage score data structure presented in Appendix C. Our exact result for maintaining regularized Lewis weights is as follows.

Theorem 5.1. *Assume there exists a* (P, c, Q)*-HeavyHitter data structure (Definition 3.1) and let* $z \geq n\cdot c/\|c\|_1 + n/m$*. Let* $\tau(\mathbf{GA}) \in \mathbb{R}^m$ *such that* $\tau(\mathbf{GA}) = \sigma(\tau(\mathbf{GA})^{1/2-1/p}\mathbf{GA}) + z$ *for* $p \in (0, 2]$*, i.e.* $\tau(\mathbf{GA})$ *are regularized Lewis weights of* $\mathbf{GA}$*. There exists a Monte-Carlo data-structure (Algorithm 3), that works against an adaptive adversary, with the following procedures:*

- *Initialize*$(\mathbf{A} \in \mathbb{R}^{m\times n}, g \in \mathbb{R}^m_{\geq 0}, z \in \mathbb{R}^m_{>0}, p \in [1/2, 2), \delta > 1, \epsilon \in (0, \frac{1}{2^{10}\delta\cdot\log n}])$*: The data structure initializes for the given matrix* $\mathbf{A} \in \mathbb{R}^{m\times n}$*, scaling* $g \in \mathbb{R}^m_{\geq 0}$*, regularization parameter* $z \in \mathbb{R}^m$*, Lewis weight parameter* $p \in (0, 2]$*, and target accuracy* $\epsilon \in (0, \frac{1}{2^{10}\delta\cdot\log n}]$*.*

The parameter δ is a bound on how much the vector g is allowed to change per iteration: Let $g^{(t)}$ be the vector g during the t-th call to Query *(with $g^{(0)}$ during* Initialization*), then we assume that $g^{(t)} \approx_{\delta\epsilon} g^{(t-1)}$ for all t.*

- Scale$(i \in [m], b \in \mathbb{R}_{\geq 0})$*: Sets $g_i \leftarrow b$.*
- Query$()$*: W.h.p. in n the data structure outputs a vector $\overline{\tau} \in \mathbb{R}^m$ with the property $\overline{\tau} \approx_\epsilon \sigma(\overline{\tau}^{1/2-1/p}\mathbf{GA}) + z$ and therefore $\overline{\tau} \approx_\epsilon \tau(\mathbf{GA})$. The vector $\overline{\tau}$ is returned as a pointer and the data structure also returns a set $I \subset [m]$ of indices i where $\overline{\tau}_i$ has changed compared to the last call to* Query*.*

The amortized complexities of this data structure depends on the parameters P, c, Q of the HeavyHitter data structure. Further, they require that some additional properties are satisfied. These properties and the resulting amortized complexities are stated in Theorem 5.2.

Theorem 5.2. *Consider the data structure of Theorem 5.1 (Algorithm 3) and let P, c, Q be the parameters of the HeavyHitter data structure (Definition 3.1). Let $g^{(t)}$ be the vectors g in Theorem 5.1 during the t-th call to query (and $t = 0$ during the initialization) and let $\overline{\tau}^{(t)}$ be the returned vector. Further, assume the following:*

1. *For any given $\overline{\mathbf{W}} \in \mathbb{R}^m_{\geq 0}$ we can solve linear systems in $(\mathbf{A}^\top \overline{\mathbf{W}} \mathbf{A})^{-1}$ with $\epsilon/(64n)$ accuracy (i.e. for input b we can output $\overline{\mathbf{H}} b$ for some $\overline{\mathbf{H}} \approx_{\epsilon/(64n)} (\mathbf{A}^\top \overline{\mathbf{W}} \mathbf{A})^{-1}$) in $\widetilde{O}(P + \Psi + \operatorname{nnz}(\overline{\mathbf{W}}\mathbf{A}))$ time. Further, if*

$$\mathbf{A}^\top \overline{\mathbf{W}} \mathbf{A} \approx_{1/\log n} \mathbf{A}^\top (\overline{\mathbf{T}}^{(t-1)})^{1-2/p} (\mathbf{G}^{(t)})^2 \mathbf{A}$$

   for some t, then the required time is only $\widetilde{O}(\Psi + \operatorname{nnz}(\overline{\mathbf{W}}\mathbf{A}))$.
2. *There exists a sequence $\widetilde{g}^{(t)}$ such that for all t*

$$g^{(t)} \in (1 \pm 1/(10^5 \log n))\ \widetilde{g}^{(t)} \tag{26}$$

$$\|(\widetilde{w}^{(t)})^{-1}(\widetilde{w}^{(t)} - \widetilde{w}^{(t+1)})\|_{\tau(\widetilde{\mathbf{G}}^{(t)}\mathbf{A})} = O(1) \tag{27}$$

   where we define $\widetilde{w}^{(t)} := \tau(\widetilde{\mathbf{G}}^{(t)}\mathbf{A})^{1/2-1/p}\widetilde{g}^{(t)}$.

Then the following time complexities hold:

- Initialize *takes $\widetilde{O}(P + \epsilon^{-2}(\Psi + \operatorname{nnz}(\mathbf{A})))$ time.*
- Scale$(i, \cdot)$ *takes $\widetilde{O}(\frac{\|c\|_1}{n\epsilon^{O(\log\delta)}}\tau(\mathbf{GA})_i)$ amortized time.*
- Query *takes $\widetilde{O}(\Psi\epsilon^{-2}\log^3\delta + \epsilon^{-4}n(\max_i \operatorname{nnz}(a_i))\log^5\delta + \epsilon^{-6}\sqrt{P\|c\|_1/n}\log^4\delta + Q\log\delta)$ amortized time.*

The idea of the data structure (i.e. the reduction to leverage scores) is as follows. It is known that the map $w \mapsto (w^{2/p-1}\sigma(\mathbf{W}^{1/2-1/p}\mathbf{A}))^{p/2}$ moves w closer to the Lewis weight $\tau(\mathbf{A})$ [CP15]. We show in Lemma 5.3 that the same is true for the regularized Lewis weight, even when using only an approximation $\overline{\sigma} \approx \sigma(\mathbf{W}^{1/2-1/p}\mathbf{A}) + z$. The proof follows directly from the techniques in [CP15], and we state the proof in Section 5.1 for completeness sake.

Lemma 5.3. *Let $w, z, \overline{\sigma} \in \mathbb{R}^m_{>0}$, $\epsilon, \gamma > 0$ and $p \in (0, 2)$ with $w \approx_\epsilon \sigma(\mathbf{W}^{1/2-1/p}\mathbf{A}) + z$ and $\overline{\sigma} \approx_\gamma \sigma(\mathbf{W}^{1/2-1/p}\mathbf{A}) + z$. Define $w' = (w^{2/p-1}\overline{\sigma})^{p/2}$, then $w' \approx_{(\epsilon+\gamma)|1-p/2|+\gamma} \sigma(\mathbf{W}'^{1/2-1/p}\mathbf{A})$ and $w \approx_{(\epsilon+\gamma)p/2} w'$.*

We leverage Lemma 5.3 critically in our data structure. To illustrate this, consider a call to Query, et g be the current state of vector g , and let g' be the state of g during the previous call to Query. Further, assume that the previous call to Query returned $w \approx_\epsilon \sigma(\mathbf{W}^{1/2-1/p}\mathbf{G}'\mathbf{A}) + z$. The assumption $g \approx_\delta g'$ (see Theorem 5.1) then implies then $w \approx_{5\delta\epsilon} \sigma(\mathbf{W}^{1/2-1/p}\mathbf{GA}) + z$. Consequently, our task is simply to counter-act this decrease in approximation quality.

For $\overline{v}^{(1)} := w$, $\overline{\sigma}^{(i)} \approx \sigma((\overline{\mathbf{V}}^{(i)})^{1/2-1/p}\mathbf{GA}) + z$, and $\overline{v}^{(i+1)} = ((\overline{v}^{(i)})^{2/p-1}\overline{\sigma}^{(i)})^{p/2}$, we have $\overline{v}^{(L)} \approx_\epsilon \sigma((\overline{\mathbf{V}}^{(L)})^{1/2-1/p}\mathbf{GA})$ for some $L = O(\log \delta)$. Thus we can maintain an $(1 \pm \epsilon)$-approximation of the regularized Lewis weight by maintaining L many leverage scores via the data structure of Theorem C.1. A formal specification of this algorithm is given in Algorithm 3, a proof of correctness is given in Lemma 5.4, and the complexity analysis is given in Section 5.2.

Algorithm 3: Algorithm of Theorem 5.1

members
  D_j for $j = 1, ..., O(1/p)$ `// Data structure of Theorem C.1`
  $\overline{v}^{(j)} \in \mathbb{R}^m$ for $j = 1, ..., O(1/p)$
  $g \in \mathbb{R}^m$, $p \in [1/2, 2)$, $L \in \mathbb{N}$, $\epsilon > 0$
procedure
  Initialize($\mathbf{A} \in \mathbb{R}^{m \times n}, g \in \mathbb{R}^m, z \in \mathbb{R}^m, p \in [1/2, 2), \delta > 1, \epsilon \in (0, 1/(2^{10}\delta \log n))$])
  Compute $\overline{v}^{(1)}$ with $\overline{v}^{(1)} \approx_\epsilon \sigma((\overline{\mathbf{V}}^{(1)})^{1/2-1/p}\mathbf{GA}) + z$
  $L \leftarrow \lceil (\log_{4/3}(200\delta) + 1 \rceil$
  for $j = 1, ..., L$ **do**
    D_j.Initialize($\mathbf{A}, (\overline{\mathbf{V}}^{(j)})^{1/2-1/p} g, z, \epsilon/(40L)$)
    $\overline{v}^{(j+1)} \leftarrow ((\overline{v}^{(j)})^{2/p-1} D_j.\text{Query}())^{p/2}$
  $g \leftarrow g$, $p \leftarrow p$, $\epsilon \leftarrow \epsilon$
procedure *Scale*(i, b)
  $g_i \leftarrow b$
  D_j.Scale($i, (\overline{v}_i^{(j)})^{1/2-1/p} b$) for $j = 1, ..., L$
procedure *Query*()
  `// Maintain` $\overline{v}^{(j+1)} = ((\overline{v}^{(j)})^{2/p-1}\overline{\sigma}^{(j)})^{p/2}$ `for` $j = 1, ..., L-1$
  for $j = 1, ..., L-1$ **do**
    $I_{\overline{\sigma}}^{(j)}, \overline{\sigma}^{(j)} \leftarrow D_j.\text{Query}()$
    $\overline{v}_i^{(j+1)} \leftarrow ((\overline{v}_i^{(j)})^{2/p-1}\overline{\sigma}_i^{(j)})^{p/2}$ for $i \in I_{\overline{\sigma}}^{(j)}$
    D_{j+1}.Scale($i, (\overline{v}_i^{(j+1)})^{1/2-1/p} g_i$) for $i \in I_{\overline{\sigma}}^{(j)}$
  `// Maintain` $\overline{v}_i^{(1)} \approx_\epsilon \tau(\mathbf{GA})_i$ `for all` $i \in [m]$`,`
  `// but update only if` $\tau(\mathbf{GA})_i$ `changed sufficiently.`
  for $i \in I_{\overline{\sigma}}^{(L-1)}$ *with* $\overline{v}_i^{(L)} \not\approx_{\epsilon/10} \overline{v}_i^{(1)}$ **do**
    $\overline{v}_i^{(1)} \leftarrow \overline{v}_i^{(L)}$
    D_1.Scale($i, (\overline{v}_i^{(1)})^{1/2-1/p} g_i$)
  return $I_{\overline{\sigma}}^{(L)}, \overline{v}^{(1)}$

5.1 Correctness

We start by proving Lemma 5.3. We then show in Lemma 5.4 that Algorithm 3 indeed maintains an $\exp(\pm\epsilon)$-approximation of the regularized Lewis weight.

Proof of Lemma 5.3. We first show that w and w' are close to each other. By the definition of w'

$$\frac{w'}{w} = \frac{(w^{2/p-1}\overline{\sigma})^{p/2}}{w} = \left(\frac{\overline{\sigma}}{w}\right)^{p/2}.$$

Thus by $w \approx_\epsilon \sigma(\mathbf{W}^{1/2-1/p}\mathbf{A}) + z \approx_\gamma \overline{\sigma}$ we have

$$w' \approx_{(\epsilon+\gamma)p/2} w. \tag{28}$$

Fix any $i \in [m]$. By definition of leverage score we have that

$$\sigma(\mathbf{W}^{1/2-1/p}\mathbf{A})_i + z_i = w_i^{1-2/p}(\mathbf{A}(\mathbf{A}^\top \mathbf{W}^{1-2/p}\mathbf{A})^{-1}\mathbf{A}^\top)_{i,i} + z_i$$

This allows us to transform w' as follows

$$\begin{aligned}
[w_i']^{2/p} &= w_i^{2/p-1}\overline{\sigma}_i \\
&\approx_\gamma w_i^{2/p-1}(\sigma(\mathbf{W}^{1/2-1/p}\mathbf{A})_i + z_i) \\
&= (\mathbf{A}(\mathbf{A}^\top \mathbf{W}^{1-2/p}\mathbf{A})^{-1}\mathbf{A}^\top)_{i,i} + w_i^{2/p-1} z_i
\end{aligned} \tag{29}$$

At last, we analyze the approximation ratio of w' to its leverage score

$$\begin{aligned}
\frac{\sigma(\mathbf{W}'^{1/2-1/p}\mathbf{A})_i + z_i}{w_i'} &= \frac{w_i'^{1-2/p}(\mathbf{A}(\mathbf{A}^\top \mathbf{W}'^{1-2/p}\mathbf{A})^{-1}\mathbf{A}^\top)_{i,i} + z_i}{w_i'} \\
&\approx_\gamma \frac{(\mathbf{A}(\mathbf{A}^\top \mathbf{W}'^{1-2/p}\mathbf{A})^{-1}\mathbf{A}^\top)_{i,i} + w_i'^{2/p-1} z_i}{(\mathbf{A}(\mathbf{A}^\top \mathbf{W}^{1-2/p}\mathbf{A})^{-1}\mathbf{A}^\top)_{i,i} + w_i^{2/p-1} z_i} \\
&\approx_{(\epsilon+\gamma)(p/2)|2/p-1|} \frac{(\mathbf{A}(\mathbf{A}^\top \mathbf{W}'^{1-2/p}\mathbf{A})^{-1}\mathbf{A}^\top)_{i,i} + w_i'^{2/p-1} z_i}{(\mathbf{A}(\mathbf{A}^\top \mathbf{W}'^{1-2/p}\mathbf{A})^{-1}\mathbf{A}^\top)_{i,i} + w_i'^{2/p-1} z_i} = 1
\end{aligned}$$

where in step 2 we used (29) and in step 3 we used (28). Thus

$$w_i' \approx_{(\epsilon+\gamma)|1-p/2|+\gamma} \sigma(\mathbf{W}'^{1/2-1/p}\mathbf{A})_i + z_i.$$

□

Next we show that our sequence of contractions $\overline{v}^{(1)}, \overline{v}^{(2)}, ...,$ improve the approximation quality sufficiently to counter-act the impact of changing g.

Lemma 5.4. *Let $\gamma = \epsilon/(40L)$ be the accuracy used for the leverage score data structures in Line 9 of Algorithm 3. After each call to* Query *we have*

$$\overline{v}^{(i)} \approx_{5\delta\epsilon(3/4)^{i-1}+\gamma i} \sigma((\overline{\mathbf{V}}^{(i)})^{1/2-1/p}\mathbf{G}\mathbf{A}) + z$$

for $i > 1$ and $\overline{v}^{(1)} \approx_\epsilon \sigma((\overline{\mathbf{V}}^{(1)})^{1/2-1/p}\mathbf{G}\mathbf{A}) + z$.

Proof. We start the proof by induction over the number of calls to Query. Directly after the initialization (i.e. zero calls to Query) we have $\overline{v}^{(1)} \approx_\epsilon \sigma((\overline{\mathbf{V}}^{(1)})^{1/2-1/p}\mathbf{G}\mathbf{A}) + z$ by Line 6. Further, each $\overline{v}^{(j+1)}$ for $j \geq 1$ is defined via $\overline{v}^{(j+1)} = ((\overline{v}^{(j)})^{2/p-1}\overline{\sigma}^{(j)})^{p/2}$ where $\overline{\sigma}^{(j)} \approx_\gamma \sigma((\overline{\mathbf{V}}^{(j)})^{1/2-1/p}\mathbf{G}\mathbf{A}) + z$ is the output of the leverage score data structure D_j. Thus by Lemma 5.3 we have $\overline{v}^{(j)} \approx_{\epsilon(1-p/2)^{j-1}+\gamma\sum_{i=0}^{j-1}(1-p/2)^i} \sigma((\overline{\mathbf{V}}^{(j)})^{1/2-1/p}\mathbf{G}\mathbf{A}) + z$. We can bound this approximation quality via $\epsilon(1-p/2)^{j-1} + \gamma\sum_{i=0}^{j-1}(1-p/2)^i \leq \epsilon(3/4)^{j-1}j\gamma$ since $p \in [1/2, 2)$.

Next, we consider a call to Query. Note that at the start of executing Query, we have $\overline{v}^{(1)} \approx_{5\delta\epsilon} \sigma((\overline{\mathbf{V}}^{(1)})^{1/2-1/p}\mathbf{G}\mathbf{A}) + z$, by induction hypothesis and because g can change by at most a $\exp(\pm\delta\epsilon)$ factor (see Initialize in Theorem 5.1).

By the recursive definition of the $\overline{v}^{(j)}$ we then have

$$\overline{v}^{(j)} \approx_{5\delta\epsilon(3/4)^{j-1}+\gamma j} \sigma((\overline{\mathbf{V}}^{(j)})^{1/2-1/p}\mathbf{G}\mathbf{A})$$

for $j > 1$ at the end of Query. For $L = \lceil(\log_{4/3}(200\delta) + 1\rceil$ and $\gamma \leq \epsilon/(40L)$ we have

$$5\delta\epsilon(4/3)^{L-1} + \gamma L \leq \frac{5\delta\epsilon}{200\delta} + \epsilon/40 \leq \epsilon/20.$$

Thus $\overline{v}^{(L)} \approx_{\epsilon/20} \sigma((\overline{\mathbf{V}}^{(L)})^{1/2-1/p}\mathbf{GA}) + z$.

The vector $v^{(1)}$ is modified again at the end of QUERY in Line 21. This update to $\overline{v}^{(1)}$ is only performed, if $\overline{v}_i^{(L)}$ and $\overline{v}^{(1)}$ differ by at least an $\exp(\pm\epsilon/10)$ factor (see Line 20). Thus

$$\overline{v}^{(1)} \approx_{\epsilon/10} \overline{v}^{(L)} \approx_{\epsilon/20} \sigma((\overline{\mathbf{V}}^{(L)})^{1/2-1/p}\mathbf{GA}) + z \approx_{6\epsilon/10} \sigma((\overline{\mathbf{V}}^{(1)})^{1/2-1/p}\mathbf{GA}) + z \tag{30}$$

where we used $p \leq 1/2$. In summary, $\overline{v}^{(1)} \approx_\epsilon \sigma((\overline{\mathbf{V}}^{(1)})^{1/2-1/p}\mathbf{GA}) + z$. □

As the data structure returns $\overline{v} := \overline{v}^{(1)}$ we have $\overline{v} \approx_\epsilon \tau(\overline{\mathbf{V}}^{1/2-1/p}\mathbf{GA}) + z$, which concludes the proof of Theorem 5.1. Next, we analyze the complexity of the data structure in Section 5.2.

5.2 Complexity

The main difficulty in analyzing the complexity of our data structure is to bound the complexity impact of all the calls to D_j.SCALE that occur during a call to QUERY. To bound this complexity we the following partial results:

First, we use that $\overline{\sigma}^{(L)}$ is a good approximation of the exact regularized Lewis weight $\tau(\mathbf{GA})$. Thus Line 22 is only executed for i, where $\tau(\mathbf{GA})_i$ changed by a sufficiently large amount, because of the condition in Line 20. Thus we can bound the total complexity impact of all calls to D_1.SCALE$(i, \cdot)$ for some i via the stability property (27).

Second, to bound how often D_{j+1}.SCALE$(i, \cdot)$ is called for $j > 0$, note that we perform such a call whenever the output $\overline{\sigma}_i^{(j)}$ of the leverage score data structure D_j (Theorem C.1) changes. Such a bound on how often the output changes is given by Lemma C.12 which bounds the number of changes to the output $\overline{\sigma}^{(j)}$ relative to the number of changes to the input, i.e. how often D_j.SCALE was called. By propagating, we are able to bound the number of calls to any D_{j+1}.SCALE relative to the number of calls to D_1.SCALE.

At last, we require complexity bounds for the data structure D_j that maintain leverage scores. The complexities are given in Theorem C.2.

We start the formal proof of the complexity analysis by showing in Lemma 5.5 that an approximate regularized Lewis weight $w \approx_\epsilon \sigma(\mathbf{W}^{1/2-1/p}\mathbf{A}) + z$ is also close to the exact regularized Lewis weight $w \approx_\epsilon \tau(\mathbf{A})$. The following Lemma 5.5 follows directly from techniques in [CP15].

Lemma 5.5. *For any $w, z \in \mathbb{R}^m_{>0}$, $\epsilon > 0$ and $p \in (0, 2]$ with $w \approx_\epsilon \sigma(\mathbf{W}^{1/2-1/p}\mathbf{A}) + z$ we have $w \approx_\epsilon \tau(\mathbf{A})$.*

Proof. Define $w^{(0)} := w$ and $w^{(k+1)} = ((w^{(k)})^{2/p-1}(\sigma((\mathbf{W}^{(k)})^{1/2-1/p}\mathbf{A}) + z))^{p/2}$. By Lemma 5.3 we have $\lim_{k\to\infty} w^{(k)} = \tau(\mathbf{A})$. Further we have $w^{(k+1)} \approx_{\epsilon|1-p/2|^k p/2} w^{(k)}$. Thus

$$\tau(\mathbf{A}) = \lim_{k\to\infty} w^{(k)} \approx_{\epsilon(p/2)\sum_{i\geq 0}|1-p/2|^i} w^{(0)},$$

where $\epsilon(p/2)\sum_{i\geq 0}|1-p/2|^i = \epsilon(p/2)/(1-|1-p/2|) = \epsilon$ for $p \leq 2$. □

The input to the leverage score data structures D_j is the matrix $(\overline{\mathbf{V}}^{(j)})^{1/2-1/p}\mathbf{GA}$, and the complexity bounds of the leverage score data structure as stated in Theorem C.2 only hold, if Condition 1 and 2 (as stated in Theorem C.2) are satisfied. Lemma 5.6 shows that these requirements hold, if Condition 1 and 2 of Theorem 5.2 are satisfied.

Lemma 5.6. *Let $(\overline{v}^{(j)})^{(t)}$ be the vector $\overline{v}^{(j)}$ when performing the $t+1$-th call to D_j.QUERY() (i.e. during the t-th call to QUERY of Algorithm 3). Let $g^{(t)}$ be the vector g during the t-th call to QUERY. Let $\widetilde{g}^{(t)}$ be the vector assumed by (26) and let $\widetilde{w}^{(t)} := \tau(\widetilde{\mathbf{G}}^{(t)}\mathbf{A})\widetilde{g}^{(t)}$. Then we have*

$$((\overline{\mathbf{V}}^{(j)})^{(t)})^{1/2-1/p} g^{(t)} \in (1 \pm 1/(64\log n))\widetilde{\mathbf{W}}^{(t)},$$

i.e. Condition 2 of Theorem C.2 is satisfied.

Further, if Condition 1 of Theorem 5.1 holds true, then Condition 1 of Theorem C.2 holds true for all instances $D_1, ..., D_L$ of Theorem C.1.

Proof. We start by analyzing the sequence $\widetilde{w}^{(0)}, \widetilde{w}^{(1)}, ...$

Sequence By Lemma 5.4 we have that

$$(\overline{v}^{(j)})^{(t)} \approx_{6\delta\epsilon} \sigma(((\overline{\mathbf{V}}^{(j)})^{(t)})^{1/2-1/p}\mathbf{G}^{(t)}\mathbf{A}) + z$$

for all $j = 1, ..., L$. Thus by Lemma 5.5, $p \in [1/2, 2]$, and assumption $\epsilon \leq 1/(2^{10}\delta \log n)$ (see Theorem 5.1) we have

$$((\overline{v}^{(j)})^{(t)})^{1/2-1/p} \approx_{1/(96\log n)} (\tau(\mathbf{G}^{(t)}\mathbf{A}))^{1/2-1/p}.$$

Via (26) and $1/2 - 1/p \leq 1/4$ we have $\tau(\mathbf{G}^{(t)}\mathbf{A})^{1/2-1/p} \approx_{1/(10^5 \log n)} \tau(\widetilde{\mathbf{G}}^{(t)}\mathbf{A})^{1/2-1/p}$ which results in

$$((\overline{\mathbf{V}}^{(j)})^{(t)})^{1/2-1/p} g^{(t)} \in (1 \pm 1/(64\log n))\widetilde{w}^{(t)}.$$

Solvers During the t-th call to QUERY, when the algorithms calls D_j.QUERY() in Line 17, it uses $(\overline{\mathbf{V}}^{(j)})^{1/2-1/p} g^{(t)}$ as scale-vector. Here $\overline{v}^{(1)}$ is exactly the vector returned by the previous call to QUERY. Thus if we have a solver as assumed in Condition 1 of Theorem 5.1, then we also have a solver as assumed in Condition 1 of Theorem C.2 for instance D_1 of Theorem C.1.

For the other instances $D_2, ..., D_L$ consider the following. By Lemma 5.4 we have for $j > 1$ that

$$\overline{v}^{(j)} \approx_{6\delta\epsilon} \sigma((\overline{\mathbf{V}}^{(j)})^{1/2-1/p}\mathbf{G}^{(t)}\mathbf{A}) + z.$$

As g changes by at most an $\exp(\pm\epsilon\delta)$-factor, we have

$$\overline{v}^{(j)} \approx_{6\delta\epsilon} \tau(\mathbf{G}^{(t)}\mathbf{A}) \approx_{4\delta\epsilon} \tau(\mathbf{G}^{(t-1)}\mathbf{A}) \approx_{\epsilon} \overline{v}^{(1)}.$$

Thus each $\overline{v}^{(j)} \approx_{11\delta\epsilon} \overline{v}^{(1)}$, and by $p \in [1/2, 2]$ we have

$$\mathbf{A}^\top(\overline{\mathbf{V}}^{(j)})^{1-2/p}(\mathbf{G}^{(t)})^2\mathbf{A} \approx_{33\delta\epsilon} \mathbf{A}^\top(\overline{\mathbf{V}}^{(j)})^{1-2/p}(\mathbf{G}^{(t)})^2\mathbf{A}.$$

Note that by the upper bound on ϵ in Theorem 5.1, we have $33\delta\epsilon + 1/(64\log n) < 1/\log n$, so if we have a solver that satisfies Condition 1 of Theorem 5.1, then we also have a solver that satisfies Condition 1 of Theorem C.2 for instance D_j of Theorem C.1.

□

We can now analyze the amortized complexity of Theorem 5.1, i.e. prove Theorem 5.2. This is done by analyzing how often D_j.SCALE and D_j.QUERY (instances of Theorem C.1) are called.

Proof of Theorem 5.2. To bound the complexity of our regularized Lewis weight data structure Algorithm 3 we first bound the total time complexity after T iterations and then charge some of the terms to SCALE and QUERY.

For that we will first state the complexities for the internal data structures D_j used by Algorithm 3.

Complexity of Leverage-Score Data Structures Our regularized Lewis weight data structure (Algorithm 3) uses L instances $D_1, ..., D_L$ of Theorem C.1. The complexity bounds of Theorem C.1 hold, if condition (89) and (90) are satisfied. These conditions are satisfied by Lemma 5.6 and assumption (27).

Since the conditions are satisfied, a call to $D_j.\textsc{Scale}(i, \cdot)$ has amortized cost (by Theorem C.1)

$$\widetilde{O}\left(\frac{\|c\|_1 \log^4 \delta}{n\epsilon^4}\sigma((\mathbf{V}^{(j)})^{1/2-1/p}\mathbf{GA})_i + \frac{c_i \log^2 \delta}{\epsilon^2}\right) = \widetilde{O}\left(\frac{\|c\|_1 \log^4 \delta}{n\epsilon^4}\tau(\mathbf{GA})_i\right)$$

because we use an accuracy parameter of $\epsilon/(40L) = \Omega(\epsilon/\log \delta)$ in the initialization of each D_j, and since $\tau(\mathbf{GA}) \approx \overline{v}^{(j)} \approx \sigma((\overline{\mathbf{V}}^{(j)})^{1/2-1/p}\mathbf{GA}) + z$ by Lemma 5.4 and Lemma 5.5, and because $z \geq n \cdot c/\|c\|_1$. Similarly, a call to $D_j.\textsc{Query}()$ has amortized cost

$$\widetilde{O}\left(\Psi\epsilon^{-2}\log^2 \delta + \epsilon^{-4}n(\max_i \operatorname{nnz}(a_i))\log^4 \delta + \epsilon^{-2}\sqrt{P\|c\|_1/n}\log^2 \delta + Q\right).$$

Total time complexity We now bound the total time spent after T iterations. We will then later charge some of the cost as amortized cost to Scale and Query.

When calling $\textsc{Scale}(i, \cdot)$, the data structure calls $D_j.\textsc{Scale}(i, \cdot)$ for $j = 1, ..., L$ with $L = O(\log \delta)$. Thus we incur the total cost

$$\widetilde{O}\left(\sum_{t=1}^{T}\sum_{i\in[m]}\left(\frac{\|c\|_1 \log^5 \delta}{n\epsilon^4}\tau(\mathbf{G}^{(t-1)}\mathbf{A})_i\right)\mathbf{1}_{g_i^{(t)}\neq g_i^{(t-1)}}\right). \tag{31}$$

When calling Query, the function $D_j.\textsc{Query}()$ is called for every $j = 1, ..., L$ with $L = O(\log \delta)$. Thus we must add

$$\widetilde{O}\left(T \cdot \left(\Psi\epsilon^{-2}\log^3 \delta + \epsilon^{-4}n(\max_i \operatorname{nnz}(a_i))\log^5 \delta + \epsilon^{-2}\sqrt{P\|c\|_1/n}\log^3 \delta + Q\log \delta\right)\right) \tag{32}$$

to the total time complexity.

Further, the data structure calls $D_1.\textsc{Scale}(i, \cdot)$ in Line 22. To bound that complexity impact, we observe that $D_1.\textsc{Scale}(i, \cdot)$ is only called whenever $\overline{v}^{(L)}$ changed by at least an $\exp(\epsilon/10)$-factor (see Line 20). Since

$$v_i^{(L)} \approx_{\epsilon/20} \tau(\mathbf{G})_i,$$

this means that $v_i^{(1)}$ is only updated if $\tau(\mathbf{G})_i$ changed at least an $\exp(\epsilon/10)$-factor. By (26) this means $\tau(\widetilde{\mathbf{G}})_i$ must have changed by at least an $\exp(\epsilon/2000)$-factor. Using the fact that $\tau(\widetilde{\mathbf{G}})$ changes slowly (27) we can thus bound

$$\sum_{t=1}^{T}\sum_{i\in[m]}\tau(\mathbf{G}^{(t)}\mathbf{A})_i\mathbf{1}_{(v_i^{(1)})^{(t)}\neq(v_i^{(1)})^{(t-1)}} = O\left(T^2/\epsilon^2\right).$$

The time complexity incurred by calling $D_1.\textsc{Scale}(i, \cdot)$ in Line 22 is thus bounded by

$$\widetilde{O}\left(\sum_{t=1}^{T}\sum_{i\in[m]}\left(\frac{\|c\|_1 \log^4 \delta}{n\epsilon^4}\tau(\mathbf{G}^{(t-1)}\mathbf{A})_i\right)\mathbf{1}_{(v_i^{(1)})^{(t)}\neq(v_i^{(1)})^{(t-1)}}\right) \leq \widetilde{O}\left(\frac{\|c\|_1 \log^4 \delta}{n\epsilon^6}T^2\right). \tag{33}$$

At last, we are left with bounding the impact of calling $D_j.\textsc{Scale}$ for $j > 1$ in Line 19. Note that $D_j.\textsc{Scale}(i, \cdot)$ is called whenever $\overline{v}_i^{(j-1)}$ changed. By Lemma C.12 we can bound for every j how often often any entry of $\overline{v}^{(j)}$ changes as follows

$$\sum_{t=1}^{T}\sum_{i\in[m]}\tau(\mathbf{G}^{(t)}\mathbf{A})_i\mathbf{1}_{(\overline{v}_i^{(j)})^{(t)}\neq(\overline{v}_i^{(j)})^{(t-1)}}$$

$$\leq O\left(\epsilon^{-1}\sum_{t=1}^{T}\sum_{i\in[m]}\tau(\mathbf{G}^{(t)}\mathbf{A})_i(\mathbf{1}_{((\overline{v}_i^{(j-1)})^{(t)}\neq(\overline{v}_i^{(j-1)})^{(t-1)}}+\mathbf{1}_{g_i^{(t)}\neq g_i^{(t-1)}})\right)$$
$$\leq O\left(\sum_{i=1}^{j}\epsilon^{-i}\sum_{t=1}^{T}\sum_{i\in[m]}\tau(\mathbf{G}^{(t)}\mathbf{A})_i\mathbf{1}_{g_i^{(t)}\neq g_i^{(t-1)}}\right)$$
$$\leq O\left(\epsilon^{-j}\sum_{t=1}^{T}\sum_{i\in[m]}\tau(\mathbf{G}^{(t)}\mathbf{A})_i\mathbf{1}_{g_i^{(t)}\neq g_i^{(t-1)}}\right)$$

where the second step comes from repeatedly applying the first step. Finally, with $j \leq L = O(\log\delta)$ this leads to a complexity cost of

$$\widetilde{O}\left(\frac{\|c\|_1}{n\epsilon^{O(\log\delta)}}\cdot\left(\sum_{t=1}^{T}\sum_{i\in[m]}\tau(\mathbf{G}^{(t)}\mathbf{A})_i\mathbf{1}_{g_i^{(t)}\neq g_i^{(t-1)}}\right)\right). \tag{34}$$

In summary, the total cost after T iterations is bounded by

$$\widetilde{O}(\underbrace{T\cdot\left(\Psi\epsilon^{-2}\log^3\delta+\epsilon^{-4}n(\max_i \mathrm{nnz}(a_i))\log^5\delta+\epsilon^{-2}\sqrt{P\|c\|_1/n}\log^3\delta+Q\log\delta\right)}_{(32)}$$
$$+\underbrace{\frac{\|c\|_1\log^4\delta}{n\epsilon^6}T^2}_{(33)}+\underbrace{\frac{\|c\|_1}{n\epsilon^{O(\log\delta)}}\cdot\left(\sum_{t=1}^{T}\sum_{i\in[m]}\tau(\mathbf{G}^{(t)}\mathbf{A})_i\mathbf{1}_{g_i^{(t)}\neq g_i^{(t-1)}}\right)}_{(31),(34)})$$

Amortized Cost We previously bounded the total time spent after T iterations, we now charge some of the terms as amortized cost to Scale and Query.

The amortized cost of Scale is

$$\widetilde{O}(\frac{\|c\|_1}{n\epsilon^{O(\log\delta)}}\tau(\mathbf{GA})_i)$$

which covers the terms depending on $\mathbf{1}_{g_i^{(t)}\neq g_i^{(t-1)}}$ in (31) and (34). The amortized cost of Query is

$$\widetilde{O}(\frac{\|c\|_1\log^4\delta}{n\epsilon^6}T+\Psi\epsilon^{-2}\log^3\delta+\epsilon^{-4}n(\max_i\mathrm{nnz}(a_i))\log^5\delta+\epsilon^{-2}\sqrt{P\|c\|_1/n}\log^3\delta+Q\log\delta)$$
$$=\widetilde{O}(\Psi\epsilon^{-2}\log^3\delta+\epsilon^{-4}n(\max_i\mathrm{nnz}(a_i))\log^5\delta+\epsilon^{-6}\sqrt{P\|c\|_1/n}\log^4\delta+Q\log\delta)$$

which covers the remaining terms in (32), and (33), when we bound $T\leq\sqrt{Pn/\|c\|_1}$ by restarting the data structure after $\sqrt{Pn/\|c\|_1}$ calls to Query. (The reinitialization cost every T iterations is subsumed by the terms above.)

Initialization The initialization requires us to compute $\overline{v}^{(1)}$. This is done by performing the contraction $w\leftarrow(w^{2/p-1}(\sigma(\mathbf{WGA})+z)^{2/p}$ a total of $O(\log_{|1-p/2|}\epsilon^{-1})=O(\log\epsilon^{-1})=\widetilde{O}(1)$ times. So one could compute this contraction by just initializing $\widetilde{O}(1)$ instances of the leverage score data structure (Theorem C.1) to compute the leverage scores required for the contraction, and then immediately discarding these instances again. The cost for computing this initial regularized Lewis weight $\overline{v}^{(1)}$ is subsumed by initializing the data structures D_j for $j=1,...,L$. Initializing these data structures requires $\widetilde{O}(P\log\delta+(\Psi+\mathrm{nnz}(\mathbf{A}))\epsilon^{-2}\log^3\delta)$ time.

□

6 Path Following

In this section we show how to efficiently implement our IPM which was given by Algorithms 1 and 2 in Section 4. Note that Algorithms 1 and 2 only specify which steps must be performed, but not how they must be implemented. For example Line 5 of Algorithm 1 specifies that one should pick an approximation $\overline{x}$ of the primal solution x, but it is not specified how this approximation must be obtained. Here we show how all the steps of Algorithm 2 can be performed efficiently, if we assume the existence of certain data structures.

These data structure may differ depending on the application, for example the HeavyHitter-problem (Definition 3.1) can be solved more efficiently if the LP is a min-cost flow instance (Lemma F.1) than when the problem is a general LP (Lemma B.1). However, while these data structures have different complexities, they implement the same interfaces (e.g. Definition 3.1). Thus the correctness proof, i.e. showing that Algorithm 2 can be implemented by using these data structures, is the same for min-cost flow and for general LPs. This is why we perform this proof in a generalized way. More accurately, we can show the following theorem.

Theorem 6.1. *Assume there exists a (P, c, Q)-HeavyHitter (Definition 3.1), a (P, c, Q)-InverseMaintenance (Definition 6.2), and a (P, c, Q)-HeavySampler (Definition 6.3). Then we can implement the IPM given by Algorithm 4 (PathFollowing, Lemma 4.12) such that the total time of PathFollowing can be bounded by*

$$\widetilde{O}\left(\left(\sqrt{P\|c\|_1} + \sqrt{n}\left(Q + n \cdot \max_i \operatorname{nnz}(a_i)\right)\right) \log \frac{\mu^{(\text{init})}}{\mu^{(\text{end})}}\right).$$

The implementation is given by Algorithm 4 and Algorithm 5.

In Section 7 we state the resulting complexity when we use data structures optimized for the min-cost flow problem. Data structures for general LPs are a bit slower and the resulting LP solver and their complexity is stated in Section 8.

We start proving Theorem 6.1 by giving a general outline of our implementation in Section 6.1 and listing the assumed data structures that we require. In Section 6.2 we then show that Algorithm 4 and Algorithm 5 do indeed implement the IPM given by Algorithm 4. In Section 6.3 we analyze the resulting complexity, which concludes the proof of Theorem 6.1.

6.1 Outline

Recall that our IPM consists of Algorithm 2 which is esssentially a WHILE-loop that repeatedly calls Algorithm 1. Consequently, we focus on the implementation of Algorithm 1. In order to implement Algorithm 1 (Line 5) we must maintain approximations $\overline{x}$ and $\overline{\tau}$ that satisfy Invariant 4.10. Here $\overline{\tau}$ is an approximation of the regularized Lewis weight $\tau(\overline{x})$ and will be maintained via the data structure of Theorem 5.1 presented in Section 5. When $D^{(\tau)}$ is an instance of the Lewis weight data structure (Theorem 5.1), then all we have to do is call $D^{(\tau)}$.Scale$(i, \overline{x}_i)$ whenever some entry $\overline{x}_i$ changes, and then $D^{(\tau)}$.Query() will return the desired approximation $\overline{\tau}$.

We now explain how to obtain the approximation $\overline{x}$ via the following data structure:

Theorem D.1 (Primal/Gradient Maintenance)**.** *There exists a deterministic data-structure that supports the following operations*

- *Initialize ($\mathbf{A} \in \mathbb{R}^{m \times n}, x^{(\text{init})} \in \mathbb{R}^m, g \in \mathbb{R}^m, \widetilde{\tau} \in \mathbb{R}^m, z \in \mathbb{R}^m, w \in [0,1]^m, \epsilon > 0$): The data-structure preprocesses the given matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, vectors $x^{(\text{init})}, g, \widetilde{\tau}, z \in \mathbb{R}^m$, and the accuracy parameters $w \in [0,1]^m$ and $\epsilon > 0$ in $\widetilde{O}(\operatorname{nnz}(\mathbf{A}))$ time. We denote $\mathbf{G}$ the diagonal matrix $\mathbf{Diag}(g)$. The data-structure assumes $0.5 \leq z \leq 2$ and $n/m \leq \widetilde{\tau} \leq 2$.*
- *Update($i \in [m], a \in \mathbb{R}, b \in \mathbb{R}, c \in \mathbb{R}$): Sets $g_i \leftarrow a$, $\widetilde{\tau}_i \leftarrow b$ and $z_i \leftarrow c$ in $O(\operatorname{nnz}(a_i) + \log n)$ time. The data-structure assumes $0.5 \leq b \leq 2$ and $n/m \leq c \leq 2$.*

- *SetAccuracy*(i, δ) *Sets* $w_i \leftarrow \delta$ *in* $O(\log n)$ *time.*
- *QueryProduct*()*: Returns* $\mathbf{A}^\top \mathbf{G} \nabla \Psi(\overline{z})^{\flat(\overline{\tau})} \in \mathbb{R}^n$ *for some* $\overline{\tau} \in \mathbb{R}^m$, $\overline{z} \in \mathbb{R}^m$ *with* $\overline{\tau} \approx_\epsilon \widetilde{\tau}$ *and* $\|\overline{z} - z\|_\infty \leq \epsilon$*, where*

$$x^{\flat(\overline{\tau})} := \operatorname{argmax}_{\|w\|_{\overline{\tau}+\infty} \leq 1} \langle x, w \rangle.$$

  Every call to QueryProduct must be followed by a call to QuerySum, and we bound their complexity together (see QuerySum).
- *QuerySum*$(h \in \mathbb{R}^m)$*: Let* $v^{(\ell)}$ *be the vector* $\mathbf{G} \nabla \Psi(\overline{z})^{\flat(\overline{\tau})}$ *used for the result of the* ℓ*-th call to QueryProduct. Let* $h^{(\ell)}$ *be the input vector* h *given to the* ℓ*-th call to QuerySum. We define*

$$x^{(t)} := x^{(\text{init})} + \sum_{\ell=1}^{t} \left(v^{(\ell)} + h^{(\ell)} \right).$$

  Then the t*-th call to QuerySum returns a vector* $\overline{x} \in \mathbb{R}^m$ *with*

$$\|w^{-1}(\overline{x} - x^{(t)})\|_\infty \leq \epsilon.$$

  Assuming the input vector h *is given in a sparse representation (e.g. a list of non-zero entries), then after* T *calls to QuerySum and QueryProduct the total time for all calls together is bounded by*

$$O\left(Tn\epsilon^{-2} \log n + \log n \cdot \sum_{\ell=0}^{T} \|h^{(\ell)}\|_0 + T \log n \cdot \sum_{\ell=1}^{T} \|v^{(\ell)}/w^{(\ell-1)}\|_2^2 / \epsilon^2 \right)$$

  The output $\overline{x} \in \mathbb{R}^m$ *is returned in a compact representation to reduce the size. In particular, the data-structure returns a pointer to* $\overline{x}$ *and a set* $J \subset [m]$ *of indices which specifies which entries of* $\overline{x}$ *have changed between the current and previous call to QuerySum.*
- *ComputeExactSum*()*: Returns the exact* $x^{(t)}$ *in* $O(m \log n)$ *time.*
- *Potential*()*: Returns* $\Psi(\overline{z}) = \sum_i \cosh(\lambda \overline{z}_i)$ *in* $O(1)$ *time for some* $\overline{z}$ *with* $\|\overline{z} - z\|_\infty \leq \epsilon$*.*

A variant of this data structure was proven in [BLN+20] to obtain an element-wise $\overline{x} \approx_\epsilon x$ approximation. Here we instead need to obtain $\|\Phi''(x)^{1/2}(\overline{x} - x)\|_\infty \leq \epsilon$. We show in Appendix D how Theorem D.1 is obtained via a small modification to data structure of [BLN+20].

We now explain how Theorem D.1 can be used to maintain the approximate $\overline{x}$. Note that by Line 12 of the IPM (Algorithm 1) the vector x changes via

$$x^{(\text{new})} \leftarrow x + \Phi''(\overline{x})^{-1/2}(\gamma \nabla \Psi(\overline{y})^{\flat(\overline{\tau})} - \mathbf{R}\delta_r).$$

Here $\mathbf{R}\delta_r$ can be written as some vector h and $\Phi''(\overline{x})^{-1/2}\gamma \nabla \Psi(\overline{y})^{\flat(\overline{\tau})}$ can be written as $v := \mathbf{G} \nabla \Psi(\overline{y})^{\flat(\overline{\tau})}$ for $\mathbf{G} := \gamma \Phi''(\overline{x})^{-1/2}$, so the update to x becomes

$$x^{(\text{new})} \leftarrow x + v + h.$$

Note that an approximation of such x satisfying Invariant 4.10 is returned by QuerySum of Theorem D.1 when also calling SetAccuracy appropriately.

Line 6 of the IPM (Algorithm 1) requires an approximation $\overline{y}$ of

$$y = \frac{s + \mu\tau\phi'(x)}{\mu\tau\sqrt{\phi''(x)}}.$$

Such an approximation can be obtained by having a vector $\overline{s}$ with small enough $\|(\mu\tau\sqrt{\phi''(x)})^{-1}(\overline{s} - s)\|_\infty$ and then replacing all τ, s, x in the definition of y by the approximate $\overline{\tau}, \overline{y}, \overline{x}$. This is proven in Lemma 6.6. The required approximation $\overline{s}$ can be obtained via the following data structure:

Theorem E.1 (Dual Maintenance). *Assuming a (P, z, Q)-HeavyHitter data structure as in Definition 3.1, there exists a data-structure (Algorithm 9) that supports the following operations. Note in the bounds we use $\widetilde{O}$ to hide polynomials in $\log(nP/\|z\|_1)$ in addition to $\log n$ factors, and in our instantiations of the data structure the former factor will be bounded by $\log n$.*

- *Initialize($\mathbf{A} \in \mathbb{R}^{m\times n}, v^{(\text{init})} \in \mathbb{R}^m, w^{(\text{init})} \in [0,1]^m, \epsilon \in [0,1]$) The data-structure pre-processes the given matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, the vector $v^{(\text{init})} \in \mathbb{R}^m$ and accuracy vector $0 < w^{(\text{init})} \leq 1$ in $\widetilde{O}(P)$ time.*
- *SetAccuracy(i, δ): Sets $w_i \leftarrow \delta$ in $\widetilde{O}(z_i)$ amortized time.*
- *Add($h \in \mathbb{R}^n$): Suppose this is the t-th time the Add operations is called, and let $h^{(k)}$ be the vector h given when the Add operation is called for the k^{th} time. Define $v^{(t)} \in \mathbb{R}^m$ to be the vector*

$$v^{(t)} = v^{(\text{init})} + \mathbf{A}\sum_{k=1}^{t} h^{(k)}.$$

Then the data structure returns a vector $\overline{v}^{(t)} \in \mathbb{R}^m$ such that $\|w^{-1}(\overline{v}^{(t)} - v^{(t)})\|_\infty \leq \epsilon$. The output will be in a compact representation to reduce the size. In particular, the data-structure returns a pointer to $\overline{v}$ and a set $I \subset [m]$ of indices i where $\overline{v}_i^{(t)}$ is changed compared to $\overline{v}_i^{(t-1)}$, i.e., the result of the previous call to Add. The amortized time for the t-th call to Add is

$$\widetilde{O}\left(Q + \sqrt{nP/\|z\|_1}\cdot\|(v^{(t)} - v^{(t-1)})/w^{(t)}\|_z^2\epsilon^{-2} + \sqrt{\|z\|_1 P/n}\right).$$

- *ComputeExact(): Returns $v^{(t)} \in \mathbb{R}^m$ in $O(\text{nnz}(\mathbf{A}))$ time, where t is the number of times Add is called so far (i.e., $v^{(t)}$ is the state of the exact vector v after the most recent call to Add).*

A variant of Theorem E.1 was proven in [BLN+20] where $\overline{s} \approx_\epsilon s$ was maintained. Since we need a slightly different type of approximation for $\overline{s}$, we added the SetAccuracy method. This is only small modification and the correctness is proven in Appendix E.

By Line 13 the exact s is defined via

$$s^{(\text{new})} \leftarrow s + \mu\overline{\mathbf{T}}\Phi''(\overline{x})^{1/2}\delta_1 = s + \mathbf{A}h$$

for some vector h. This vector h is exactly the input given to Add of Theorem E.1, so we can use the data structure to maintain an approximation $\overline{s}$ of s.

Line 7 of the IPM (Algorithm 1) requires $g = -\gamma\nabla\Psi(\overline{y})^{\flat(\overline{\tau})}$ which in the next line is multiplied by $\mathbf{A}\Phi''(\overline{x})^{-1/2}$. This product can be obtained by QueryProduct of Theorem D.1.

Line 8 asks us to approximately solve a linear system in $\mathbf{A}^\top\overline{\mathbf{T}}\Phi''(\overline{x})^{-1}\mathbf{A}$ which can be done via the following data structure:

Definition 6.2. *We call a data structure a (P, c, Ψ)-InverseMaintenance, if it supports the following operations:*

- *Initialize($\mathbf{A}, v, \overline{\sigma}$) Initializes in $O(P)$ time for $\overline{\sigma} \geq \frac{1}{2}\sigma(\mathbf{V}^{1/2}\mathbf{A})$ and $\|\overline{\sigma}\|_1 = O(n)$.*
- *Update(i, a, b) Set $v_i \leftarrow a$ and $\overline{\sigma}_i \leftarrow b$ in $O(c_i)$ amortized time.*
- *Solve($\overline{v}, b, \epsilon$) Assume $\overline{\sigma} \geq 1/2\sigma(\mathbf{V}^{1/2}\mathbf{A})$ and the given $\overline{v}$ satisfies $\mathbf{A}^\top\mathbf{V}\mathbf{A} \approx_{1/2} \mathbf{A}^\top\overline{\mathbf{V}}\mathbf{A}$. Then Solve returns $\mathbf{H}^{-1}b$ for $\mathbf{H} \approx_\epsilon \mathbf{A}^\top\overline{\mathbf{V}}\mathbf{A}$ in $O(Q + \text{nnz}(\overline{\mathbf{V}}, \mathbf{A})\log\epsilon^{-1})$ time. Furthermore, for the same $\overline{v}$ and ϵ, the algorithm uses the same $\mathbf{H}$.*

For the complexity bounds one may further assume the following stability assumption: Let $\overline{v}^{(1)}, \overline{v}^{(2)}, \ldots$ be the sequence of inputs given to Solve, then there exists a sequence $\widetilde{v}^{(1)}, \widetilde{v}^{(2)}, \ldots$ such that for all $t > 0$

$$\overline{v}^{(t)} \in (1 \pm 1/(100\log n))\,\widetilde{v}^{(t)} \text{ and } \|(\widetilde{v}^{(t)})^{-1}(\widetilde{v}^{(t)} - \widetilde{v}^{(t+1)})\|_{\overline{\sigma}} = O(1)$$

At last, we are only left with implementing Line 11 of the IPM (Algorithm 1), because all further lines (which update x and s) were already covered when we discussed how to maintain $\overline{x}$ and $\overline{s}$. Line 11 wants us to sample a random diagonal matrix $\mathbf{R}$ according to some distribution that satisfied Definition 4.13. This can be done by assuming the existence of the following data structure:

Definition 6.3. *We call a data structure a (P, c, Q)-HeavySampler data structure if it supports the following operations:*

- *Initialize($\mathbf{A} \in \mathbb{R}^{m \times n}, g \in \mathbb{R}^m_{>0}, \overline{\tau} \in \mathbb{R}^m_{>0}$) Let $\mathbf{A}$ be a matrix with $c_i \geq \text{nnz}(a_i)$. The data structure initializes in $O(P)$ time.*
- *Scale(i, a, b): Sets $g_i \leftarrow a$ and $\overline{\tau}_i \leftarrow b$ in $O(c_i)$ amortized time.*
- *Sample($h \in \mathbb{R}^m$): Returns a random diagonal matrix $\mathbf{R} \in \mathbb{R}^{m \times m}$ that satisfies Definition 4.13 for $\delta_r = \mathbf{G}\mathbf{A}h$ with $\|\delta_r\|_2 \leq m/n$ and $\overline{\tau} \approx_{1/2} \sigma(\overline{\mathbf{A}})$ in $O(Q)$ expected time. Further we have $\mathbb{E}[\text{nnz}(\mathbf{R}\mathbf{A})] = O(Q)$.*

In summary, we can implement all steps of IPM (Algorithm 1) efficiently via the assumed data structures. While so far we only outlined how to use these data structures, Section 6.2 proves in Lemma 6.4 the correctness of these claims in detail. At last, Section 6.3 analyses the resulting complexity in Lemma 6.7. Lemma 6.4 and Lemma 6.7 together form the proof of Theorem 6.1.

6.2 Correctness

We now proceed with the correctness proof of Theorem 6.1 by proving in Lemma 6.4 that Algorithm 4 and Algorithm 5 do indeed implement our IPM.

Lemma 6.4. *Algorithm 5 (ShortStep) and Algorithm 4 (PathFollowing) implement Algorithm 1 (ShortStep) and Algorithm 2 (PathFollowing) respectively.*

Proof. As Algorithm 5 uses many different data structures we will use the notation $D.var$ to refer to variable var of data structure D. For example $D^{(\tau)}.g$ refers to variable g of data structure $D^{(\tau)}$ (which is an instance of Lewis weight data structure Theorem 5.1).

Algorithm 2 consists only of a while loop that calls ShortStep (Algorithm 1). This while loop is also present in Algorithm 4, which calls ShortStep (Algorithm 5). Thus we need to prove that Algorithm 5 implements Algorithm 1.

In addition to the while loop Algorithm 4 initializes data structure such that the following assumptions hold during the first call to ShortStep (Algorithm 5).

Definition 6.5 (Parameter assumptions)**.** *We assume that the following assumptions hold true at the start of the first call to ShortStep (Algorithm 5).*

$$\|\Phi''(\overline{x})^{1/2}(\overline{x} - x)\|_\infty \leq \gamma/2^9, \|\overline{\mu}^{-1}\overline{\tau}^{-1}\Phi''(\overline{x})^{-1/2}(\overline{s} - s)\|_\infty \leq \gamma/2^9, \tag{35}$$

$$\overline{\mu} \approx_{\gamma/2^9} \mu, \overline{\tau} \approx_{\gamma/2^9} \tau(\overline{x}) \tag{36}$$

$$\Delta = \mathbf{A}^\top x - b \tag{37}$$

$$D^{(x,\nabla)}.g = -\gamma\phi''(\overline{x})^{-1/2}, \; D^{(x,\nabla)}.z = \frac{\overline{s} + \overline{\mu\tau}\phi'(\overline{x})}{\overline{\mu\tau}\sqrt{\phi''(\overline{x})}}, \; D^{(x,\nabla)}.w = \phi''(\overline{x})^{-1/2} \tag{38}$$

$$D^{(\text{sample})}.g = \overline{\tau}^{-1}\phi''(\overline{x})^{-1/2}, \; D^{(s)}.w = \overline{\mu\tau}\phi''(\overline{x})^{1/2}, \; D^{(\tau)}.v = \phi''(\overline{x})^{-1/2} \tag{39}$$

$$D^{(-1)}.v = \overline{\tau}^{-1}\phi''(\overline{x})^{-1}, \; D^{(-1)}.\overline{\sigma} = \overline{\tau} \tag{40}$$

We show by induction that these assumptions then also holds true for all subsequent calls to ShortStep.

Before we prove that these assumptions hold true for all subsequent calls to ShortStep (Algorithm 5) we will first prove that Algorithm 5 performs the computations required by the IPM of Algorithm 1, i.e. we show that Algorithm 5 does indeed implement Algorithm 1.

Algorithm 4: Implementation of Algorithm 2

global variables

- $D^{(x,\nabla)}$ instance of primal/gradient maintenance (Theorem D.1) using $\gamma/2^{12}$ accuracy
- $D^{(s)}$ instance of dual maintenance (Theorem E.1) using $\gamma/2^{12}$ accuracy
- $D^{(\tau)}$ instance of Lewis weight data structure (Theorem 5.1) with accuracy $\gamma/2^{12}$
- $D^{(\text{sample})}$ instance of HeavySampler (Definition 6.3)
- $D^{(-1)}$ instance of InverseMaintenance (Definition 6.2)
- $\overline{\tau} \in \mathbb{R}^m$ element-wise approximation of $\tau(\overline{x})$ (multiplicative error)
- $\overline{x} \in \mathbb{R}^m$ element-wise approximation of x (error relative to $\Phi''(\overline{x})$)
- $\overline{s} \in \mathbb{R}^m$ element-wise approximation of s (multiplicative error)
- $\Delta \in \mathbb{R}^n$ (Infeasibility $\Delta = A^\top x - b$)
- $\overline{\mu} \in \mathbb{R}$ approximation of μ
- `/* Parameters where` C `is a sufficiently large constant */`
- $\alpha \leftarrow \frac{1}{4\log(4m/n)}, \varepsilon \leftarrow \frac{\alpha}{C}, \lambda \leftarrow \frac{C\log(Cm/\varepsilon^2)}{\varepsilon}, \gamma \leftarrow \frac{\varepsilon}{C\lambda}, r \leftarrow \frac{\varepsilon\gamma}{C_{\text{norm}}\sqrt{n}}$

procedure PathFollowing($\mathbf{A}, x^{(\text{init})}, s^{(\text{init})}, \mu^{(\text{init})}, \mu^{(\text{end})}$)

- $\overline{x} \leftarrow x^{(\text{init})}$, $\overline{s} \leftarrow s^{(\text{init})}$, $\overline{\mu} \leftarrow \mu^{(\text{init})}$, $\mu \leftarrow \mu^{(\text{init})}$, $\Delta \leftarrow 0$
- Let c be the parameter assumed in Definition 6.2, Definition 3.1, and Definition 6.3, then define $z \leftarrow n/m + nc/\|c\|_1$.
- $\overline{\tau} \leftarrow D^{(\tau)}.\text{Initialize}(\mathbf{A}, \phi''(\overline{x})^{-1/2}, z, 1 - 1/(4\log(4m/n)), 2^{12}C_{4.31}, \gamma/2^{16})$ `//` $C_{4.31}$ `is the constant suppressed by the first item of Lemma 4.31`
- $D^{(x,\nabla)}.\text{Initialize}(\mathbf{A}, x^{(\text{init})}, -\gamma\phi(\overline{x})''^{-1/2}, \overline{\tau}, \frac{\overline{s}+\overline{\mu\tau}\phi'(\overline{x})}{\overline{\mu\tau}\sqrt{\phi''(\overline{x})}}, \phi''(\overline{x})^{-1/2}, \gamma/2^{16})$
- $D^{(s)}.\text{Initialize}(\mathbf{A}, s^{(\text{init})}, \overline{\mu\tau}\phi''(\overline{x})^{1/2}, \gamma/2^{16})$
- $D^{(\text{sample})}.\text{Initialize}(\mathbf{A}, \overline{\tau}^{-1}\phi''(\overline{x})^{-1/2}, \overline{\tau})$
- $D^{(-1)}.\text{Initialize}(\mathbf{A}, \overline{\tau}^{-1}\phi''(\overline{x})^{-1}, \overline{\tau})$
- **while** $\mu > \mu^{(\text{end})}$ **do**
  - ShortStep(μ) (Algorithm 5)
  - $\mu \leftarrow (1-r)\mu$
- **return** $D^{(x,\nabla)}.\text{ComputeExact}()$, $D^{(s)}.\text{ComputeExact}()$

Algorithm 5 implements Algorithm 1 We argue the correctness line by line of Algorithm 1. Line 5 (Algorithm 1) requires that we satisfy Invariant 4.10. This is given by the assumption on $\overline{x}$ in (35) and $\overline{\tau}$ in (36).

Line 6 (Algorithm 1) requires to find a $\overline{y}$ with $\|\overline{y} - y\|_\infty \le \gamma/20$ for

$$y = \frac{s + \mu\tau\phi'(x)}{\mu\tau\sqrt{\phi''(x)}}.$$

We have this $\overline{y}$ implicitly by replacing x, s, μ, and τ in the definition of y by $\overline{x}$, $\overline{s}$, $\overline{\mu}$, and $\overline{\tau}$. We now prove that this $\overline{y}$ satisfies the slightly stronger guarantee $\|\overline{y} - y\|_\infty \le \gamma/40$.

Lemma 6.6 (Approximation of y)**.** *Under the assumptions in Definition 6.5 for* $\overline{y} \stackrel{\text{def}}{=} \frac{\overline{s}+\overline{\mu\tau}\phi'(\overline{x})}{\overline{\mu\tau}\sqrt{\phi''(\overline{x})}}$ *we have that* $\|y - \overline{y}\|_\infty \le \gamma/40$.

Proof. Using the approximations above, we get that

$$\left\| \frac{s + \mu\tau\phi'(x)}{\mu\tau\sqrt{\phi''(x)}} - \frac{\overline{s} + \overline{\mu\tau}\phi'(\overline{x})}{\overline{\mu\tau}\sqrt{\phi''(\overline{x})}} \right\|_\infty$$

Algorithm 5: Implementation of Algorithm 1

global variables

  Same variables as in Algorithm 4.

procedure *ShortStep*($\mu^{(\text{new})} > 0$)

  if $\overline{\mu} \not\approx_{\gamma/2^{12}} \mu^{(\text{new})}$ **then**

    $\overline{\mu} \leftarrow \mu^{(\text{new})}$

    for $i \in [m]$ **do**

      $D^{(x,\nabla)}.\textsc{Update}(i, -\gamma\phi_i''(\overline{x}_i)^{-1/2}, \overline{\tau}_i, (\overline{s}_i + \overline{\mu\tau}_i\phi_i'(\overline{x}_i))/(\overline{\mu\tau}_i\sqrt{\phi_i''(\overline{x}_i)}))$

      $D^{(s)}.\textsc{SetAccuracy}(i, \overline{\mu\tau}_i\phi''(\overline{x}_i)^{1/2})$

  $h' \leftarrow D^{(x,\nabla)}.\textsc{QueryProduct}()$ `//` $h' = -\gamma\mathbf{A}^\top\Phi''(\overline{x})^{-1/2}\nabla\Phi(\overline{y})^{\flat(\overline{\tau})}$

  `/* Leverage score sampling gives` $\mathbf{H} \overset{\text{def}}{=} \mathbf{A}^\top\overline{\mathbf{V}}\mathbf{A} \approx_{\gamma/2} \mathbf{A}^\top\overline{\mathbf{T}}^{-1}\Phi''(\overline{x})^{-1}\mathbf{A}$ `*/`

  $\overline{v}_i \leftarrow \begin{cases} \frac{1}{\min(1, 100\overline{\tau}\log(n)/\gamma^2)} & \text{with probability } \min(1, 100\overline{\tau}\log(n)/\gamma^2) \\ 0 & \text{otherwise} \end{cases}$.

  `/*` $h'' = \mathbf{H}^{-1}(h' + (\mathbf{A}^\top x - b))$, $\delta_r = \overline{\mathbf{T}}^{-1}\Phi''(\overline{x})^{-1/2}\mathbf{A}h''$ `*/`

  $h'' \leftarrow D^{(-1)}.\textsc{Solve}(\overline{v}, h' + \Delta, \gamma/2)$

  $\mathbf{R} \leftarrow D^{(\text{sample})}.\textsc{Sample}(h'')$

  $x^{(\text{tmp})}, I_x \leftarrow D^{(x,\nabla)}.\textsc{QuerySum}(-\mathbf{R}\overline{\mathbf{T}}^{-1}\Phi''(\overline{x})^{-1}\mathbf{A}h'')$

  $\Delta \leftarrow \Delta + h' - \mathbf{A}^\top\mathbf{R}\overline{\mathbf{T}}^{-1}\Phi''(\overline{x})^{-1}\mathbf{A}h''$ `//` $\texttt{Maintain } \Delta = \mathbf{A}^\top x - b$

  for $i \in I_x$ **do**

    if $|\sqrt{\phi_i''(x_i^{(\text{tmp})})}(x_i^{(\text{tmp})} - \overline{x}_i)| > \gamma/2^{12}$ **then**

      $\overline{x}_i \leftarrow x_i^{(\text{tmp})}$

      $D^{(x,\nabla)}.\textsc{Update}(i, -\gamma\phi_i''(\overline{x}_i)^{-1/2}, \overline{\tau}_i, (\overline{s}_i + \overline{\mu\tau}_i\phi_i'(\overline{x}_i))/(\overline{\mu\tau}_i\sqrt{\phi_i''(\overline{x}_i)}))$

      $D^{(x,\nabla)}.\textsc{SetAccuracy}(i, \phi''(\overline{x}_i)^{-1/2})$

      $D^{(\tau)}.\textsc{Scale}(i, \phi''(\overline{x}_i)^{-1/2})$

      $D^{(\text{sample})}.\textsc{Scale}(i, \overline{\tau}_i^{-1}\phi''(\overline{x}_i)^{-1/2})$

      $D^{(-1)}.\textsc{Update}(i, \overline{\tau}_i^{-1}\phi''(\overline{x}_i)^{-1}, \overline{\tau}_i)$

      $D^{(s)}.\textsc{SetAccuracy}(i, \overline{\mu\tau}_i\phi''(\overline{x}_i)^{1/2})$

  $\tau^{(\text{tmp})}, I_\tau \leftarrow D^{(\tau)}.\textsc{Query}()$

  for $i \in I_\tau$ **do**

    if $\tau_i^{(\text{tmp})} \not\approx_{\gamma/2^{10}} \overline{\tau}_i$ **then**

      $\overline{\tau}_i \leftarrow \tau_i^{(\text{tmp})}$

      $D^{(x,\nabla)}.\textsc{Update}(i, -\gamma\phi_i''(\overline{x}_i)^{-1/2}, \overline{\tau}_i, (\overline{s}_i + \overline{\mu\tau}_i\phi_i'(\overline{x}_i))/(\overline{\mu\tau}_i\sqrt{\phi_i''(\overline{x}_i)}))$

      $D^{(\text{sample})}.\textsc{Scale}(i, \overline{\tau}_i^{-1}\phi''(\overline{x}_i)^{-1/2})$

      $D^{(-1)}.\textsc{Update}(i, \overline{\tau}_i^{-1}\phi''(\overline{x}_i)^{-1}, \overline{\tau}_i)$

      $D^{(s)}.\textsc{SetAccuracy}(i, \overline{\mu\tau}_i\phi''(\overline{x}_i)^{1/2})$

  $s^{(\text{tmp})}, I_s \leftarrow D^{(s)}.\textsc{Add}(\mu D^{(-1)}.\textsc{Solve}(\overline{v}, h', \gamma/2))$

  for $i \in I_s$ **do**

    if $|\overline{\mu}^{-1}\overline{\tau}^{-1}\Phi''(\overline{x})^{-1/2}(s_i^{(\text{tmp})} - \overline{s}_i)| > \gamma/2^{10}$ **then**

      $\overline{s}_i \leftarrow s_i^{(\text{tmp})}$

      $D^{(x,\nabla)}.\textsc{Update}(i, -\gamma\phi_i''(\overline{x}_i)^{-1/2}, (\overline{s}_i + \overline{\mu\tau}_i\phi_i'(\overline{x}_i))/(\overline{\mu\tau}_i\sqrt{\phi_i''(\overline{x}_i)}))$

$$
\begin{aligned}
&\leq \left\| \frac{s-\overline{s}}{\overline{\mu\tau}\sqrt{\phi''(\overline{x})}} \right\|_\infty + \left\| \frac{s+\mu\tau\phi'(x)}{\mu\tau\sqrt{\phi''(x)}} - \frac{s+\overline{\mu\tau}\phi'(\overline{x})}{\overline{\mu\tau}\sqrt{\phi''(\overline{x})}} \right\|_\infty \\
&\leq \gamma/2^9 + \left\| \frac{s+\mu\tau\phi'(x)}{\mu\tau\sqrt{\phi''(x)}} - \frac{s+\mu\overline{\tau}\phi'(\overline{x})}{\overline{\mu\tau}\sqrt{\phi''(\overline{x})}} \right\|_\infty + \left\| \frac{(\mu-\overline{\mu})\phi'(\overline{x})}{\overline{\mu}\sqrt{\phi''(\overline{x})}} \right\| \\
&\leq \gamma/2^8 + \left\| \frac{s+\mu\tau\phi'(x)}{\mu\tau\sqrt{\phi''(x)}} - \frac{s+\mu\overline{\tau}\phi'(\overline{x})}{\overline{\mu\tau}\sqrt{\phi''(\overline{x})}} \right\|_\infty,
\end{aligned}
$$

where we used 1-self-concordance, specifically that $|\phi'(\overline{x})| \leq \sqrt{\phi''(\overline{x})}$ in the last step. Now, we calculate the errors resulting from $\overline{\tau}$ and $\phi'(\overline{x})$, which gives that

$$
\begin{aligned}
&\left\| \frac{s+\mu\tau\phi'(x)}{\mu\tau\sqrt{\phi''(x)}} - \frac{s+\mu\overline{\tau}\phi'(\overline{x})}{\overline{\mu\tau}\sqrt{\phi''(\overline{x})}} \right\|_\infty \leq \left\| \frac{s+\mu\tau\phi'(x)}{\mu\tau\sqrt{\phi''(x)}} - \frac{s+\mu\tau\phi'(\overline{x})}{\overline{\mu\tau}\sqrt{\phi''(\overline{x})}} \right\|_\infty + \left\| \frac{\mu(\tau-\overline{\tau})\phi'(\overline{x})}{\overline{\mu\tau}\sqrt{\phi''(\overline{x})}} \right\|_\infty \\
&\leq 1.1\gamma/2^9 + \left\| \frac{s+\mu\tau\phi'(x)}{\mu\tau\sqrt{\phi''(x)}} - \frac{s+\mu\tau\phi'(x)}{\overline{\mu\tau}\sqrt{\phi''(\overline{x})}} \right\|_\infty + \left\| \frac{\mu\tau(\phi'(\overline{x})-\phi'(x))}{\overline{\mu\tau}\sqrt{\phi''(\overline{x})}} \right\|_\infty \\
&\leq 2.2\gamma/2^9 + \left\| \frac{s+\mu\tau\phi'(x)}{\mu\tau\sqrt{\phi''(x)}} - \frac{s+\mu\tau\phi'(x)}{\overline{\mu\tau}\sqrt{\phi''(\overline{x})}} \right\|_\infty
\end{aligned}
$$

where we used 1-self-concordance in the last step. Note that $\phi''(x)^{1/2} \approx_{1.1\gamma/2^9} \phi''(\overline{x})^{1/2}$ by self-concordance (Lemma 4.16), hence $\mu\tau\sqrt{\phi''(x)} \approx_{\gamma/2^7} \overline{\mu\tau}\sqrt{\phi''(\overline{x})}$. This gives us

$$
\left\| \frac{s+\mu\tau\phi'(x)}{\mu\tau\sqrt{\phi''(x)}} - \frac{s+\mu\tau\phi'(x)}{\overline{\mu\tau}\sqrt{\phi''(\overline{x})}} \right\|_\infty \leq \left\| \frac{s+\mu\tau\phi'(x)}{\mu\tau\sqrt{\phi''(x)}} \right\|_\infty \left\| 1 - \frac{\mu\tau\sqrt{\phi''(x)}}{\overline{\mu\tau}\sqrt{\phi''(\overline{x})}} \right\|_\infty \leq \varepsilon\gamma/2^6 \leq \gamma/2^9
$$

because (x, s, μ) is ε-centered. Combining everything, we have that the total error is $\gamma/2^8 + 2.2\gamma/2^9 + \gamma/2^9 \leq \gamma/40$. □

Line 7 (Algorithm 1) asks us to compute

$$
g \leftarrow -\gamma\nabla\Phi(\overline{z})^{\flat(\overline{\tau})}
$$

for $\|\overline{z}-y\|_\infty \leq \gamma/20$. While we do not compute g, our implementation does compute $\mathbf{A}^\top\Phi''(\overline{x})^{-1/2}g$ as follows: Line 9 of Algorithm 5 computes

$$
h' = -\gamma\mathbf{A}^\top\Phi''(\overline{x})^{-1/2}\nabla\Phi(\overline{z})^{\flat(\overline{\tau})} = \mathbf{A}^\top\Phi''(\overline{x})^{-1/2}g
$$

for some $\overline{z} \approx_{\gamma/2^{10}} D^{(x,\nabla)}.z = \overline{y} \approx_{\gamma/40} y$ by the guarantees of the primal/gradient data structure Theorem D.1 and assumption (38).

By Line 8 of Algorithm 1 we must obtain a matrix $\mathbf{H} \approx_\epsilon \mathbf{A}^\top\overline{\mathbf{T}}^{-1}\Phi''(\overline{x})^{-1}\mathbf{A}$ which is done in Line 10 of Algorithm 5 with higher accuracy $\mathbf{H} \approx_{\gamma/2} \mathbf{A}^\top\overline{\mathbf{T}}^{-1}\Phi''(\overline{x})^{-1}\mathbf{A}$. Next, since $D^{(-1)}.v = \overline{\tau}^{-1}\phi''(\overline{x})^{-1}$ and $D^{(-1)}.\overline{\sigma} = \overline{\tau}$ by (40) we have $\overline{\sigma} \geq \frac{1}{2}\sigma((\overline{\mathbf{T}}^{-1}\Phi''(\overline{x})^{-1})^{1/2}\mathbf{A})$ and a call to $D^{(-1)}.\textsc{Solve}(\cdot,\cdot,\gamma/2)$ (e.g. in Line 11) is equivalent to multiplying by some matrix $\mathbf{H}^{-1}$ with

$$
\mathbf{H}^{-1} \approx_\gamma (\mathbf{A}^\top\overline{\mathbf{T}}^{-1}\Phi''(\overline{x})^{-1}\mathbf{A})^{-1}
$$

which with $\gamma \leq \epsilon$ is accurate enough.

Line 10 of Algorithm 1 wants us to compute

$$
\delta_r = \overline{\mathbf{T}}^{-1}\Phi''(\overline{x})^{-1/2}\mathbf{A}\mathbf{H}^{-1}\mathbf{A}^\top(\Phi''(\overline{x})^{-1/2}g + \mathbf{A}^\top x - b)
$$

This is done implicitly in Line 11 of our implementation Algorithm 5. By assumption (37) we have $\Delta = \mathbf{A}^\top x - b$, thus Line 11 computes h'' with

$$
h'' = \mathbf{H}^{-1}(h' + \mathbf{A}^\top x - b).
$$

The vector δ_r can be represented via h'' by

$$\delta_r = \overline{\mathbf{T}}^{-1}\Phi''(\overline{x})^{-1/2}\mathbf{A}\mathbf{H}^{-1}(h' + \mathbf{A}^\top x - b) = \overline{\mathbf{T}}^{-1}\Phi''(\overline{x})^{-1/2}\mathbf{A}h''$$

because of $h' = \mathbf{A}^\top\Phi''(\overline{x})^{-1}g$.

Line 11 of Algorithm 1 wants us to compute a random diagonal matrix $\mathbf{R}$ that satisfies the conditions of Definition 4.13. By Definition 6.3 (HeavySampler), assumption (39) on $D^{(\text{sample})}$, and assumption (36) on $\overline{\tau}$, such a random matrix can be obtained via $D^{(\text{sample})}.\textsc{Sample}(h'')$. Note that for $\delta_r := D^{(\text{sample})}.\mathbf{G}\mathbf{A}h''$ we have by Corollary 4.30 (item 2, the bound on δ_r) and $\tau \geq n/m$ that

$$\|\delta_r\|_2^2 \leq \frac{m}{n}\|\delta_r\|_\tau^2 \leq \frac{m}{n}.$$

So the requirements for the Sample procedure stated in Definition 6.3 are satisfied.

Line 12 and Line 14 of Algorithm 1 want us to compute

$$x^{(\text{new})} \leftarrow x + \Phi''(\overline{x})^{-1/2}(g - \mathbf{R}\delta_r).$$

By guarantees of the primal/gradient maintenance (Theorem D.1) and assumption (38), Line 13 of our implementation Algorithm 5 computes $x^{(\text{tmp})}$ with

$$\|\Phi''(\overline{x})^{1/2}(x^{(\text{new})} - x^{(\text{tmp})})\|_\infty \leq \gamma/2^{12} \tag{41}$$

Our implementation also computes an $\tau^{(\text{tmp})}$ in Line 24. By Line 20 our implementation makes sure that $D^{(\tau)}.g = \phi''(\overline{x})^{-1/2}$. Thus the vector $\tau^{(\text{tmp})}$ in Line 24 satisfies $\tau^{(\text{tmp})} \approx_{\gamma/2^{10}} \tau(\overline{x})$ for the new $\overline{x}$ and after Line 25 we have

$$\overline{\tau} \approx_{\gamma/2^{10}} \tau(\overline{x}) \tag{42}$$

for the new $\overline{x}$.

Line 13 and Line 14 of Algorithm 1 asks us to compute

$$s^{(\text{new})} \leftarrow s + \mu\mathbf{A}\mathbf{H}^{-1}\mathbf{A}^\top\Phi''(\overline{x})^{-1/2}g$$

By dual maintenance Theorem E.1 and $D^{(s)}.w = \overline{\mu\tau}\Phi''(\overline{x})^{-1/2}$ by (39), Line 32 of our implementation Algorithm 5 computes $s^{(\text{tmp})}$ with

$$\|\overline{\mu}^{-1}\overline{\tau}^{-1}\Phi''(\overline{x})^{-1/2}(s^{(\text{new})} - s^{(\text{tmp})})\|_\infty \leq \gamma/2^{12} \tag{43}$$

Note that $\overline{x}$ and $\overline{\tau}$ in (43) refer to the new values of $\overline{x}$ and $\overline{\tau}$ as they were changed in Line 15 and Line 25.

Assumptions on $\overline{x}$, $\overline{s}$: To argue assumption (35) on $\overline{x}$, define $\overline{x}$ to be the value of $\overline{x}$ at the start of ShortStep and $\overline{x}^{(\text{new})}$ to be the new value at the end of ShortStep. We have $\|\Phi''(\overline{x})^{1/2}(x^{(\text{new})} - x^{(\text{tmp})})\|_\infty \leq \gamma/2^{10}$ by (41). If $\overline{x}_i = \overline{x}_i^{(\text{new})}$, then by Line 15 we have $|\phi''(\overline{x}_i)^{1/2}(x_i^{(\text{tmp})} - \overline{x}_i)| \leq \gamma/2^{12}$. Thus

$$\begin{aligned}|\phi''(\overline{x}_i^{(\text{new})})^{1/2}(x_i^{(\text{new})} - \overline{x}_i^{(\text{new})})| &= |\phi''(\overline{x}_i)^{1/2}(x_i^{(\text{new})} - \overline{x}_i)| \\ \leq |\phi''(\overline{x}_i)^{1/2}(x_i^{(\text{tmp})} - \overline{x}_i)| + |\phi''(\overline{x}_i)^{1/2}(x^{(\text{new})} - x_i^{(\text{tmp})})| &\leq \gamma/2^{10} + \gamma/2^{12} \leq \gamma/2^9.\end{aligned}$$

On the other hand, if $\overline{x}_i \neq \overline{x}_i^{(\text{new})}$, then $\overline{x}_i^{(\text{new})} = x_i^{(\text{tmp})}$, so

$$|\phi''(\overline{x}_i^{(\text{new})})^{1/2}(x_i^{(\text{new})} - \overline{x}_i^{(\text{new})})| \leq 1.1|\phi''(\overline{x}_i)^{1/2}(x_i^{(\text{new})} - x_i^{(\text{tmp})})| \leq \gamma/2^9$$

by self-concordance (Lemma 4.16) and $\overline{x}_i^{(\text{new})} \approx_\gamma \overline{x}_i$. In summary, we have $\|\Phi(\overline{x}^{(\text{new})})^{1/2}(\overline{x}^{(\text{new})} - x)\|_\infty \leq \gamma/2^9$, so at the start of the next call of ShortStep we satisfy the assumption on $\overline{x}$ in (35) again.

For the assumption in (35) on $\overline{s}$, note that we have

$$\|\overline{\mu}^{-1}(\overline{\tau}^{(\text{new})})^{-1}\Phi''(\overline{x}^{(\text{new})})^{-1/2}(s^{(\text{tmp})}-s)\|_\infty \leq \gamma/2^{12}$$

by (43). So if $\overline{s}_i^{(\text{new})} = s_i^{(\text{tmp})}$ then

$$|\overline{\mu}^{-1}(\overline{\tau}_i^{(\text{new})})^{-1}\phi''(\overline{x}_i^{(\text{new})})^{-1/2}(s_i^{(\text{new})}-\overline{s}_i^{(\text{new})})| \leq \gamma/2^{12}.$$

Alternatively, if $\overline{s}_i^{(\text{new})} \neq s_i^{(\text{tmp})}$, then $\overline{s}_i^{(\text{new})} = \overline{s}_i$ and by the condition in Line 33 we have

$$\begin{aligned}
&|\overline{\mu}^{-1}(\overline{\tau}_i^{(\text{new})})^{-1}\phi''(\overline{x}_i^{(\text{new})})^{-1/2}(s_i^{(\text{new})}-\overline{s}_i^{(\text{new})})| \\
&\leq |\overline{\mu}^{-1}(\overline{\tau}_i^{(\text{new})})^{-1}\phi''(\overline{x}_i^{(\text{new})})^{-1/2}(s_i^{(\text{new})}-s_i^{(\text{tmp})})| + |\overline{\mu}^{-1}(\overline{\tau}_i^{(\text{new})})^{-1}\phi''(\overline{x}_i^{(\text{new})})^{-1/2}(s_i^{(\text{tmp})}-\overline{s}_i^{(\text{new})})| \\
&\leq \gamma/2^{12} + \gamma/2^{10} \leq \gamma/2^9
\end{aligned}$$

Since $\overline{\mu}$ changes by at most an $\exp(\gamma/2^{12})$ factor at the start of ShortStep, assumption (35) will be satisfied during the next call to ShortStep.

Assumptions on $\overline{\tau}$ and $\overline{\mu}$ We already argued in (42) that $\overline{\tau} \approx_{\gamma/2^{10}} \tau(\overline{x})$. Further $\mu \approx_{\gamma/2^{12}} \overline{\mu}$ is verified at the start of each call to ShortStep in Line 4. Thus the assumptions of (36) are satisfied.

Assumption on D^{-1}, $D^{(\tau)}$, $D^{(\text{sample})}$, and $D^{(x,\nabla)}$: All data structures that depend on $\overline{\mu}$, $\overline{x}$, $\overline{s}$, or $\overline{\tau}$ are updated, whenever $\overline{\mu}$ or an entry of $\overline{x}$, $\overline{s}$, or $\overline{\tau}$ changes (see Line 4, Line 15, Line 25, and Line 33). So the assumptions in (38), (39), and (40) are always satisfied.

Assumption $\Delta = \mathbf{A}^\top x - b$: $\Delta = \mathbf{A}^\top x - b$ initially because the input x is feasible. Then after Line 14 we have $\Delta = \mathbf{A}^\top x^{(\text{new})} - b$ for $x^{(\text{new})} = x + \Phi''(\overline{x})^{-1/2}g - \mathbf{R}\overline{\mathbf{T}}^{-1}\Phi''(\overline{x})^{-1/2}\mathbf{A}h''$. Thus we always maintain $\mathbf{A}^\top x - b$, whenever x changes.

□

6.3 Complexity

We now analyze in Lemma 6.7 the complexity of Algorithm 4 which together with Lemma 6.4 concludes the proof of Theorem 6.1.

Lemma 6.7. *Assume (P, c, Q) heavy hitter, (P, c, Q) inverse maintenance, and (P, c, Q) heavy sampler. Then the total time of PathFollowing can be bounded by*

$$\widetilde{O}\left(\left(\sqrt{P\|c\|_1} + \sqrt{n}\left(Q + n \cdot \max_i \text{nnz}(a_i)\right)\right)\log\frac{\mu^{(\text{init})}}{\mu^{(\text{end})}}\right)$$

Proof. We analyze the complexity of PathFollowing in multiple parts: First we analyze the initialization of all data structures, i.e. the time spent until the first call of ShortStep. Then we analyze the total time spent on all calls to ShortStep.

Initialization Initializing $D^{(x,\nabla)}$ takes $\widetilde{O}(\text{nnz}(\mathbf{A})) = \widetilde{O}(P)$ time by Theorem D.1 and $\text{nnz}(\mathbf{A}) \leq P$ (Definition 3.1). The initialization of $D^{(s)}$, $D^{(\text{sample})}$, and $D^{(-1)}$ take $\widetilde{O}(P)$ time each by Theorem E.1, Definition 6.3 and Definition 6.2.

The initialization of $D^{(\tau)}$ takes $\widetilde{O}(P+Q+n(\max_i \text{nnz}(a_i))+\sqrt{P\|c\|_1/n})$ time by Theorem 5.2, because we can solve any linear system of the form $\mathbf{A}^\top\mathbf{V}\mathbf{A}x = b$ by initializing an instance of Definition 6.2 for that w and then solving the system via D.Solve.

ShortStep Let T be the number of calls to ShortStep (Algorithm 5). We start by bounding how often $\overline{\mu}$, $\overline{x}$, $\overline{s}$, and $\overline{\tau}$ are modified. Let $\overline{\mu}^{(t)}$, $\overline{x}^{(t)}$, $\overline{s}^{(t)}$, $\overline{\tau}^{(t)}$ refer to the respective variables during iteration number t. We will prove the following bounds.

$$\sum_{t=2}^{T} \mathbf{1}_{\overline{\mu}^{(t)} \neq \overline{\mu}^{(t-1)}} \leq \widetilde{O}(T/\sqrt{n}) \tag{44}$$

$$\sum_{t=2}^{T}\sum_{i=1}^{m} \tau(\overline{x}^{(t)})_i \mathbf{1}_{\overline{x}_i^{(t)} \neq \overline{x}_i^{(t-1)}} \leq \widetilde{O}(T^2) \tag{45}$$

$$\sum_{t=2}^{T}\sum_{i=1}^{m} \tau(\overline{x}^{(t)})_i \mathbf{1}_{\overline{s}_i^{(t)} \neq \overline{s}_i^{(t-1)}} \leq \widetilde{O}(T^2) \tag{46}$$

$$\sum_{t=2}^{T}\sum_{i=1}^{m} \tau(\overline{x}^{(t)})_i \mathbf{1}_{\overline{\tau}_i^{(t)} \neq \overline{\tau}_i^{(t-1)}} \leq \widetilde{O}(T^2) \tag{47}$$

The bound on $\overline{\mu}$ follows because μ changes by an $(1-r)$ factor for $r = \widetilde{O}(1/\sqrt{n})$ in each iteration, and we update $\overline{\mu}$ whenever $\mu \not\approx_{\gamma/2^{12}} \overline{\mu}$. Thus is takes $\widetilde{\Omega}(\sqrt{n})$ iterations until we have to change $\overline{\mu}$ which results in the $\widetilde{O}(T/\sqrt{n})$ bound.

For the bound on $\overline{x}$ note that we update $\overline{x}_i \leftarrow x_i^{(\mathrm{tmp})}$ whenever $|\phi''(x_i^{(\mathrm{tmp})})^{1/2}(x_i^{(\mathrm{tmp})} - \overline{x}_i)| > \gamma/2^{12}$. Let t_1, t_2 be two iterations where we update $\overline{x}_i$, and let $\widehat{x}^{(1)}, \widehat{x}^{(2)}, \ldots$ be the sequence from Lemma 4.44 for $\beta = \gamma/2^{15}$. Then we have

$$\begin{aligned}
|\phi''((x^{(\mathrm{tmp})})_i^{(t_2)})^{1/2}(\widehat{x}^{(t_1)} - \widehat{x}^{(t_2)})_i| &\geq |\phi''((x^{(\mathrm{tmp})})_i^{(t_2)})^{1/2}(x^{(t_1)} - x^{(t_2)})_i| - \gamma/2^{14} \\
&\geq |\phi''((x^{(\mathrm{tmp})})_i^{(t_2)})^{1/2}((x^{(\mathrm{tmp})})^{(t_1)} - (x^{(\mathrm{tmp})})^{(t_2)})_i| - \gamma/2^{13} \\
&\geq \gamma/2^{12}
\end{aligned}$$

where we used $\|\Phi''(\widehat{x})^{1/2}(\widehat{x} - x)\|_\infty \leq \gamma/2^{15}$, $\|\Phi''(x^{(\mathrm{tmp})})^{1/2}(x^{(\mathrm{tmp})} - x)\|_\infty \leq \gamma/2^{16}$, and the fact that we only update when $x_i^{(\mathrm{tmp})}$ changed by at least $\gamma/2^{12}$.

Thus we can bound

$$\begin{aligned}
\sum_{t=2}^{T}\sum_{i=1}^{m} \tau(\overline{x}^{(t)})_i \mathbf{1}_{\overline{x}_i^{(t)} \neq \overline{x}_i^{(t-1)}} &= \widetilde{O}\left(\sum_{t=2}^{T}\sum_{i=1}^{m} \tau(\overline{x}^{(t)})_i \left(\phi''(\widehat{x}_i^{(t-1)})^{1/2}|\widehat{x}_i^{(t)} - \widehat{x}_i^{(t-1)}|\right)^2\right) \\
&= \widetilde{O}\left(\left(\sum_{t=2}^{T} \|\Phi''(\widehat{x}^{(t)})^{1/2}(\widehat{x}^{(t)} - \widehat{x}^{(t-1)})\|_{\tau(\widehat{x}^{(t-1)})}\right)^2\right) = \widetilde{O}\left(T^2\right).
\end{aligned}$$

In a similar way we can bound

$$\begin{aligned}
\sum_{t=2}^{T}\sum_{i=1}^{m} \tau(\overline{x}^{(t)})_i \mathbf{1}_{\overline{s}_i^{(t)} \neq \overline{s}_i^{(t-1)}} &= \widetilde{O}\left(\sum_{t=2}^{T}\sum_{i=1}^{m} \tau(\overline{x}^{(t)})_i \left((\mu^{-1}\tau(\overline{x}^{(t)})_i \phi''(\overline{x}_i^{(t)})^{1/2})^{-1}|s_i^{(t)} - s_i^{(t-1)}|\right)^2\right) \\
&= \widetilde{O}\left(\left(\sum_{t=2}^{T} \|(\mu^{-1}\tau(\overline{x}^{(t)})\Phi''(\overline{x}^{(t)})^{1/2})^{-1}(s^{(t)} - s^{(t-1)})\|_{\tau(\overline{x}^{(t-1)})}\right)^2\right) \\
&= \widetilde{O}\left(T^2\right)
\end{aligned}$$

by using $s^{(t)} - s^{(t-1)} = \delta_s$ and Corollary 4.30 Part 1. At last, consider the bound on $\overline{\tau}$, where we have

$$\sum_{t=2}^{T}\sum_{i=1}^{m} \tau(\overline{x}^{(t)})_i \mathbf{1}_{\overline{\tau}_i^{(t)} \neq \overline{\tau}_i^{(t-1)}} = \widetilde{O}\left(\sum_{t=2}^{T}\sum_{i=1}^{m} \tau(\widehat{x}^{(t)})_i \left(\tau(\widehat{x}^{(t)})_i^{-1}|\tau(\widehat{x}^{(t)})_i - \tau(\widehat{x}^{(t-1)})_i|\right)^2\right)$$

$$= \widetilde{O}\left(\left(\sum_{t=2}^{T} \|\tau(\widehat{x}^{(t)})^{-1}(\tau(\widehat{x}^{(t)}) - \tau(\widehat{x}^{(t-1)}))\|_{\tau(\widehat{x}^{(t-1)})}\right)^2\right) = \widetilde{O}\left(T^2\right)$$

by Lemma 4.45.

Cost of $D^{(x,\nabla)}$, $D^{(s)}$, $D^{(\text{sample})}$ The time spent per call to $D^{(s)}.\textsc{Scale}(i,\cdot)$, $D^{(s)}.\textsc{SetAccuracy}(i,\cdot)$, $D^{(x,\nabla)}.\textsc{Update}(i,\cdot)$, and $D^{(\text{sample})}.\textsc{Scale}(i,\cdot)$ is bounded by $\widetilde{O}(c_i) = \widetilde{O}(\tau_i \cdot \|c\|_1/n)$, because of $\tau \geq nc/\|c\|_1$. These functions are called whenever $\overline{\tau}_i$, $\overline{x}_i$, $\overline{s}_i$, or $\overline{\mu}$ are changed, so by the previous bounds on (44)-(47) we can bound the total time spent on calling these functions by $\widetilde{O}(T^2\|c\|_1/n + \|c\|_1 T/\sqrt{n})$.

In each iteration we call $D^{(s)}.\textsc{Add}$ and the total time for these calls is bounded by

$$\begin{aligned}
&\widetilde{O}(TQ + T\sqrt{nP/\|c\|_1}\|\overline{\delta}_s/(\mu\tau\phi''(x)^{1/2})\|_c^2 + T\sqrt{\|c\|_1 P/n}) \\
&= \widetilde{O}(TQ + T\sqrt{nP/\|c\|_1}(\|c\|_1/n)\|\overline{\delta}_s/(\mu\tau\phi''(x)^{1/2})\|_\tau^2 + T\sqrt{\|c\|_1 P/n}) \\
&= \widetilde{O}(T(Q + \sqrt{P\|c\|_1/n}))
\end{aligned}$$

by Theorem E.1, Corollary 4.30 Part 1, and $\tau \geq nc/\|c\|_1$.

Likewise, we can bound the time spent on all calls to $D^{(x,\nabla)}.\textsc{QueryProduct}$ and $D^{(x,\nabla)}.\textsc{QuerySum}$ by

$$\begin{aligned}
&\widetilde{O}(Tn + T(\max_i \text{nnz}(a_i)) + \mathbb{E}[\|\mathbf{R}\mathbf{A}h''\|_0] + T\sum_{t=1}^{T} \|\phi''(\overline{x}^{(t)})^{-1/2}g^{(t)}/\phi''(\overline{x}^{(t)})^{-1/2}\|_2^2) \\
&= \widetilde{O}(Tn + T(\max_i \text{nnz}(a_i)) + \mathbb{E}[\text{nnz}(\mathbf{R}\mathbf{A})] + (Tm/n)\sum_{t=1}^{T} \|g^{(t)}\|_{\tau(\overline{x}^{(t)})}) \\
&= \widetilde{O}(T(n + (\max_i \text{nnz}(a_i)) + \mathbb{E}[\text{nnz}(\mathbf{R}\mathbf{A})]) + T^2 m/n)
\end{aligned}$$

where $\mathbb{E}[\text{nnz}(\mathbf{R}\mathbf{A})]$ is the expected number of entries returned by Definition 6.3 and we used that g by definition has $\|g\|_{\tau+\infty} \leq 1$ and that $\tau \geq n/m$.

We can bound the expected time of a call to $D^{(\text{sample})}.\textsc{Sample}$ and the expected size of $\mathbb{E}[\text{nnz}(\mathbf{R}\mathbf{A})]$ by Q using Definition 6.3.

Cost of $D^{(\tau)}$ The complexity bounds of $D^{(\tau)}$ stated in Theorem 5.1 only holds, if Condition 1 and Condition 2 are satisfied. The first condition requires us to be able to solve linear systems in $(\mathbf{A}^\top\overline{\mathbf{V}}\mathbf{A})^{-1}$ for $\mathbf{A}^\top\overline{\mathbf{V}}\mathbf{A} \approx \mathbf{A}^\top\overline{\mathbf{T}}^{1-2/p}\Phi''(\overline{x})^{-1}\mathbf{A}$ in $\widetilde{O}(Q + \text{nnz}(\overline{\mathbf{V}}\mathbf{A}))$ time. Such a solver by D^{-1} (Definition 6.2).

Next, we require the existence of a sequence $\widehat{x}^{(1)}, \widehat{x}^{(2)}, \ldots$ where for all $t \in [T]$ we have

$$\phi''(\overline{x}^{(t)})^{-1/2} \in (1 \pm 1/(10^5 \log n))\phi''(\widehat{x}^{(t)})^{-1/2}$$

$$\|\tau(\widehat{x}^{(t)})^{\frac{1}{p}-\frac{1}{2}}\phi''(\widehat{x}^{(t)})^{-\frac{1}{2}}(\tau(\widehat{x}^{(t)})^{\frac{1}{2}-\frac{1}{p}}\phi''(\widehat{x}^{(t)})^{-\frac{1}{2}} - \tau(\widehat{x}^{(t+1)})^{\frac{1}{2}-\frac{1}{p}}\phi''(\widehat{x}^{(t+1)})^{-\frac{1}{2}})\|_{\tau(\widehat{x}^{(t)})} = O(1).$$

for $p = 1 - 1/(4\log(4m/n))$. This sequence is given by Lemma 4.44 and Lemma 4.45. Thus in summary, the complexity bounds stated in Theorem 5.1 apply.

We can now bound the total cost of all calls to $D^{(\tau)}.\textsc{Scale}(i,\cdot)$ by $\widetilde{O}(T^2\|c\|_1/n)$. This is because on such function call has amortized cost $\widetilde{O}(\frac{\|c\|_1}{n}\tau_i)$ and the function is called whenever $\overline{x}_i$ changes. By the bound on (45) we then obtain the total cost over T iterations of $\textsc{ShortStep}$.

At last, the time requires for all calls to $D^{(\tau)}.\textsc{Query}$ is bounded by

$$\widetilde{O}\left(T(Q + n(\max_i \text{nnz}(a_i)) + \sqrt{P\|c\|_1/n})\right).$$

Cost of $D^{(-1)}$ The stability assumption stated in Definition 6.2 is given by Lemma 4.44 and Lemma 4.45, so we can use the complexities stated in Definition 6.2.

A call to $D^{(-1)}.\textsc{Update}(i,\cdot)$ takes $O(c_i)$ amortized time and occurs whenever $\overline{\tau}_i$ or $\overline{x}_i$ changes. By (47) and (45) we can thus bound the cost by $\widetilde{O}(T^2\|c\|_1/n)$.

Performing the calls to $D^{(-1)}.\textsc{Solve}(\overline{v}, h'+\Delta, \gamma/2)$ and $D^{(-1)}.\textsc{Solve}(\overline{v}, h', \gamma/2)$ takes $O(Q+n(\max_i \operatorname{nnz}(a_i)))$ time, which is subsumed by the cost of $D^{(\tau)}.\textsc{Query}$.

Total Cost When calling ShortStep a total of T times, the total cost is

$$\begin{aligned}&\widetilde{O}\left((T(Q+n(\max_i \operatorname{nnz}(a_i))+\sqrt{P\|c\|_1/n}+\|c\|_1/\sqrt{n})+T^2\|c\|_1/n+T^2m/n\right)\\ =&\widetilde{O}\left((T(Q+n(\max_i \operatorname{nnz}(a_i))+\sqrt{P\|c\|_1/n}+\|c\|_1/\sqrt{n})+T^2\|c\|_1/n\right)\end{aligned}$$

where we used $\|c\|_1 \geq m$.

In general the algorithm will call ShortStep a total of $T = \widetilde{O}(\sqrt{n}\log\frac{\mu^{(\text{init})}}{\mu^{(\text{end})}})$ times. However, we can speed-up the algorithm by stopping the loop after $T' = \Theta((Pn/\|c\|_1)^{1/2})$ iterations and then calling PathFollowing again for the obtained solution x, s, the current value of μ, and the desired target value $\mu^{(\text{end})}$. This way we call PathFollowing for a total of $\widetilde{O}((\|c\|_1/P)^{1/2}\log\frac{\mu^{(\text{init})}}{\mu^{(\text{end})}})$ times. Hence, the term $T^2\|c\|_1/n$ becomes $\widetilde{O}(\sqrt{P\|c\|_1}\log\frac{\mu^{(\text{init})}}{\mu^{(\text{end})}})$

Note that by assumption $P \leq \|c\|_1$, we can perform this reset before ever changing $\overline{\mu}$ in Line 4. Thus we can remove the $\widetilde{O}(\|c\|_1/\sqrt{n})$ term from our cost and we obtain the total cost

$$\widetilde{O}\left(\left(\sqrt{P\|c\|_1}+\sqrt{n}Q+n^{3/2}(\max_i \operatorname{nnz}(a_i))\right)\log\frac{\mu^{(\text{init})}}{\mu^{(\text{end})}}\right).$$

□

7 Minimum Cost Flow and Applications

In this section, we prove Theorem 1.4. Mincost flow with general demands can be reduced to single source / sink mincost flow by adding a super-source and super-sink to collect positive / negative demands respectively and assigning edges outwards with the proper capacities. We assume that we know the maximum flow value F as well, as we can perform a binary search on F, and add a s-t edge of sufficiently large demand and capacity. Therefore, the minimum cost flow problem we consider can be precisely formulated as

$$\min_{\substack{\mathbf{A}^\top x = F e_{s,t}\\ 0\leq x_e\leq u_e \forall e\in E}} c^\top x \tag{48}$$

where $\mathbf{A} \in \{-1,0,1\}^{E\times V}$ is an incidence matrix of G where $\mathbf{A}_{e,u} = -1, \mathbf{A}_{e,v} = 1$ for every edge $e = (u,v) \in E$. We denote the optimal value of the above LP by $\text{OPT}(G)$.

To prove Theorem 1.4, there are two main parts. For the first part, we show that, given an initial point $(x^{(\text{init})}, s^{(\text{init})})$ where $x^{(\text{init})}$ is an initial primal feasible solution and $s^{(\text{init})}$ is an initial dual slack of (48), the path following algorithm (Algorithm 5) returns $(x^{(\text{end})}, s^{(\text{end})})$ where $x^{(\text{end})}$ is a near-optimal and near-feasible flow in $\widetilde{O}(m+n^{1.5})$ time. This is proved in Section 7.1 by putting together the tools developed in previous sections.

For the second part, we describe how to obtain an initial point $(x^{(\text{init})}, s^{(\text{init})})$ and how to obtain an exactly optimal and feasible flow instead of a near-optimal and near-feasible flow, which gives Theorem 1.4. This part follows using standard techniques and, for completeness, we show how to do these tasks in Section 7.2.

Our algorithms in this section will use the following linear system solver.

Lemma 7.1 (See e.g. [ST04, KS16]). *There is an algorithm that, given a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ with at most two non-zero entries per row, a diagonal non-negative weight matrix $\mathbf{D} \in \mathbb{R}^{m \times m}$ and a vector $b \in \mathbb{R}^n$ such that $\mathbf{A}^\top \mathbf{D}\mathbf{A}$ is a symmetric diagonally dominant (SDD) matrix*[5] *and there exists a vector $x \in \mathbb{R}^n$ where $(\mathbf{A}^\top \mathbf{D}\mathbf{A})x = b$, w.h.p. returns a vector $\overline{x}$ such that $\|\overline{x} - x\|_{\mathbf{A}^\top \mathbf{D}\mathbf{A}} \leq \varepsilon \|x\|_{\mathbf{A}^\top \mathbf{D}\mathbf{A}}$ in $\widetilde{O}(\mathrm{nnz}(\mathbf{A}) \log \varepsilon^{-1})$ time.*

7.1 Path Following for Graph Problems

In Appendix F we show that HeavyHitter, InverseMaintenance, and HeavySampler data structures exist for graphical problems, and leverage these to show the following.

Lemma 7.2. *Consider a linear program*

$$\Pi : \min_{\substack{\mathbf{A}^\top x = b \\ \ell_i \leq x_i \leq u_i \forall i}} c^\top x$$

where $\mathbf{A}$ is obtained by removing one column (corresponding to one vertex) from an incidence matrix of a graph with n vertices and m edges. Let $\varepsilon = 1/(4C \log(m/n))$ for a large enough constant C. Given an ε-centered initial point $(x^{(\mathrm{init})}, s^{(\mathrm{init})}, \mu^{(\mathrm{init})})$ for Π and a target $\mu^{(\mathrm{end})}$, Algorithm 5 (when using graph-based data structures) returns an ε-centered point $(x^{(\mathrm{end})}, s^{(\mathrm{end})}, \mu^{(\mathrm{end})})$ in time

$$\widetilde{O}\left((m + n^{1.5} \cdot (\log W'' + |\log \mu^{(\mathrm{init})}/\mu^{(\mathrm{end})}|)) \cdot |\log \mu^{(\mathrm{init})}/\mu^{(\mathrm{end})}|\right)$$

where W'' is the ratio of largest to smallest entry in the vector $\phi''(x^{(\mathrm{init})}) = \frac{1}{(u - x^{(\mathrm{init})})^2} + \frac{1}{(x^{(\mathrm{init})} - \ell)^2}$.

Proof. We use Algorithm 4, which is an implementation of Algorithm 2. By Lemma 4.12 the algorithm returns an ϵ-centered $(x^{(\mathrm{end})}, s^{(\mathrm{end})}, \mu^{(\mathrm{end})})$. For the complexity, note that we have a (P, c, Q)-HeavyHitter, (P, c, Q)-HeavySampler, and (P, c, Q)-InverseMaintenance for $P = \widetilde{O}(m)$, $c_i = \widetilde{O}(1)$ for $i \in [m]$, $Q = \widetilde{O}(m/\sqrt{n} + n \log W')$ according to Lemma F.1, Corollary F.4, and Lemma F.2. Here W' is the ratio of the largest to smallest entry of any $\Phi''(\overline{x})$ maintained in Algorithm 4. By Lemma 4.46 this is bounded by $\widetilde{O}(\log W'' + |\log \mu^{(\mathrm{init})}/\mu^{(\mathrm{end})}|)$. Thus by using Lemma 6.7 the time complexity of Algorithm 4 is bounded by

$$\widetilde{O}\left((m + n^{1.5} \cdot (\log W'' + |\log \mu^{(\mathrm{init})}/\mu^{(\mathrm{end})}|)) \cdot |\log \mu^{(\mathrm{init})}/\mu^{(\mathrm{end})}|\right).$$

□

7.2 Initial and Final Points

In this section we discuss how to construct an initial flow which is on the central path. To do this, we augment the graph with a star which allows us to route a feasible flow. This corresponds to adding an identity block in the case of linear programs. Then we discuss how to extract a final point from our central path flow for sufficiently small path parameter μ.

Throughout this subsection, let $\varepsilon = 1/(4C \log(m/n))$ for a large enough constant C. We assume that u_e and c_e are integral and let W be the maximum absolute value of u_e and c_e over all edges e.

[5] A SDD matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is a matrix that is symmetric, and for all $i \in [n]$ satisfies $\mathbf{A}_{ii} \geq \sum_{j \in [n]} |\mathbf{A}_{ij}|$.

Defining Modified Graph and Linear Program with Trivial Initial Points. Recall that our path following algorithm (Algorithm 5) requires an initial point $(x^{(\text{init})}, s^{(\text{init})}, \mu^{(\text{init})})$ which is ε-centered (as defined in Definition 4.7). However, it is not clear how to quickly obtain such ε-centered point for an arbitrary graph G. Therefore, we will modify the graph G to another graph $\widetilde{G}$ where an ε-centered initial point can be easily defined.

Given $G = (V, E)$, let $\widetilde{G} = (V \cup \{z\}, E \cup \widetilde{E})$ where $\widetilde{E} = \{(v, z), (z, v) \mid v \in V\}$. That is, we add a bi-directional star rooted at z into G. We let $\begin{bmatrix} x^{(\text{init})} \\ \widetilde{x}^{(\text{init})} \end{bmatrix}$ denote an initial flow in $\widetilde{G}$ where $x^{(\text{init})}$ and $\widetilde{x}^{(\text{init})}$ specify the flow values on E and $\widetilde{E}$, respectively. For each $e \in E$, we set $x_e^{(\text{init})} \stackrel{\text{def}}{=} u_e/2$. For all star-edges $e \in \widetilde{E}$, we first set $\widetilde{x}_e^{(\text{init})} = 1$ (just to prevent them from having flow value too close to zero). Then, we additionally route the excess at each vertex induced by $x^{(\text{init})}$ using the star-edges from $\widetilde{E}$. More formally, for each $v \in V$, if the flow excess at v is $[\mathbf{A}^\top x^{(\text{init})} - Fe_{s,t}]_v > 0$, then we set $\widetilde{x}_{(v,z)}^{(\text{init})} \stackrel{\text{def}}{=} 1 + [\mathbf{A}^\top x^{(\text{init})} - Fe_{s,t}]_v$ and $\widetilde{x}_{(z,v)}^{(\text{init})} = 1$. Otherwise, the flow deficit at v is $[Fe_{s,t} - \mathbf{A}^\top x^{(\text{init})}]_v \geq 0$ and we set $\widetilde{x}_{(z,v)}^{(\text{init})} = 1 + [Fe_{s,t} - \mathbf{A}^\top x^{(\text{init})}]_v$ and $\widetilde{x}_{(v,z)}^{(\text{init})} \stackrel{\text{def}}{=} 1$. The capacity u_e and cost c_e of each original edge $e \in E$ stay the same. For each star-edge $e \in \widetilde{E}$, we set the capacity $\widetilde{u}_e = 2\widetilde{x}_e^{(\text{init})}$ and $\widetilde{c}_e \stackrel{\text{def}}{=} 50m\|u\|_\infty\|c\|_\infty$. Consider the modified linear program for minimum cost flow on $\widetilde{G}$

$$\begin{array}{ll} \min & c^\top x + \widetilde{c}^\top \widetilde{x}. \\ \widetilde{\mathbf{A}}^\top \begin{bmatrix} x \\ \widetilde{x} \end{bmatrix} = Fe_{s,t} & \\ 0 \leq x_e \leq u_e \, \forall e \in E & \\ 0 \leq \widetilde{x}_e \leq \widetilde{u}_e \, \forall e \in \widetilde{E} & \end{array} \tag{49}$$

where $\widetilde{\mathbf{A}}$ is an incidence matrix of $\widetilde{G}$. We denote the optimal value of the above LP by $\text{OPT}(\widetilde{G})$. As $\widetilde{G}$ is obtained from G by adding some edges, we obtain this simple fact:

Fact 7.3. $\text{OPT}(\widetilde{G}) \leq \text{OPT}(G)$.

For a small technical reason, we need to further modify the linear program since an incidence matrix is degenerate and our path following algorithm only works on a full rank matrix. Let $\widetilde{\mathbf{A}}_z$ be obtained from $\widetilde{\mathbf{A}}$ by removing a single column corresponds to the vertex z.[6]

Fact 7.4. *We have the following:*

1. $\widetilde{\mathbf{A}}_z$ *has full rank, and*
2. *For any* $\begin{bmatrix} x \\ \widetilde{x} \end{bmatrix}$, $\widetilde{\mathbf{A}}_z^\top \begin{bmatrix} x \\ \widetilde{x} \end{bmatrix} = Fe_{s,t}$ *if and only if* $\widetilde{\mathbf{A}}^\top \begin{bmatrix} x \\ \widetilde{x} \end{bmatrix} = Fe_{s,t}$.

Proof. (1) This is well-known and straightforward to verify, (2) Let $\begin{bmatrix} x \\ \widetilde{x} \end{bmatrix}$ satisfy $\widetilde{\mathbf{A}}_z^\top \begin{bmatrix} x \\ \widetilde{x} \end{bmatrix} = Fe_{s,t}$ which is the same as $\widetilde{\mathbf{A}}^\top \begin{bmatrix} x \\ \widetilde{x} \end{bmatrix} = Fe_{s,t}$ except that we do not require that the excess flow at z is zero. However, once we fix the excess at every vertex except at z, when we can determine the excess at z. In our case, the excesses are all zero except at s and t. This implies that the excess at z is zero. So $\widetilde{\mathbf{A}}^\top \begin{bmatrix} x \\ \widetilde{x} \end{bmatrix} = Fe_{s,t}$. Proving the claim in the reverse direction is trivial. □

By Fact 7.4(2), the following linear program is equivalent to (49).

$$\begin{array}{ll} \min & c^\top x + \widetilde{c}^\top \widetilde{x}. \\ \widetilde{\mathbf{A}}_z^\top \begin{bmatrix} x \\ \widetilde{x} \end{bmatrix} = Fe_{s,t} & \\ 0 \leq x_e \leq u_e \, \forall e \in E & \\ 0 \leq \widetilde{x}_e \leq \widetilde{u}_e \, \forall e \in \widetilde{E} & \end{array} \tag{50}$$

[6] The modification will actually work if we remove an arbitrary vertex. We choose the vertex z for convenience.

Lemma 7.5 (Initial point for the modified min cost flow)**.** *Let* $\begin{bmatrix} s^{(\text{init})} \\ \widetilde{s}^{(\text{init})} \end{bmatrix} = \begin{bmatrix} c \\ \widetilde{c} \end{bmatrix}$ *and* $\mu^{(\text{init})} = 100m^2W^3\varepsilon^{-1}$*. Then, the point* $\left(\begin{bmatrix} x^{(\text{init})} \\ \widetilde{x}^{(\text{init})} \end{bmatrix}, \begin{bmatrix} s^{(\text{init})} \\ \widetilde{s}^{(\text{init})} \end{bmatrix}, \mu^{(\text{init})} \right)$ *is* ε*-centered w.r.t. the linear program in (49) and (50).*

Proof. As $\widetilde{x}$ is defined such that all excess induced by x is canceled except at the source s and the sink t, we have that $\widetilde{\mathbf{A}}^\top \begin{bmatrix} x \\ \widetilde{x} \end{bmatrix} = Fe_{s,t}$. That is, the flow is exactly feasible in $\widetilde{G}$. Also, the vector $\begin{bmatrix} c \\ \widetilde{c} \end{bmatrix}$ is clearly dual feasible for $z = 0$ in Definition 4.7.

Now we bound the centrality.

By the choices of $x^{(\text{init})}$ and $\widetilde{x}^{(\text{init})}$ and Fact 4.3, we have the following. For each original edge $e \in E$, we have $\phi'(x_e^{(\text{init})}) = 0$ and $\phi''(x_e^{(\text{init})}) \geq 1/u_e^2$. For each star-edge $e \in \widetilde{E}$, we have $\phi'(\widetilde{x}_e^{(\text{init})}) = 0$ and $\phi''(\widetilde{x}_e^{(\text{init})}) \geq 1/\widetilde{u}_e^2 = 1/(2\widetilde{x}_e^{(\text{init})})^2 \geq 1/(2\|u\|_\infty n)^2$ because $\widetilde{x}_e^{(\text{init})} \leq \|u\|_\infty n$.

Therefore, for $x = \begin{bmatrix} x^{(\text{init})} \\ \widetilde{x}^{(\text{init})} \end{bmatrix}$ and $s = \begin{bmatrix} c \\ \widetilde{c} \end{bmatrix}$, because $\tau(x) \geq \frac{n}{m}$, we can bound

$$\left\| \frac{s + \mu^{(\text{init})}\tau(x)\phi'(x)}{\mu^{(\text{init})}\tau(x)\sqrt{\phi''(x)}} \right\|_\infty \leq \left\| \frac{50m\|u\|_\infty \|c\|_\infty}{\mu^{(\text{init})}(n/m)/(2\|u\|_\infty n)} \right\|_\infty \leq \frac{100m^2W^3}{\mu^{(\text{init})}} = \varepsilon.$$

□

Following the Path: Near-optimal Near-Feasible Flow for the Modified Graph. Given an initial point from Lemma 7.5, we can run the path following algorithm and obtain a near optimal point as formalized below:

Lemma 7.6. *Consider an* ε*-centered initial point*

$$\left(\begin{bmatrix} x^{(\text{init})} \\ \widetilde{x}^{(\text{init})} \end{bmatrix}, \begin{bmatrix} s^{(\text{init})} \\ \widetilde{s}^{(\text{init})} \end{bmatrix}, \mu^{(\text{init})} \right)$$

of (50) obtained from Lemma 7.5. Let $\mu^{(\text{end})} = 1/\operatorname{poly}(mW)$ *where* $\operatorname{poly}(mW)$ *is an arbitrarily large polynomial on* mW*. By invoking Lemma 7.2, we obtain an* ε*-centered point*

$$\left(\begin{bmatrix} x^{(\text{end})} \\ \widetilde{x}^{(\text{end})} \end{bmatrix}, \begin{bmatrix} s^{(\text{end})} \\ \widetilde{s}^{(\text{end})} \end{bmatrix}, \mu^{(\text{end})} \right)$$

of (50) in $\widetilde{O}(m \log W + n^{1.5} \log^2 W)$ *time.*

Proof. Note that we can invoke Lemma 7.2 because the constraint matrix $\widetilde{A}_z$ of (50) is exactly an incidence matrix with one column removed.

Now, we only need to analyze the running time. This follows because $\widetilde{\mathbf{A}}_z$ corresponds to the graph $\widetilde{G}$ with $O(n)$ vertices and $O(m)$ edges. Also, $\mu^{(\text{init})}/\mu^{(\text{end})} = \operatorname{poly}(mW)$. It remains to bound the ratio of largest to smallest entry in $\phi''(\begin{bmatrix} x^{(\text{init})} \\ \widetilde{x}^{(\text{init})} \end{bmatrix})$ denoted by W. It suffices to show that $W'' = \operatorname{poly}(W)$.

By definition of $x^{(\text{init})}$ and $\widetilde{x}^{(\text{init})}$ and Fact 4.3, we have $\phi''(x_e^{(\text{init})}) = \frac{2}{(u_e/2)^2}$ for each $e \in E$ and $\phi''(\widetilde{x}_e^{(\text{init})}) = \frac{2}{(\widetilde{x}_e^{(\text{init})})^2}$ where $1 \leq \widetilde{x}_e^{(\text{init})} \leq n\|u\|_\infty$ by definition of $\widetilde{x}_e^{(\text{init})}$. Therefore, we have that the ratio W'' is at most $\operatorname{poly}(n\|u\|_\infty) = \operatorname{poly}(mW)$ as desired. □

Near-optimal Feasible Flow for the Modified Graph. As the point from Lemma 7.6 may not be even feasible. In particular, there might exist a vertex u where total flow value entering u may not equal the one leaving u. Now, we show how to obtain a near-optimal *feasible* flow on the modified graph $\widetilde{G}$. Moreover, we additionally guarantee the flow value on every star-edge incident to the dummy vertex z is small. Intuitively, this is because the cost of every star-edge is very large. This is formalized in the following lemma:

Lemma 7.7. *Given an ε-centered point $\left(\begin{bmatrix} x \\ \widetilde{x} \end{bmatrix}, \begin{bmatrix} s \\ \widetilde{s} \end{bmatrix}, \mu\right)$ of (50) where $\mu < 1/n$, there exists $\begin{bmatrix} x^{(\text{final})} \\ \widetilde{x}^{(\text{final})} \end{bmatrix}$ which is a feasible flow for $\widetilde{G}$ where $c^\top x^{(\text{final})} + \widetilde{c}^\top \widetilde{x}^{(\text{final})} \leq \text{OPT}(\widetilde{G}) + \mu n$ in $\widetilde{O}(m \log W)$ time. Moreover, if there exists a feasible solution to the original LP (48), then $\|\widetilde{x}^{(\text{final})}\|_\infty < 0.1$.*

We can compute in $\widetilde{O}(m \log W)$ time $\begin{bmatrix} x^{(\text{apx-final})} \\ \widetilde{x}^{(\text{apx-final})} \end{bmatrix}$ whose each entry differs from $\begin{bmatrix} x^{(\text{final})} \\ \widetilde{x}^{(\text{final})} \end{bmatrix}$ by at most $1/(mW)^{10}$.

Proof. Lemma 4.11 implies the existence of $\begin{bmatrix} x^{(\text{final})} \\ \widetilde{x}^{(\text{final})} \end{bmatrix}$ which is also an exactly feasible solution of (50). By Fact 7.4(2), it is exactly feasible for (49) and so it is a feasible flow for $\widetilde{G}$.

For the "moreover" part. If there exists a feasible solution to (48), then $\text{OPT}(G) \leq m\|u\|_\infty \|c\|_\infty$. By Fact 7.3, we also have $\text{OPT}(\widetilde{G}) \leq m\|u\|_\infty \|c\|_\infty$. However, if $\|\widetilde{x}^{(\text{final})}\|_\infty \geq 0.1$, then $\widetilde{c}^\top \widetilde{x}^{(\text{final})} \geq 0.1 \cdot \min_{e \in \widetilde{E}} \widetilde{c}_e \geq 5m\|u\|_\infty \|c\|_\infty$. This contradicts the fact that

$$\widetilde{c}^\top \widetilde{x}^{(\text{final})} \leq \text{OPT}(\widetilde{G}) + \mu n - c^\top x^{(\text{final})} \leq 2m\|u\|_\infty \|c\|_\infty + 1.$$

Finally, as Lemma 4.11 says that $\begin{bmatrix} x^{(\text{final})} \\ \widetilde{x}^{(\text{final})} \end{bmatrix}$ can be obtained in time $O(m)$ plus the time for solving a Laplacian system exactly. We can use the approximate solver from Lemma 7.1 to obtain the desired $\begin{bmatrix} x^{(\text{apx-final})} \\ \widetilde{x}^{(\text{apx-final})} \end{bmatrix}$ in $\widetilde{O}(m \log W)$ time. □

Optimal Feasible Flow for the Original Graph. Lemma 7.7 only gives us a near-optimal flow for $\widetilde{G}$, but our goal is to obtain an exactly optimal flow for G. To achieve this goal, we will use a convenient lemma based on the Isolation Lemma below.

Lemma 7.8 ([DS08], or Lemma 8.10 of [BLN+20])**.** *Let $\Pi = (G, b, c)$ be an instance for minimum-cost flow problem where G is a directed graph with m edges, the demand vector $b \in \{-W, \ldots, W\}^V$, the cost vector $c \in \{-W, \ldots, W\}^E$ and the capacity vector $u \in \{0, \ldots, W\}^E$.*

Let the perturbed instance $\Pi' = (G, b, c')$ be such that $c'_e = c_e + z_e$ where z_e is a random number from the set $\left\{\frac{1}{4m^2W^2}, \ldots, \frac{2mW}{4m^2W^2}\right\}$. Let x' be a feasible flow for Π' whose cost is at most $\text{OPT}(\Pi') + \frac{1}{12m^2W^3}$ where $\text{OPT}(\Pi')$ is the optimal cost for problem Π'. With probability at least $1/2$, there exists an optimal feasible and integral flow x for Π such that $\|x - x'\|_\infty \leq 1/3$.

Combining these pieces allows us to prove Theorem 1.4.

Proof of Theorem 1.4. Given the input graph $G = (V, E, u, c)$, we first perturb the edge costs c to c' according Lemma 7.8. Let $G' = (V, E, u, c')$ denote the perturbed graph. Then, we construct the modified graph $\widetilde{G}'$ and the initial point for $\widetilde{G}'$ according to Lemma 7.5 (instead of the initial point for $\widetilde{G}$ as we did before). Then, we again invoke the path following algorithm in $\widetilde{O}(m \log W + n^{1.5} \log^2 W)$ time using Lemma 7.6 with $\mu^{(\text{end})} = \frac{1}{12m^2W^3n}$, and then invoke Lemma 7.7 for $\widetilde{G}'$ in $\widetilde{O}(m \log W)$ time. This process gives us $\begin{bmatrix} (x')^{(\text{apx-final})} \\ (\widetilde{x}')^{(\text{apx-final})} \end{bmatrix}$

whose each entry differs by at most $1/(mW)^{10}$ from a feasible flow $\begin{bmatrix}(x')^{(\text{final})}\\(\widetilde{x}')^{(\text{final})}\end{bmatrix}$ for $\widetilde{G}'$ where $c^\top(x')^{(\text{final})}+\widetilde{c}^\top(\widetilde{x}')^{(\text{final})} \leq \text{OPT}(\widetilde{G}')+\mu^{(\text{end})}n \leq \text{OPT}(\widetilde{G}')+\frac{1}{12m^2W^3}$. Moreover, $\|(\widetilde{x}')^{(\text{final})}\|_\infty < 0.1$ (otherwise, we declare that, with this specific choice of flow value F, there is no feasible solution for G' and hence for G).

Now, let $\begin{bmatrix}x^{(\text{final})}\\\widetilde{x}^{(\text{final})}\end{bmatrix}$ be obtained from $\begin{bmatrix}(x')^{(\text{apx-final})}\\(\widetilde{x}')^{(\text{apx-final})}\end{bmatrix}$ by rounding the flow on each edge to the nearest integer. Lemma 7.8 guarantees that, with probability at least $1/2$, there exists an integral optimal flow $\begin{bmatrix}x^{\text{OPT}}\\\widetilde{x}^{\text{OPT}}\end{bmatrix}$ that entry-wise differs from $\begin{bmatrix}x'^{(\text{final})}\\\widetilde{x}'^{(\text{final})}\end{bmatrix}$ by at most $1/3$. So $\begin{bmatrix}x^{\text{OPT}}\\\widetilde{x}^{\text{OPT}}\end{bmatrix}$ differs from $\begin{bmatrix}x'^{(\text{apx-final})}\\\widetilde{x}'^{(\text{apx-final})}\end{bmatrix}$ entry-wise by less than $1/2$. So, with probability at least $1/2$, $\begin{bmatrix}x^{(\text{final})}\\\widetilde{x}^{(\text{final})}\end{bmatrix} = \begin{bmatrix}x^{\text{OPT}}\\\widetilde{x}^{\text{OPT}}\end{bmatrix}$ is indeed optimal solution for $\widetilde{G}$. Moreover, as $\|(\widetilde{x}')^{(\text{final})}\|_\infty < 0.1$, we have that $\widetilde{x}^{(\text{final})} = 0$. Therefore, $x^{(\text{final})}$ is also a feasible flow for G with cost $c^\top x^{(\text{final})} = c^\top x^{(\text{final})} + \widetilde{c}^\top\widetilde{x}^{(\text{final})} = \text{OPT}(\widetilde{G})$. As $\text{OPT}(\widetilde{G}) \leq \text{OPT}(G)$ by Fact 7.3, we conclude that $x^{(\text{final})}$ is an optimal feasible flow for G. Finally, we can boost the success probability to hold with high probability by repeating the algorithm $O(\log n)$ times. This concludes the proof of Theorem 1.4. □

7.3 Application: Maximum Flow

In this section, we provide our main results regarding computing a maximum flow. In particular we prove the following corollary.

Corollary 7.9 (Maximum flow)**.** *There is an algorithm that, given a directed graph $G = (V, E, u, c)$ with n vertices and m edges that have integral capacities $u \in \mathbb{Z}^E_{\geq 0}$, with high probability, computes a maximum flow in $\widetilde{O}((m + n^{1.5}) \log \|u\|_\infty)$ time.*

Corollary 7.9 is an immediate corollary of our min-cost flow algorithm from Theorem 1.4 with modifications to decrease the $\log^2 \|u\|_\infty$ to $\log \|u\|_\infty$. This is done using a standard scaling technique (e.g. Section 6 of [AO91] and Chapter 2.6 of [Wil19]) and we provide the following proof of Corollary 7.9 only for completeness.

Proof of Theorem 7.9. For any graph G and flow f in G, let G_f be the residual graph of G w.r.t. f. Moreover, let $G_f(\Delta)$ denote the graph obtained G_f by removing all edges with capacity in G_f less than Δ. Now, consider the following algorithm.

- $\Delta = 2^{\lfloor \log_2 \|u\|_\infty \rfloor}$
- While $\Delta \geq 1$,
  - find a maximum flow f' in $G_f(\Delta)$.
  - set $f \leftarrow f + f'$ and $\Delta \leftarrow \Delta/2$.
- Return f.

Clearly, this algorithm correctly computes a maximum flow because, after the last iteration when $\Delta = 1$, there is no augmenting path left in G_f. Observe the following:

Proposition 7.10. *In the beginning of each iteration in the while loop, the value of a maximum flow in G_f is at most $2m\Delta$.*

Proof. This is trivially true in the first iteration. For other iterations, we know from the previous iteration that the value of a maximum flow in $G_f(2\Delta)$ is zero. As G_f can be obtained from $G_f(2\Delta)$ by adding at most m edges each of which has capacity less than 2Δ, the value of a maximum flow in G_f is at most $2m\Delta$. □

The above observation implies that, to compute a maximum flow in $G_f(\Delta)$, we can safely cap the capacity each edge to be at most $2m\Delta$. (That is, if the capacity of e in G_f is more than $2m\Delta$, set it to be $2m\Delta$.) As the ratio between the maximum and minimum capacity is at most $2m$, we can compute the maximum flow in $G_f(\Delta)$ in $\widetilde{O}(m+n^{1.5})$ using Theorem 1.4. Since there are $\lfloor \log_2 \|u\|_\infty \rfloor$ iterations, the total running time of algorithm is $\widetilde{O}((m+n^{1.5}) \log \|u\|_\infty)$. This completes the proof of Corollary 7.9. □

8 General Linear Programs

In this section, we prove Theorem 1.1 and Theorem 1.2. There are two main parts. For the first part, we show that, given an initial point $(x^{(\text{init})}, s^{(\text{init})})$, the path following algorithm (Algorithm 5) returns $(x^{(\text{end})}, s^{(\text{end})})$ where $x^{(\text{end})}$ is a near-optimal point in $\widetilde{O}(mn+n^{2.5})$ time. This is proved in Section 7.1 by putting together the tools developed in previous sections.

For the second part, we describe how to obtain an initial point $(x^{(\text{init})}, s^{(\text{init})})$ and how to obtain a near optimal primal solution in Section 8.2 hence proving Theorem 1.1. In Section 8.3, we then show how to extract a near optimal dual solution hence proving Theorem 1.2. This part uses standard techniques and we show how to do these tasks for completeness.

Our algorithms in this section will use the following linear system solver.

Lemma 8.1 ([NN13, LMP13, CLM+15]). *There is an algorithm that, given a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, a diagonal non-negative weight matrix $\mathbf{D} \in \mathbb{R}^{m\times m}$, and a vector $b \in \mathbb{R}^n$ such that there exists a vector $x \in \mathbb{R}^n$ where $(\mathbf{A}^\top \mathbf{D}\mathbf{A})x = b$, w.h.p. returns a vector $\overline{x}$ such that $\|\overline{x}-x\|_{\mathbf{A}^\top\mathbf{D}\mathbf{A}} \leq \varepsilon \|x\|_{\mathbf{A}^\top\mathbf{D}\mathbf{A}}$ in $\widetilde{O}((\text{nnz}(\mathbf{A})+n^\omega)\log \varepsilon^{-1})$ time.*

8.1 Path Following for General LPs

In Appendix B we give efficient HeavyHitter, InverseMaintenance, and HeavySampler data structures for general linear programs, and use these to show the following.

Lemma 8.2. *Consider a linear program*

$$\Pi: \min_{\substack{\mathbf{A}^\top x = b \\ \ell_i \leq x_i \leq u_i \forall i}} c^\top x$$

where $\mathbf{A} \in \mathbb{R}^{m\times n}$. Let $\varepsilon = 1/(4C\log(m/n))$ for a large enough constant C. Given an ε-centered initial point $(x^{(\text{init})}, s^{(\text{init})}, \mu^{(\text{init})})$ for Π and a target $\mu^{(\text{end})}$, Algorithm 5 (when using matrix-based data structures) returns an ε-centered point $(x^{(\text{end})}, s^{(\text{end})}, \mu^{(\text{end})})$ in time

$$\widetilde{O}\left((mn+n^{2.5}) \cdot |\log \mu^{(\text{init})}/\mu^{(\text{end})}|\right).$$

Proof. We use Algorithm 4, which is an implementation of Algorithm 2. By Lemma 4.12 the algorithm returns an ϵ-centered $(x^{(\text{end})}, s^{(\text{end})}, \mu^{(\text{end})})$. For the complexity, note that we have a (P,c,Q)-HeavyHitter, (P,c,Q)-HeavySampler, and (P,c,Q)-InverseMaintenance for $P = \widetilde{O}(\text{nnz}(\mathbf{A})+n^\omega)$, $c_i = \widetilde{O}(n)$ for $i \in [m]$, $Q = \widetilde{O}(n^2+m\sqrt{n})$ according to Lemma B.1, Corollary B.6, and Lemma B.2. By using Lemma 6.7 the time complexity of Algorithm 4 is bounded by

$$\widetilde{O}\left((mn+n^{2.5}) \cdot |\log \mu^{(\text{init})}/\mu^{(\text{end})}|\right).$$

□

8.2 Initial and Final Primal Solutions

In this section, we prove Theorem 1.1 using Lemma 8.2. Throughout this section, let $\delta' = \frac{\delta}{10mW^2}$ and let $\varepsilon = 1/(4C\log(m/n))$ for a large enough constant C. The path following algorithm, and therefore Lemma 8.2, requires an initial point which is ε-centered. However, it is not clear how to obtain such point efficiently. We first show how to modify the linear program so that it is easy to compute an ε-centered initial point similar to how we did in Section 7.2. As described in Section 7, we construct initial point by adding an identity block to the constraint matrix $\mathbf{A}$. Interestingly, this is simpler than in previous work [BLSS20, BLN+20] because we are able to handle two-sided constraints.

Given matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, vector $b \in \mathbb{R}^n$, vector $c \in \mathbb{R}^m$, an accuracy parameter δ, and the linear program

$$\min_{\substack{\mathbf{A}^\top x = b \\ \ell_i \le x_i \le u_i \forall i}} c^\top x, \tag{51}$$

define matrix $\widetilde{\mathbf{A}} \stackrel{\text{def}}{=} \begin{bmatrix} \mathbf{A} \\ \beta \mathbf{I}_n \end{bmatrix}$, $\beta = \|b - \mathbf{A}^\top x^{(\text{init})}\|_\infty / \Xi$, $\Xi \stackrel{\text{def}}{=} \max_i |u_i - \ell_i|$, $x_i^{(\text{init})} \stackrel{\text{def}}{=} (\ell_i + u_i)/2$, $\widetilde{x}^{(\text{init})} \stackrel{\text{def}}{=} \frac{1}{\beta}(b - \mathbf{A}^\top x^{(\text{init})})$. By flipping the signs of columns of $\mathbf{A}$, we can assume $\widetilde{x}^{(\text{init})} \ge 0$. If $\widetilde{x}_i^{(\text{init})} = 0$, enforce that coordinate to be 0 always by removing the variable $\widetilde{x}_i^{(\text{init})}$ (as it is unnecessary for constructing the initial point), and otherwise define $\widetilde{\ell}_i = -\Xi$ and $\widetilde{u}_i = 2\widetilde{x}_i^{(\text{init})} + \Xi$ (the terms $-\Xi$ and $+\Xi$ are just to ensure that $\widetilde{u}_i > \widetilde{\ell}_i$). We define $\widetilde{c} \stackrel{\text{def}}{=} 2\|c\|_1/\delta'$. We overload notation and let $\widetilde{c}$ be the vector in $\mathbb{R}^n$ with value $\widetilde{c}$ in all coordinates.

Consider the modified linear program

$$\min_{\substack{\widetilde{\mathbf{A}}^\top \begin{bmatrix} x \\ \widetilde{x} \end{bmatrix} = b \\ \ell_i \le x_i \le u_i \forall i \\ \widetilde{\ell}_i \le \widetilde{x}_i \le \widetilde{u}_i \forall i}} c^\top x + \widetilde{c}^\top \widetilde{x}. \tag{52}$$

Lemma 8.3 (Initial point for the modified LP)**.** *For the linear program in (52), the point*

$$\left(\begin{bmatrix} x^{(\text{init})} \\ \widetilde{x}^{(\text{init})} \end{bmatrix}, \begin{bmatrix} c \\ \widetilde{c} \end{bmatrix}, \mu \right)$$

is ε-centered for $\mu = \frac{8m\|c\|_1 \Xi}{\varepsilon \delta'}$.

Proof. By the definition of $\widetilde{x}^{(\text{init})} = \beta^{-1}(b - \mathbf{A}^\top x^{(\text{init})})$ it is clear that $\widetilde{\mathbf{A}} \begin{bmatrix} x \\ \widetilde{x} \end{bmatrix} = b$, so the point is exactly feasible. Also, the vector $\begin{bmatrix} c \\ \widetilde{c} \end{bmatrix}$ is clearly dual feasible as in Definition 4.7 by taking $z = 0$.

Now we bound the centrality. Recall that the barrier function $\phi(x)$ on the interval $[\ell, u]$ is given by $\phi(x) = -\log(x - \ell) - \log(u - x)$. Therefore, $\phi'((\ell + u)/2) = 0$, and $\phi''(x) \ge 1/(u - \ell)^2$. In particular, this is lower bounded by $1/\Xi^2$ for all original constraints. For the new constraints, similarly, we have that $|\widetilde{u}_i - \widetilde{\ell}_i| \le \|\frac{1}{\beta}(b - \mathbf{A}^\top x^{(\text{init})})\|_\infty \le 4\Xi$. Hence, the Hessian $\phi''(x_i)$ is lower bounded by $1/(16\Xi^2)$ for all constraints.

Therefore, for $x = \begin{bmatrix} x^{(\text{init})} \\ \widetilde{x}^{(\text{init})} \end{bmatrix}$ and $s = \begin{bmatrix} c \\ \widetilde{c} \end{bmatrix}$, because $\tau(x) \ge \frac{n}{m}$, we can bound

$$\left\| \frac{s + \mu\tau(x)\phi'(x)}{\mu\tau(x)\sqrt{\phi''(x)}} \right\|_\infty \le 4mn^{-1}\mu^{-1}\Xi\|s\|_\infty \le \varepsilon$$

by the choice of μ. $\square$

Next, we show that a solution to the modified LP can be used to round to nearly feasible and optimal solutions to the original LP.

Lemma 8.4 (Final Point for the Original LP)**.** *Assume that the the linear program in (51) has a feasible solution. Given an ε-centered point for the modified LP in (52) for $\mu = \delta'\|c\|_1\Xi/Cn$ for some large enough constant C and some $\delta \leq 1$, we can in $\widetilde{O}((\mathrm{nnz}(\mathbf{A}) + n^\omega)\log(W/\delta))$ time compute a point $x^{(\mathrm{final})}$ that satisfies*

$$c^\top x^{(\mathrm{final})} \leq \min_{\substack{\mathbf{A}^\top x = b \\ \ell_i \leq x_i \leq u_i \forall i}} c^\top x + \delta \text{ and } \|\mathbf{A}^\top x^{(\mathrm{final})} - b\|_\infty \leq \delta$$

and $\ell_i \leq x_i^{(\mathrm{final})} \leq u_i$ for all i.

Proof. By Lemma 4.11 we can compute an exactly feasible point $\begin{bmatrix} x^{(\mathrm{final})} \\ \widetilde{x}^{(\mathrm{final})} \end{bmatrix}$, and error $\lesssim n\mu \leq \delta'\|c\|_1\Xi \leq \delta$ off optimal with a single solve to $\mathbf{A}^\top\mathbf{D}\mathbf{A}$. We discuss at the end how to deal with inexact solvers. We claim that $x^{(\mathrm{final})}$ satisfies the necessary conditions. Note that the optimum for (51) is at least that of (52) as long as (51) is feasible. Therefore, we have that

$$c^\top x^{(\mathrm{final})} \leq \min_{\substack{\mathbf{A}^\top x = b \\ \ell_i \leq x_i \leq u_i \forall i}} c^\top x - \widetilde{c}^\top \widetilde{x}^{(\mathrm{final})} + \delta \leq \min_{\substack{\mathbf{A}^\top x = b \\ \ell_i \leq x_i \leq u_i \forall i}} c^\top x + \delta.$$

Finally, we show that $\|\mathbf{A}^\top x^{(\mathrm{final})} - b\|_\infty \leq \delta'\|\mathbf{A}^\top x^{(\mathrm{init})} - b\|_\infty$, this suffices for the first claim.

Because $x^{(\mathrm{final})}$ is feasible, note that $\|\mathbf{A}^\top x^{(\mathrm{final})} - b\|_\infty = \beta\|\widetilde{x}^{(\mathrm{final})}\|_\infty$. Assume $\|\mathbf{A}^\top x^{(\mathrm{final})} - b\|_\infty \leq \delta'\|\mathbf{A}^\top x^{(\mathrm{init})} - b\|_\infty$ is false, using $\beta = \|\mathbf{A}^\top x^{(\mathrm{init})} - b\|_\infty/\Xi$, then there is some i such that $\widetilde{x}_i^{(\mathrm{final})} > \delta'\Xi$. In this case, we get that

$$\begin{aligned} c^\top x^{(\mathrm{final})} &= \left(c^\top x^{(\mathrm{final})} + \widetilde{c}^\top \widetilde{x}^{(\mathrm{final})}\right) - \widetilde{c}^\top \widetilde{x}^{(\mathrm{final})} \leq \min_{\substack{\mathbf{A}^\top x = b \\ \ell_i \leq x_i \leq u_i \forall i}} c^\top x + \delta'\|c\|_1\Xi - \widetilde{c}^\top \widetilde{x}^{(\mathrm{final})} \\ &< \sum_i \min(c_i\ell_i, c_iu_i) + 2\|c\|_1\Xi - \widetilde{c}\delta'\Xi = \sum_i \min(c_i\ell_i, c_iu_i) \end{aligned}$$

This is clearly a contradiction, as $c^\top x^{(\mathrm{final})} \geq \sum_i \min(c_i\ell_i, c_iu_i)$ because $x^{(\mathrm{final})}$ is feasible. Therefore, we have

$$\|\mathbf{A}^\top x^{(\mathrm{final})} - b\|_\infty \leq \delta'\|\mathbf{A}^\top x^{(\mathrm{init})} - b\|_\infty \leq \delta$$

because $\|\mathbf{A}^\top x^{(\mathrm{init})} - b\|_\infty \leq m\|\mathbf{A}\|_\infty \max\{\|u\|_\infty, \|\ell\|_\infty\} + \|b\|_\infty\} \leq 2mW^2$.

Finally, we have an approximate instead of exact solver for $\begin{bmatrix} x^{(\mathrm{final})} \\ \widetilde{x}^{(\mathrm{final})} \end{bmatrix}$. However, we can compute the vector where each entry differs from $\begin{bmatrix} x^{(\mathrm{final})} \\ \widetilde{x}^{(\mathrm{final})} \end{bmatrix}$ at most $\delta_{\mathrm{inexact}}$ in time $\widetilde{O}((\mathrm{nnz}(\mathbf{A}) + n^\omega)\log(W/\delta_{\mathrm{inexact}}))$ using Lemma 8.1. As long as we can guarantee that $\delta_{\mathrm{inexact}} \leq 1/\operatorname{poly}(mW\lambda_{\min}(\widetilde{\mathbf{A}}^\top\mathbf{D}\widetilde{\mathbf{A}}))$, all entries will differ by an arbitrarily small polynomial, so this approximate vector satisfies all the conditions of Lemma 8.4 as well. Recall that in Lemma 4.11 that $\mathbf{D} = \overline{\mathbf{T}}^{-1}\Phi''(x)^{-1}$, and $\log \Phi''(x)^{-1} \geq -\widetilde{O}(\log W + \log \mu^{(\mathrm{final})} + \log \|c\|_\infty)$ by Lemma 4.46. Hence

$$\begin{aligned} \log \lambda_{\min}(\mathbf{A}^\top\mathbf{D}\mathbf{A}) &\geq -\widetilde{O}(\log W + \log(1/\mu^{(\mathrm{final})}) + \log \|c\|_\infty) + \log \lambda_{\min}(\widetilde{\mathbf{A}}^\top\widetilde{\mathbf{A}}) \\ &\geq -\widetilde{O}(\log(W/\delta)) \end{aligned}$$

where we used that $\lambda_{\min}(\widetilde{\mathbf{A}}^\top\widetilde{\mathbf{A}}) \geq \beta^2 \geq \operatorname{poly}(1/(mW))$ and $\mu^{(\mathrm{final})} \geq \delta\operatorname{poly}(1/(mW))$. $\square$

Now, we are ready to prove Theorem 1.1 by combining the two lemmas above.

Proof of Theorem 1.1. Using Lemma 8.3, we can obtain an ε-centered initial point

$$\left(\begin{bmatrix} x^{(\text{init})} \\ \widetilde{x}^{(\text{init})} \end{bmatrix}, \begin{bmatrix} c \\ \widetilde{c} \end{bmatrix}, \mu^{(\text{init})}\right)$$

for the modified linear program (52) in $O(\text{nnz}(\mathbf{A}))$ time where $\mu^{(\text{init})} = 4m\|c\|_1\Xi/\varepsilon\delta'$.

Let $\mu^{(\text{end})} = \delta'\|c\|_1\Xi/Cn$ where C is a large enough constant. We invoke Lemma 8.2 which returns $\left(\begin{bmatrix} x^{(\text{end})} \\ \widetilde{x}^{(\text{end})} \end{bmatrix}, \begin{bmatrix} s^{(\text{end})} \\ \widetilde{s}^{(\text{end})} \end{bmatrix}, \mu^{(\text{end})}\right)$ in time $\widetilde{O}((mn + n^{2.5}) \cdot \log^2 W)$. This running time follows because $\mu^{(\text{init})}/\mu^{(\text{end})} = \text{poly}(mW/\delta)$. It remains to bound the ratio W'' of largest to smallest entry in $\phi''(\begin{bmatrix} x^{(\text{init})} \\ \widetilde{x}^{(\text{init})} \end{bmatrix})$. For each entry in $\widetilde{x}^{(\text{init})}$, we have $\phi''(\widetilde{x}_i^{(\text{init})}) = 2/(\widetilde{x}_i^{(\text{init})} + \Xi)^2$. Note that $\Xi \le \widetilde{x}_i^{(\text{init})} + \Xi \le 2\Xi \lesssim W$, where $\Xi = \max_i |u_i - \ell_i|$ was set previously For each entry in $x^{(\text{init})}$, we have $\phi''(x_i^{(\text{init})}) = \Theta(1/(u_i - \ell_i)^2)$. As $W \ge \frac{\max_i(u_i-\ell_i)}{\min_i(u_i-\ell_i)}$, we have that indeed the ratio $W'' \le \text{poly}(mW)$ as desired.

As the last step, we give $\left(\begin{bmatrix} x^{(\text{end})} \\ \widetilde{x}^{(\text{end})} \end{bmatrix}, \begin{bmatrix} s^{(\text{end})} \\ \widetilde{s}^{(\text{end})} \end{bmatrix}, \mu^{(\text{end})}\right)$ as an input to Lemma 8.4 and obtain a final point $x^{(\text{final})}$ which satisfies all condition of Theorem 1.1 in time $\widetilde{O}((\text{nnz}(\mathbf{A})+n^\omega)\log(W/\delta))$. In total, the running time of the algorithm is $\widetilde{O}((mn + n^{2.5})\log(W/\delta))$. □

8.3 Final Dual Solutions

In this section, we prove Theorem 1.2. Before proving it, we show the key subroutine for extracting a dual solution from the primal LP as formalized below. By scaling x and $\mathbf{A}$, we can focus on the $\ell = -1, u = 1$ case. Additionally, this formulation suffices for our application of solving MDPs in Theorem 1.3.

Lemma 8.5 (Dual solution bound)**.** *Given an ε-centered point (x, s, μ) where $\varepsilon \le 1/(C\log(m/n))$ and $\mu \le \delta/Cn$ for sufficiently large C to the LP*

$$\min_{\substack{\mathbf{A}^\top x = 0 \\ -1 \le x_i \le 1 \forall i}} c^\top x, \tag{53}$$

we can compute in time $\widetilde{O}((\text{nnz}(\mathbf{A}) + n^\omega)\log(W/\delta))$ a vector $z \in \mathbb{R}^n$ satisfying

$$\|\mathbf{A}z + c\|_1 \le \min_{z \in \mathbb{R}^n} \|\mathbf{A}z + c\|_1 + \delta.$$

Proof. Define

$$\text{OPT} = \min_{\substack{\mathbf{A}^\top x = 0 \\ -1 \le x_i \le 1 \forall i}} c^\top x.$$

By Sion's minimax theorem, we have that

$$\begin{aligned} &\min_{\substack{\mathbf{A}^\top x = 0 \\ -1 \le x_i \le 1 \forall i}} c^\top x = \min_{\|x\|_\infty \le 1} \max_{z \in \mathbb{R}^n} c^\top x + z^\top \mathbf{A}^\top x \\ &= \max_{z \in \mathbb{R}^n} \min_{\|x\|_\infty \le 1} (\mathbf{A}z + c)^\top x = \max_{z \in \mathbb{R}^n} -\|\mathbf{A}z + c\|_1 = -\min_{z \in \mathbb{R}^n} \|\mathbf{A}z + c\|_1. \end{aligned}$$

Hence, we have that $\min_{z \in \mathbb{R}^n} \|\mathbf{A}z + c\|_1 = -\text{OPT}$. Now, find $x^{(\text{final})}$ and $s^{(\text{final})} = \mathbf{A}z + c$ satisfying the conclusions of Lemma 4.11. Specifically, z may be computed as $z = (\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top(s^{(\text{final})} - c)$ in time $\widetilde{O}((\text{nnz}(A) + n^\omega)\log\delta^{-1})$ by Lemma 8.1.

We now prove that this z satisfies the conclusions of this lemma: $\|\mathbf{A}z+c\|_1 \le \min_{z\in\mathbb{R}^n} \|\mathbf{A}z+c\|_1 + \delta$. To simplify notation, we will write x for $x^{(\text{final})}$ and s for $s^{(\text{final})}$. Note that because $\mathbf{A}^\top x = 0$,

$$\|\mathbf{A}z + c\|_1 = (\mathbf{A}z)^\top x + \|s\|_1 = -c^\top x + x^\top s + \|s\|_1 \le -\text{OPT} + x^\top s + \|s\|_1,$$

as $c^\top x \ge \text{OPT}$ by definition. Therefore, it suffices to bound $x^\top s + \|s\|_1$. We will show that $x_i s_i + |s_i| \lesssim \mu\tau_i$. This suffices, as then

$$x^\top s + \|s\|_1 \lesssim \sum_i \mu\tau_i \lesssim n\mu \le \delta$$

for sufficiently large choice of C in the definition of μ.

To show this, define $y = \frac{s+\mu\tau\phi'(x)}{\mu\tau\sqrt{\phi''(x)}}$, so that $\|y\|_\infty \le \frac{1}{10}$ by Claim A.6, which shows approximate centrality of the final point returned. We have that

$$y_i\sqrt{\phi_i''(x_i)} - \phi_i'(x_i) = \mu^{-1}\tau_i^{-1}s_i. \tag{54}$$

If $|x_i| \le \frac{1}{2}$, then both $\phi_i''(x_i)$ and $|\phi_i'(x_i)|$ are $O(1)$. Hence, eq. (54) shows that $x_i s_i + |s_i| \lesssim |s_i| \lesssim \mu\tau_i$. If $|x_i| \ge \frac{1}{2}$, we have

$$y_i\sqrt{\phi_i''(x_i)} = y_i\sqrt{\frac{1}{(1-x_i)^2} + \frac{1}{(1+x_i)^2}} \le \frac{2y_i}{1-|x_i|} \le \frac{1}{5(1-|x_i|)} \le |\phi_i'(x_i)|.$$

Hence, eq. (54) shows that $0 \ge \mu^{-1}\tau_i^{-1}s_i \ge -\frac{1}{1-x_i}$ and

$$x_i s_i + |s_i| = -(1-x_i)s_i \le \mu\tau_i.$$

The proof for $x_i \le -\frac{1}{2}$ is similar. Therefore, we proved the claim $x_i s_i + |s_i| \lesssim \mu\tau_i$ in all the cases. Handling the inexactness of solvers may be done as in the proof of Lemma 8.4. ☐

With the above lemma at hand, we are ready to prove Theorem 1.2.

Proof of Theorem 1.2. First, observe that we can easily define an initial solution

$$(x^{(\text{init})}, s^{(\text{init})}, \mu^{(\text{init})}) \stackrel{\text{def}}{=} (0, c, \varepsilon^{-1}\|c\|_\infty m/n)$$

of (53). Note that $(x^{(\text{init})}, s^{(\text{init})}, \mu^{(\text{init})})$ is ε-centered because $x^{(\text{init})} = 0$ and $s^{(\text{init})} = c$ are exactly feasible. It remains to bound the centrality. As the barrier function is defined on $[-1, 1]$, we have $\phi'(0) = 0$, and $\phi''(0) = 2$ by Fact 4.3. As $\tau(x^{(\text{init})}) \ge \frac{n}{m}$, we have

$$\left\| \frac{s^{(\text{init})} + \mu^{(\text{init})}\tau(x^{(\text{init})})\phi'(x^{(\text{init})})}{\mu^{(\text{init})}\tau(x^{(\text{init})})\sqrt{\phi''(x^{(\text{init})})}} \right\|_\infty \le \frac{m\|c\|_\infty}{\mu n} \le \varepsilon$$

as desired.

Then, we invoke Lemma 8.2 to obtain an ε-centered point $(x^{(\text{end})}, s^{(\text{end})}, \mu^{(\text{end})})$ of (53) where $\mu^{(\text{end})} = \delta/Cn$. Lemma 8.2 takes time $\widetilde{O}\left((mn + n^{2.5}) \cdot \log(W/\delta)\right)$ because $\mu^{(\text{init})}/\mu^{(\text{end})} = \text{poly}(mW/\delta)$.

Lastly, we invoke Lemma 8.5 to obtain a vector z which satisfies the guarantee of Theorem 1.2 in time $\widetilde{O}((\text{nnz}(\mathbf{A}) + n^\omega)\log(W/\delta))$ by using Lemma 8.1. Therefore, the total running time is $\widetilde{O}\left((mn + n^{2.5})\log(W/\delta)\right)$. ☐

Year	Authors	Refs	Algorithm	Running Time up to $\widetilde{O}(\cdot)$
1990	Tseng, Littman, Dean, Kaelbling	[Tse90, LDK95]	Value Iteration	$\|S\|^2\|A\|\frac{1}{1-\gamma}$
2014	Lee, Sidford	[LS14, LS15, SWWY18]	IPM	$\|S\|^{2.5}\|A\|$
2018	Sidford, Wang, Wu, Yang	[SWWY18]	High Precision RVI	$\|S\|^2\|A\| + \frac{\|S\|\|A\|}{(1-\gamma)^3}$
2020		This paper	Robust IPM	$S\|^2\|A\| + \|S\|^{2.5}$

Table 6: High-precision algorithms for discounted MDPs. This table gives only the fastest high precision results, i.e. those that depend polylogarithmically on ϵ. We ignore the $\widetilde{O}$ in the running time, and suppress factors of $\log\left(\max(|S|, |A|, \varepsilon^{-1}, (1-\gamma)^{-1}, M)\right)$. RVI denotes Randomized Value Iteration. [SWWY18] showed that discounted MDPs could be solved via linear programming, so previous results towards faster linear programming immediately gave results for discounted MDPs.

8.4 Application: Discounted Markov Decision Process

In this section, we show an algorithm (Theorem 1.3) for solving the discounted Markov Decision Process problem in time $\widetilde{O}((|S|^2|A| + |S|^{2.5}) \log(\frac{M}{(1-\gamma)\epsilon}))$ (these parameters are defined below). See Table 6 for the comparison with previous results. Below, we formally define the problem and prove this bound.

Problem Definition. A discounted Markov Decision Process (DMDP) is specified by a tuple (S, A, P, r, γ) where S is a the finite state space, A is the finite action space, $P = \{p_a\}_{a \in A}$ is the collection of state-action-state transition probabilities where $p_a(i, j)$ denote the probability of going to state j from state i when taking action a, r is the collection of state-action rewards where $r_a(i) \in [-M, M]$ is the collected reward when we are currently in state i and take action a, and $\gamma \in (0, 1)$ is a discount factor.

Given a DMDP (S, A, P, r, γ), the *value operator* $T : \mathbb{R}^S \to \mathbb{R}^S$ is defined for all $u \in \mathbb{R}^S$ and $i \in S$ by

$$T(u)_i = \max_{a \in A} [r_a(i) + \gamma \cdot p_a(i)^\top u]$$

where $p_a(i) \in \mathbb{R}^S$ with $(p_a(i))_j = p_a(i, j)$. It is known that there is a unique vector v^* such that $T(v^*) = v^*$.

A vector $\pi \in A^S$ that tells the actor which action to take from any state is called a *policy* and π_i denotes the action prescribed by π to be taken at state $i \in S$. The *value operator associated* T_π *with* π is defined for all $u \in \mathbb{R}^S$ and $i \in S$ by

$$T_\pi(u)_i = r_{\pi_i}(i) + \gamma \cdot p_{\pi_i}(i)^\top u.$$

Note that T_π can be viewed as the value operator for the modified DMDP where the only available action from each state is given by the policy π. Let v_π denote the unique vector such that $T_\pi(v_\pi) = v_\pi$.

We says that values $u \in \mathbb{R}^S$ are ε-optimal if $\|v^* - u\|_\infty \leq \varepsilon$ and we say that a policy $\pi \in A^S$ is ε-optimal if $\|v^* - v_\pi\|_\infty \leq \varepsilon$, i.e. the values of the policy π are ε-optimal. Our goal is to find an ε-optimal policy π.

Reductions to Solving LP and ℓ_1 Regression. In Section B of [SWWY18], the authors show how to reduce the problem of finding an ε-optimal policy to finding ϵ-approximate solution of the following LP. We leverage this reduction along with Theorem 1.2 (ℓ_1 regression) to prove our main result Theorem 1.3.

Definition 8.6 (DMDP linear program)**.** *We call the following linear program the* DMDP LP

$$\min_{\mathbf{A}v \geq r} v^\top \vec{1}. \tag{55}$$

where $r \in \mathbb{R}^{(S \times A)}$ is the vector of rewards, i.e. $r_{i,a} = r_a(i)$ for all $i \in S$ and $a \in A$ and $\mathbf{A} = \mathbf{E} - \gamma \mathbf{P}$ where $\mathbf{E} \in \mathbb{R}^{(S \times A) \times S}$ is the matrix where for all $i, j \in S$ and $a \in A$ we have that

*the j-th entry of row (i,a) of $\mathbf{E}$ is 1 if $i=j$ and 0 otherwise, and $\mathbf{P}\in\mathbb{R}^{(S\times A)\times S}$ is a matrix where for all $i\in S$ and $a\in A$ we have that row (i,a) of $\mathbf{P}$ is $p_a(i)$. We call a vector $v\in\mathbb{R}^S$ an ϵ-*approximate DMDP LP solution *if $\mathbf{A}v\geq r-\epsilon\vec{1}$ and $v^\top\vec{1}\leq OPT+\epsilon$ where OPT is the optimal value of the DMDP LP, Equation (55).*

The reduction is stated as follows:

Lemma 8.7 (Lemma B.3 of [SWWY18])**.** *If v is an ϵ-approximate DMDP LP solution and if $\pi\in A^S$ is defined with $\pi_i=\operatorname{argmax}_{a\in A} r_a+\gamma\cdot p(i,a)^\top v$ for all $i\in S$ then π is an $8\epsilon|S|(1-\gamma)^{-2}$-optimal policy.*

[SWWY18] further reduces the problem of finding an ϵ-approximate DMDP LP solution to finding a ϵ-optimal solution of the following instance of the ℓ_1 regression problem.

Definition 8.8 (DMDP ℓ_1-regression)**.** *For a given DMDP (S,A,P,r,γ) and a parameter α we call the following ℓ_1 regression problem the* DMDP ℓ_1 problem

$$\min_v f(v)=\left|\alpha\left(\frac{|S|M}{1-\gamma}+\vec{1}^\top v\right)\right|+\left\|\mathbf{S}^{-1}\mathbf{A}v-\mathbf{S}^{-1}b-\vec{1}\right\|_1+\left\|\mathbf{S}^{-1}\mathbf{A}v-\mathbf{S}^{-1}b+\vec{1}\right\|_1 \tag{56}$$

where $s=\frac{1}{2}(\frac{2M}{1-\gamma}\vec{1}-r)$ and $b=\frac{1}{2}(\frac{2M}{1-\gamma}\vec{1}+r)$, and $\mathbf{S}=\mathbf{diag}(s)$. We let v_f^* denote the optimal solution to this ℓ_1 regression problem and we call v an ϵ-optimal solution to f if $f(v)\leq f(v_f^*)+\epsilon$.

Observe that (56) is indeed an instance of the ℓ_1 regression problem from Theorem 1.2 where the input matrix and the input vector c are defined as

$$\begin{bmatrix}\mathbf{S}^{-1}\mathbf{A}\\ \mathbf{S}^{-1}\mathbf{A}\\ \alpha\vec{1}^\top\end{bmatrix}\in\mathbb{R}^{(2|S\times A|+1)\times|S|}\text{ and }c=\begin{bmatrix}\mathbf{S}^{-1}b+\vec{1}\\ \mathbf{S}^{-1}b-\vec{1}\\ \alpha\frac{|S|M}{1-\gamma}\end{bmatrix}\in\mathbb{R}^{(2|S\times A|+1)}.$$

The next reduction is stated as follows:

Lemma 8.9 (Lemma B.7 of [SWWY18])**.** *Suppose that v is an ϵ-approximate solution to the DMDP ℓ_1 problem then v is an ϵ'-approximate DMDP LP solution for*

$$\epsilon'\leq\max\left\{\frac{\epsilon}{\alpha}\ ,\ \frac{2\alpha|S|M^2}{(1-\gamma)^2}+\frac{\epsilon M}{(1-\gamma)}\right\}.$$

By combining the above two reductions and the ℓ_1-regression algorithm from Theorem 1.2, we obtain the new algorithm for solving DMDP, proving Theorem 1.3.

Proof of Theorem 1.3. To obtain an ϵ-optimal policy π, by Lemma 8.7, it suffices to compute an ε_2-approximate DMDP LP solution v where $\varepsilon_2=\frac{\epsilon(1-\gamma)^2}{8|S|}$. Let v be a ε_3-approximate solution to the DMDP ℓ_1-regression instance in (56) where $\alpha=\frac{\varepsilon_2(1-\gamma)^2}{4|S|M^2}$ and $\varepsilon_3=\min\{\alpha\varepsilon_2,\frac{\varepsilon_2(1-\gamma)}{2M}\}$. By Lemma 8.9, v is ε'-approximate DMDP LP solution where

$$\varepsilon'\leq\max\left\{\frac{\epsilon_3}{\alpha}\ ,\ \frac{2\alpha|S|M^2}{(1-\gamma)^2}+\frac{\epsilon_3 M}{(1-\gamma)}\right\}\leq\max\left\{\varepsilon_2\ ,\ \varepsilon_2/2+\varepsilon_2/2\right\}=\varepsilon_2$$

as desired. To solve the ℓ_1-regression instance in (56), we invoke the algorithm from Theorem 1.2. As $\varepsilon_3\geq\text{poly}(\frac{\varepsilon\cdot(1-\gamma)}{M|S|})$ and the input matrix has size $(2|S\times A|+1)\times|S|$ and the input vector has size $(2|S\times A|+1)$, with high probability, we obtain v in time $\widetilde{O}((|S|^2|A|+|S|^{2.5})\log(\frac{M}{(1-\gamma)\epsilon}))$. ☐

Note that the running time above is nearly linear time because the size of the input is $\Omega(|S|^2|A|)$. Note that Theorem 1.3 gives the first nearly linear time algorithm for which dependencies on $M,\frac{1}{\varepsilon},\frac{1}{(1-\gamma)}$ are all logarithmic.

Acknowledgment

This project has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme under grant agreement No 715672. Jan van den Brand is partially supported by the Google PhD Fellowship Program. Aaron Sidford is supported by a Microsoft Research Faculty Fellowship, NSF CAREER Award CCF-1844855, NSF Grant CCF-1955039, a PayPal research award, and a Sloan Research Fellowship. Yin Tat Lee is supported by NSF awards CCF-1749609, CCF-1740551, DMS-1839116, DMS-2023166, Microsoft Research Faculty Fellowship, Sloan Research Fellowship, Packard Fellowships. Di Wang did part of this work while at Georgia Tech, and was partially supported by NSF grant CCF-1718533. Yang P. Liu was supported by the Department of Defense (DoD) through the National Defense Science and Engineering Graduate Fellowship (NDSEG) Program. We sincerely thank Danupon Nanongkai and Richard Peng for helpful discussions on this project. Also, we thank Danupon Nanongkai for his help on improving the quality in this paper.

References

[AKY20] Alekh Agarwal, Sham M. Kakade, and Lin F. Yang. Model-based reinforcement learning with a generative model is minimax optimal. In *Conference on Learning Theory, COLT 2020, 9-12 July 2020, Virtual Event [Graz, Austria]*, pages 67–83, 2020.

[AMK13] Mohammad Gheshlaghi Azar, Rémi Munos, and Hilbert J Kappen. Minimax pac bounds on the sample complexity of reinforcement learning with a generative model. *Machine learning*, 91(3):325–349, 2013.

[AMV20] Kyriakos Axiotis, Aleksander Madry, and Adrian Vladu. Circulation control for faster minimum cost flow in unit-capacity graphs. In *FOCS*. https//arxiv.org/pdf/2003.04863.pdf, 2020.

[AO91] Ravindra K Ahuja and James B Orlin. Distance-directed augmenting path algorithms for maximum flow and parametric maximum flow problems. *Naval Research Logistics (NRL)*, 38(3):413–430, 1991.

[AO06] Peter Auer and Ronald Ortner. Logarithmic online regret bounds for undiscounted reinforcement learning. In Bernhard Schölkopf, John C. Platt, and Thomas Hofmann, editors, *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*, pages 49–56. MIT Press, 2006.

[AW21] Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In *SODA*. https://arxiv.org/pdf/2010.05846.pdf, 2021.

[BBN+20] Aaron Bernstein, Jan van den Brand, Danupon Nanongkai, Maximilian Probst, Thatchaphol Saranurak, Aaron Sidford, and He Sun. Fully-dynamic graph sparsifiers against an adaptive adversary. In *manuscript*. https://arxiv.org/pdf/2004.08432, 2020.

[BLN+20] Jan van den Brand, Yin-Tat Lee, Danupon Nanongkai, Richard Peng, Thatchaphol Saranurak, Aaron Sidford, Zhao Song, and Di Wang. Bipartite matching in nearly-linear time on moderately dense graphs. In *FOCS*, 2020.

[BLSS20] Jan van den Brand, Yin Tat Lee, Aaron Sidford, and Zhao Song. Solving tall dense linear programs in nearly linear time. In *STOC*. https://arxiv.org/pdf/2002.02304.pdf, 2020.

[BNS19] Jan van den Brand, Danupon Nanongkai, and Thatchaphol Saranurak. Dynamic matrix inverse: Improved algorithms and matching conditional lower bounds. In *FOCS*, pages 456–480. IEEE Computer Society, 2019.

[Bra20] Jan van den Brand. A deterministic linear program solver in current matrix multiplication time. In *SODA*, pages 259–278. SIAM, 2020.

[Bra21] Jan van den Brand. Unifying matrix data structures: Simplifying and speeding up iterative algorithms. 2021.

[CJN18] Michael B Cohen, TS Jayram, and Jelani Nelson. Simple analyses of the sparse johnson-lindenstrauss transform. In *1st Symposium on Simplicity in Algorithms (SOSA)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[CKM+14] Michael B. Cohen, Rasmus Kyng, Gary L. Miller, Jakub W. Pachocki, Richard Peng, Anup B. Rao, and Shen Chen Xu. Solving sdd linear systems in nearly $m \log^{1/2} n$ time. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*, pages 343–352, 2014.

[Cla05] Kenneth L Clarkson. Subgradient and sampling algorithms for l 1 regression. 2005.

[CLM+15] Michael B Cohen, Yin Tat Lee, Cameron Musco, Christopher Musco, Richard Peng, and Aaron Sidford. Uniform sampling for matrix approximation. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 181–190, 2015.

[CLS19] Michael B Cohen, Yin Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. In *STOC*. https://arxiv.org/pdf/1810.07896, 2019.

[CMMP13] Hui Han Chin, Aleksander Madry, Gary L. Miller, and Richard Peng. Runtime guarantees for regression problems. In Robert D. Kleinberg, editor, *Innovations in Theoretical Computer Science, ITCS '13, Berkeley, CA, USA, January 9-12, 2013*, pages 269–282. ACM, 2013.

[CMSV17] Michael B Cohen, Aleksander Madry, Piotr Sankowski, and Adrian Vladu. Negative-weight shortest paths and unit capacity minimum cost flow in $O(m^{10/7} \log W)$ time. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 752–771. SIAM, 2017.

[CP15] Michael B. Cohen and Richard Peng. ℓ_p row sampling by lewis weights. In *STOC*, pages 183–192. ACM, 2015.

[CTCG+98] Jean Cochet-Terrasson, Guy Cohen, Stéphane Gaubert, Michael McGettrick, and Jean-Pierre Quadrat. Numerical computation of spectral elements in max-plus algebra. *IFAC Proceedings Volumes*, 31(18):667–674, 1998.

[DG98] Ali Dasdan and Rajesh K. Gupta. Faster maximum and minimum mean cycle algorithms for system-performance analysis. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 17(10):889–899, 1998.

[DHNV20] Daniel Dadush, Sophie Huiberts, Bento Natura, and László A. Végh. A scaling-invariant algorithm for linear programming whose running time depends only on the constraint matrix. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 761–774. ACM, 2020.

[Din70] Efim A Dinic. Algorithm for solution of a problem of maximum flow in networks with power estimation. In *Soviet Math. Doklady*, volume 11, pages 1277–1280, 1970.

[DLS18] David Durfee, Kevin A. Lai, and Saurabh Sawlani. ℓ_1 regression using lewis weights preconditioning and stochastic gradient descent. In Sébastien Bubeck, Vianney Perchet, and Philippe Rigollet, editors, *Conference On Learning Theory, COLT 2018, Stockholm, Sweden, 6-9 July 2018*, volume 75 of *Proceedings of Machine Learning Research*, pages 1626–1656. PMLR, 2018.

[DNV20] Daniel Dadush, Bento Natura, and László A Végh. Revisiting tardos's framework for linear programming: faster exact solutions using approximate solvers. *arXiv preprint arXiv:2009.04942*, 2020.

[DS08] Samuel I Daitch and Daniel A Spielman. Faster approximate lossy generalized flow via interior point algorithms. In *Proceedings of the fortieth annual ACM symposium on Theory of computing (STOC)*, pages 451–460, 2008.

[EK72] Jack Edmonds and Richard M Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM (JACM)*, 19(2):248–264, 1972.

[Gal14] François Le Gall. Powers of tensors and fast matrix multiplication. In *ISSAC*, pages 296–303. ACM, 2014.

[GR98] Andrew V. Goldberg and Satish Rao. Beyond the flow decomposition barrier. *J. ACM*, 45(5):783–797, 1998. announced at FOCS'97.

[GRST21] Gramoz Goranci, Harald Räcke, Thatchaphol Saranurak, and Zihan Tan. The expander hierarchy and its applications to dynamic graph algorithms. pages 2212–2228, 2021.

[GT88] Zvi Galil and Éva Tardos. An $O(n^2(m+n\log n)\log n)$ min-cost flow algorithm. *J. ACM*, 35(2):374–386, 1988. announced at FOCS'86.

[GT90] Andrew V Goldberg and Robert E Tarjan. Finding minimum-cost circulations by successive approximation. *Mathematics of Operations Research*, 15(3):430–466, 1990.

[JL84] William B. Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. In *Conference in modern analysis and probability (New Haven, Conn., 1982)*, volume 26 of *Contemp. Math.*, pages 189–206. Amer. Math. Soc., Providence, RI, 1984.

[JOA10] Thomas Jaksch, Ronald Ortner, and Peter Auer. Near-optimal regret bounds for reinforcement learning. *J. Mach. Learn. Res.*, 11:1563–1600, 2010.

[JSWZ20] Shunhua Jiang, Zhao Song, Omri Weinstein, and Hengjie Zhang. Faster dynamic matrix inverse for faster lps. *arXiv preprint arXiv:2004.07470*, 2020.

[Kar84] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–396, 1984. Announced at STOC'84.

[Kat20] Tarun Kathuria. A potential reduction inspired algorithm for exact max flow in almost o(m$\hat{4}$/3) time. In *FOCS*. https://arxiv.org/pdf/2009.03260.pdf, 2020.

[KLP+16] Rasmus Kyng, Yin Tat Lee, Richard Peng, Sushant Sachdeva, and Daniel A. Spielman. Sparsified cholesky and multigrid solvers for connection laplacians. In STOC'16: Proceedings of the 48th Annual ACM Symposium on Theory of Computing, 2016.

[KMP10] Ioannis Koutis, Gary L. Miller, and Richard Peng. Approaching optimality for solving SDD systems. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 235–244, 2010.

[KMP11] Ioannis Koutis, Gary L. Miller, and Richard Peng. A nearly $m\log n$-time solver for SDD linear systems. In *Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 590–598, 2011.

[KN14] Daniel M Kane and Jelani Nelson. Sparser johnson-lindenstrauss transforms. *Journal of the ACM (JACM)*, 61(1):1–23, 2014.

[KNPW11] Daniel M Kane, Jelani Nelson, Ely Porat, and David P Woodruff. Fast moment estimation in data streams in optimal space. In *Proceedings of the forty-third annual ACM symposium on Theory of computing (STOC)*, pages 745–754, 2011.

[KOSA13] Jonathan A. Kelner, Lorenzo Orecchia, Aaron Sidford, and Zeyuan Allen Zhu. A simple, combinatorial algorithm for solving SDD systems in nearly-linear time. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 911–920, 2013.

[KS98] Michael J. Kearns and Satinder P. Singh. Finite-sample convergence rates for q-learning and indirect algorithms. In Michael J. Kearns, Sara A. Solla, and David A. Cohn, editors, *Advances in Neural Information Processing Systems 11, [NIPS Conference, Denver, Colorado, USA, November 30 - December 5, 1998]*, pages 996–1002. The MIT Press, 1998.

[KS16] 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 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 573–582, 2016.

[LDK95] Michael L. Littman, Thomas L. Dean, and Leslie Pack Kaelbling. On the complexity of solving markov decision problems. In *UAI '95: Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence, Montreal, Quebec, Canada, August 18-20, 1995*, pages 394–402, 1995.

[LMP13] Mu Li, Gary L. Miller, and Richard Peng. Iterative row sampling. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 127–136. IEEE Computer Society, 2013.

[LNNT16] Kasper Green Larsen, Jelani Nelson, Huy L Nguyen, and Mikkel Thorup. Heavy hitters via cluster-preserving clustering. In *57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 61–70. IEEE, https://arxiv.org/pdf/1604.01357, 2016.

[LPS15] Yin Tat Lee, Richard Peng, and Daniel A Spielman. Sparsified cholesky solvers for sdd linear systems. *arXiv preprint arXiv:1506.08204*, 2015.

[LS14] Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in $O(\sqrt{rank})$ iterations and faster algorithms for maximum flow. In *55th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 424–433. https://arxiv.org/pdf/1312.6677.pdf, https://arxiv.org/pdf/1312.6713.pdf, 2014.

[LS15] Yin Tat Lee and Aaron Sidford. Efficient inverse maintenance and faster algorithms for linear programming. In *FOCS*, pages 230–249. IEEE Computer Society, 2015.

[LS19] Yin Tat Lee and Aaron Sidford. Solving linear programs with $\sqrt{rank}$ linear system solves. In *arXiv preprint*. https://arxiv.org/pdf/1910.08033.pdf, 2019.

[LS20a] Yang P Liu and Aaron Sidford. Faster divergence maximization for faster maximum flow. In *FOCS*. https://arxiv.org/pdf/2003.08929.pdf, 2020.

[LS20b] Yang P Liu and Aaron Sidford. Faster energy maximization for faster maximum flow. In *STOC*. https://arxiv.org/pdf/1910.14276.pdf, 2020.

[LSZ19] Yin Tat Lee, Zhao Song, and Qiuyi Zhang. Solving empirical risk minimization in the current matrix multiplication time. In *COLT*. https://arxiv.org/pdf/1905.04447, 2019.

[LSZ20] S Cliff Liu, Zhao Song, and Hengjie Zhang. Breaking the n-pass barrier: A streaming algorithm for maximum weight bipartite matching. *arXiv preprint arXiv:2009.06106*, 2020.

[LWC+20] Gen Li, Yuting Wei, Yuejie Chi, Yuantao Gu, and Yuxin Chen. Breaking the sample size barrier in model-based reinforcement learning with a generative model. *arXiv preprint arXiv:2005.12900*, 2020.

[Mad02] Omid Madani. Polynomial value iteration algorithms for detrerminstic mdps. In *UAI*, pages 311–318, 2002.

[Mad13] Aleksander Madry. Navigating central path with electrical flows: From flows to matchings, and back. In *FOCS*, pages 253–262. IEEE Computer Society, 2013.

[Mad16] Aleksander Madry. Computing maximum flow with augmenting electrical flows. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 593–602. IEEE, 2016.

[Mah96] Sridhar Mahadevan. Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Mach. Learn.*, 22(1-3):159–195, 1996.

[MM13] Xiangrui Meng and Michael W. Mahoney. Robust regression on mapreduce. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 888–896. JMLR.org, 2013.

[MT03] Renato D. C. Monteiro and Takashi Tsuchiya. A variant of the vavasis–ye layered-step interior-point algorithm for linear programming. *SIAM J. Optim.*, 13(4):1054–1079, 2003.

[MT05] Renato D. C. Monteiro and Takashi Tsuchiya. A new iteration-complexity bound for the MTY predictor-corrector algorithm. *SIAM J. Optim.*, 15(2):319–347, 2005.

[Nes98] Yurii Nesterov. Introductory lectures on convex programming volume i: Basic course. *Lecture notes*, 3(4):5, 1998.

[Nes09] Yurii E. Nesterov. Unconstrained convex minimization in relative scale. *Math. Oper. Res.*, 34(1):180–193, 2009.

[NN91] Yurii E. Nesterov and Arkadii Nemirovskii. Acceleration and parallelization of the path-following interior point method for a linearly constrained convex quadratic problem. *SIAM J. Optim.*, 1(4):548–564, 1991.

[NN13] Jelani Nelson and Huy L Nguyên. OSNAP: Faster numerical linear algebra algorithms via sparser subspace embeddings. In *54th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 117–126. IEEE, https://arxiv.org/pdf/1211.1002, 2013.

[NS19] Vasileios Nakos and Zhao Song. Stronger l2/l2 compressed sensing; without iterating. In *STOC*. https://arxiv.org/pdf/1903.02742, 2019.

[NSW17] Danupon Nanongkai, Thatchaphol Saranurak, and Christian Wulff-Nilsen. Dynamic minimum spanning forest with subpolynomial worst-case update time. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 950–961, 2017.

[NSW19] Vasileios Nakos, Zhao Song, and Zhengyu Wang. (Nearly) Sample-optimal sparse Fourier transform in any dimension; RIPless and Filterless. In *FOCS*. https://arxiv.org/pdf/1909.11123.pdf, 2019.

[Orl84] James B Orlin. Genuinely polynomial simplex and non-simplex algorithms for the minimum cost flow problem. 1984.

[Orl93] James B Orlin. A faster strongly polynomial minimum cost flow algorithm. *Operations research*, 41(2):338–350, 1993.

[Pag13] Rasmus Pagh. Compressed matrix multiplication. *ACM Transactions on Computation Theory (TOCT)*, 5(3):1–17, 2013.

[PS14] Richard Peng and Daniel A. Spielman. An efficient parallel solver for SDD linear systems. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 333–342. ACM, 2014.

[PSW17] Eric Price, Zhao Song, and David P Woodruff. Fast regression with an ell_infty guarantee. In *44th International Colloquium on Automata, Languages, and Programming (ICALP)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[Ren88] James Renegar. A polynomial-time algorithm, based on newton's method, for linear programming. *Math. Program.*, 40(1-3):59–93, 1988.

[San04] Piotr Sankowski. Dynamic transitive closure via dynamic matrix inverse (extended abstract). In *FOCS*, pages 509–517. IEEE Computer Society, 2004.

[Sch16] Bruno Scherrer. Improved and generalized upper bounds on the complexity of policy iteration. *Mathematics of Operations Research*, 41(3):758–774, 2016.

[SS08] Daniel A. Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 563–568. ACM, 2008.

[SS11] Daniel A Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011.

[ST04] Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *STOC*, pages 81–90. ACM, 2004.

[SW19] Thatchaphol Saranurak and Di Wang. Expander decomposition and pruning: Faster, stronger, and simpler. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2616–2635, 2019.

[SWW+18] Aaron Sidford, Mengdi Wang, Xian Wu, Lin Yang, and Yinyu Ye. Near-optimal time and sample complexities for solving markov decision processes with a generative model. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pages 5192–5202, 2018.

[SWWY18] Aaron Sidford, Mengdi Wang, Xian Wu, and Yinyu Ye. Variance reduced value iteration and faster algorithms for solving markov decision processes. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 770–787, 2018.

[SY20] Zhao Song and Zheng Yu. Oblivious sketching-based central path method for solving linear programming problems. In *manuscript*. https://openreview.net/forum?id=fGiKxvF-eub, 2020.

[Tar85] Éva Tardos. A strongly polynomial minimum cost circulation algorithm. *Combinatorica*, 5(3):247–255, 1985.

[Tro11] Joel Tropp. Freedman's inequality for matrix martingales. *Electronic Communications in Probability*, 16:262–270, 2011.

[Tse90] Paul Tseng. Solving h-horizon, stationary markov decision problems in time proportional to log (h). *Operations Research Letters*, 9(5):287–297, 1990.

[Vai89] Pravin M. Vaidya. Speeding-up linear programming using fast matrix multiplication (extended abstract). In *FOCS*, pages 332–337. IEEE Computer Society, 1989.

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

[Wai19] Martin J Wainwright. Variance-reduced q-learning is minimax optimal. *arXiv preprint arXiv:1906.04697*, 2019.

[Wan17] Mengdi Wang. Randomized linear programming solves the discounted markov decision problem in nearly-linear (sometimes sublinear) running time. *arXiv preprint arXiv:1704.01869*, 2017.

[Wan20] Mengdi Wang. Randomized linear programming solves the markov decision problem in nearly linear (sometimes sublinear) time. *Mathematics of Operations Research*, 45(2):517–546, 2020.

[Wil12] Virginia Vassilevska Williams. Multiplying matrices faster than coppersmith-winograd. In *STOC*, pages 887–898. ACM, 2012.

[Wil19] David P Williamson. *Network Flow Algorithms*. Cambridge University Press, 2019.

[YCRM16] Jiyan Yang, Yinlam Chow, Christopher Ré, and Michael W. Mahoney. Weighted SGD for ℓ_p regression with randomized preconditioning. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 558–569. SIAM, 2016.

[Ye05] Yinyu Ye. A new complexity result on solving the markov decision problem. *Mathematics of Operations Research*, 30(3):733–749, 2005.

[Ye11] Yinyu Ye. The simplex and policy-iteration methods are strongly polynomial for the markov decision problem with a fixed discount rate. *Mathematics of Operations Research*, 36(4):593–603, 2011.

A IPM Proofs

A.1 Basic Analysis Tools

Lemma 4.2. *For all $\ell \le u$ the function $\phi(x) = -\log(x-\ell) - \log(u-x)$ is highly 1-self-concordant on the interval (ℓ, u).*

Proof. Note that $\phi''(x) = (u-x)^{-2} + (x-\ell)^{-2}$. For odd n, note that $\phi^{(n)}(x) = (n-1)!\left((u-x)^{-n} - (x-\ell)^{-n}\right)$, so

$$|\phi^{(n)}(x)| \le (n-1)!\min(u-x, x-\ell)^{-n} \le (n-1)!\phi''(x)^{n/2}.$$

This shows the first two items. For even $n \ge 2$, we have $\phi^{(n)}(x) = (n-1)!\left((u-x)^{-n} + (x-\ell)^{-n}\right)$, and

$$(n-1)!\left((u-x)^{-n} + (x-\ell)^{-n}\right) \le (n-1)!\left((u-x)^{-2} + (x-\ell)^{-2}\right)^{n/2} \le (n-1)!\phi''(x)^{n/2}.$$

This shows the last item. □

Lemma 4.15 (Potential change bound)**.** *Define for $u_i^{(j)} \ge 0$ and y_i*

$$w_i = \prod_{j\in[k]} (u_i^{(j)})^{c_j} \quad \text{and} \quad w_i^{(\text{new})} = \prod_{j\in[k]} (u_i^{(j)} + \delta_i^{(j)})^{c_j}$$

and

$$v_i = y_i w_i \quad \text{and} \quad v_i^{(\text{new})} = (y_i + \eta_i) w_i^{(\text{new})}$$

where $\|(\mathbf{U}^{(j)})^{-1}\delta^{(j)}\|_\infty \le \frac{1}{50(1+\|c\|_1)}$ for all $j \in [k]$, $\|v\|_\infty \le 1/50$, $\|\mathbf{W}\eta\|_\infty \le \frac{1}{50\lambda(1+\|c\|_1)}$, and

$$\|v\|_\infty \sum_{j\in[k]} |c_j| \|(\mathbf{U}^{(j)})^{-1}\delta^{(j)}\|_\infty \le \frac{1}{100\lambda}.$$

Then we have that $\|v^{(\text{new})} - v\|_\infty \le \frac{1}{20\lambda}$

$$\Psi(v^{(\text{new})}) \le \Psi(v) + \psi'(v)^\top \left(\mathbf{W}\eta + \sum_{j\in[k]} c_j \mathbf{V}(\mathbf{U}^{(j)})^{-1}\delta^{(j)}\right) \tag{13}$$

$$+ 8\|\mathbf{W}\eta\|^2_{\psi''(v)} + 8(1+\|c\|_1)\|v\|^2_\infty \sum_{j\in[k]} |c_j| \|(\mathbf{U}^{(j)})^{-1}\delta^{(j)}\|^2_{\psi''(v)} \tag{14}$$

$$+ 8\|\mathbf{W}\eta\|_{|\psi'(v)|} \sum_{j\in[k]} |c_j| \|(\mathbf{U}^{(j)})^{-1}\delta^{(j)}\|_{|\psi'(v)|} + 8(1+\|c\|_1)\|v\|_\infty \sum_{j\in[k]} |c_j| \|(\mathbf{U}^{(j)})^{-1}\delta^{(j)}\|^2_{|\psi'(v)|}. \tag{15}$$

Proof. For $0 \le t \le 1$ define

$$w_i^{(t)} = \prod_{j\in[k]} (u_i^{(j)} + t\cdot\delta_i^{(j)})^{c_j} \quad \text{and} \quad v_i^{(t)} = (y_i + t\cdot\eta_i)w_i^{(t)}.$$

We first calculate

$$\frac{\mathrm{d}}{\mathrm{d}t} w_i^{(t)} = w_i^{(t)} \sum_{j\in[k]} \frac{c_j\delta_i^{(j)}}{u_i^{(j)} + t\cdot\delta_i^{(j)}} \tag{57}$$

and

$$\frac{\mathrm{d}}{\mathrm{d}t} v_i^{(t)} = y_i \frac{\mathrm{d}}{\mathrm{d}t} w_i^{(t)} + \eta_i w_i^{(t)} = w_i^{(t)} \left(\eta_i + y_i \sum_{j\in[k]} \frac{c_j\delta_i^{(j)}}{u_i^{(j)} + t\cdot\delta_i^{(j)}}\right). \tag{58}$$

Applying (57) gives that

$$\begin{aligned} |\log(w_i^{(\mathrm{new})}) - \log(w_i)| &\le \left| \int_0^1 (w_i^{(t)})^{-1} \frac{d}{\mathrm{d}t} w_i^{(t)} \right| \mathrm{d}t \\ &\le \int_0^1 \left| \sum_{j\in[k]} \frac{c_j\delta_i^{(j)}}{u_i^{(j)} + t\cdot\delta_i^{(j)}} \right| \mathrm{d}t \\ &\le 2\sum_{j\in[k]} |c_j| \|(\mathbf{U}^{(j)})^{-1}\delta^{(j)})\|_\infty \le \frac{1}{25}. \end{aligned} \tag{59}$$

Therefore, $\|\mathbf{W}^{-1}(w^{(\mathrm{new})} - w)\|_\infty \le \exp(|\log(w_i^{(\mathrm{new})}) - \log(w_i)|) - 1 \le 2|\log(w_i^{(\mathrm{new})}) - \log(w_i)|$. Now we can use $v = \mathbf{W}y$ and (59) to get that

$$\begin{aligned} \|v^{(\mathrm{new})} - v\|_\infty &= \left\| (\mathbf{W}^{(\mathrm{new})} - \mathbf{W})(y+\eta) + \mathbf{W}\eta \right\|_\infty \\ &\le \|\mathbf{W}^{-1}(w^{(\mathrm{new})} - w)\|_\infty (\|\mathbf{W}y\|_\infty + \|\mathbf{W}\eta\|_\infty) + \|\mathbf{W}\eta\|_\infty \\ &\le \|\mathbf{W}^{-1}(w^{(\mathrm{new})} - w)\|_\infty \|v\|_\infty + \|\mathbf{W}^{-1}(w^{(\mathrm{new})} - w)\|_\infty \|\mathbf{W}\eta\|_\infty + \|\mathbf{W}\eta\|_\infty \\ &\le 2\|v\|_\infty \sum_{j\in[k]} |c_j| \|(\mathbf{U}^{(j)})^{-1}\delta^{(j)})\|_\infty + \frac{1}{25}\cdot\frac{1}{50\lambda} + \frac{1}{50\lambda} \\ &\le \frac{1}{50\lambda} + \frac{1}{2500\lambda} + \frac{1}{50\lambda} \le \frac{1}{20\lambda} \end{aligned}$$

as desired. We now proceed to controlling the potential Ψ. To this end, define $f(t) = \psi(v_i^{(t)})$. By Taylor's theorem we know that there is a $\zeta \in [0,1]$ so that $f(1) = f(0) + f'(0) + \frac{1}{2}f''(\zeta)$. Now compute using (58) that

$$f'(t) = \psi'(v_i^{(t)}) \frac{\mathrm{d}}{\mathrm{d}t} v_i^{(t)} = \psi'(v_i^{(t)}) w_i^{(t)} \left(\eta_i + y_i \sum_{j\in[k]} \frac{c_j\delta_i^{(j)}}{u_i^{(j)} + t\cdot\delta_i^{(j)}}\right)$$

and

$$\begin{aligned} f''(t) = \psi''(v_i^{(t)}) \frac{\mathrm{d}}{\mathrm{d}t} v_i^{(t)} w_i^{(t)} &\left(\eta_i + y_i \sum_{j\in[k]} \frac{c_j\delta_i^{(j)}}{u_i^{(j)} + t\cdot\delta_i^{(j)}}\right) \\ + \psi'(v_i^{(t)}) \frac{\mathrm{d}}{\mathrm{d}t} w_i^{(t)} &\left(\eta_i + y_i \sum_{j\in[k]} \frac{c_j\delta_i^{(j)}}{u_i^{(j)} + t\cdot\delta_i^{(j)}}\right) - \psi'(v_i^{(t)}) w_i^{(t)} y_i \sum_{j\in[k]} \frac{c_j(\delta_i^{(j)})^2}{(u_i^{(j)} + t\cdot\delta_i^{(j)})^2} \end{aligned}$$

$$= \psi''(v_i^{(t)})(w_i^{(t)})^2 \left(\eta_i + y_i \sum_{j\in[k]} \frac{c_j \delta_i^{(j)}}{u_i^{(j)} + t\cdot\delta_i^{(j)}}\right)^2 \tag{60}$$

$$+ \psi'(v_i^{(t)}) w_i^{(t)} \left(\sum_{j\in[k]} \frac{c_j \delta_i^{(j)}}{u_i^{(j)} + t\cdot\delta_i^{(j)}}\right)\left(\eta_i + y_i \sum_{j\in[k]} \frac{c_j \delta_i^{(j)}}{u_i^{(j)} + t\cdot\delta_i^{(j)}}\right) \tag{61}$$

$$- \psi'(v_i^{(t)}) w_i^{(t)} y_i \sum_{j\in[k]} \frac{c_j (\delta_i^{(j)})^2}{(u_i^{(j)} + t\cdot\delta_i^{(j)})^2}. \tag{62}$$

We now bound (60), (61), (62). We will heavily use Lemma 4.14 and the fact that $|w_i^{(t)} y_i| \le 2|w_i y_i| = 2|v_i| \le 2\|v\|_\infty \le 1/10$. To bound (60) use Cauchy-Schwarz to get that

$$\begin{aligned}
&\left|\psi''(v_i^{(t)})(w_i^{(t)})^2 \left(\eta_i + y_i \sum_{j\in[k]} \frac{c_j \delta_i^{(j)}}{u_i^{(j)} + t\cdot\delta_i^{(j)}}\right)^2\right| \\
&\le 2\psi''(v_i^{(t)})(w_i^{(t)}\eta_i)^2 + 2\psi''(v_i^{(t)})(w_i^{(t)} y_i)^2 \left(\sum_{j\in[k]} \frac{c_j \delta_i^{(j)}}{u_i^{(j)} + t\cdot\delta_i^{(j)}}\right)^2 \\
&\le 8\psi''(v_i)(w_i\eta_i)^2 + 8\psi''(v_i)\|c\|_1\|v\|_\infty^2 \sum_{j\in[k]} |c_j|((u_i^{(j)})^{-1}\delta_i^{(j)})^2.
\end{aligned}$$

To bound (61) we compute that

$$\begin{aligned}
&\left|\psi'(v_i^{(t)}) w_i^{(t)} \left(\sum_{j\in[k]} \frac{c_j \delta_i^{(j)}}{u_i^{(j)} + t\cdot\delta_i^{(j)}}\right)\left(\eta_i + y_i \sum_{j\in[k]} \frac{c_j \delta_i^{(j)}}{u_i^{(j)} + t\cdot\delta_i^{(j)}}\right)\right| \\
&\le 8|\psi'(v_i)|w_i|\eta_i| \sum_{j\in[k]} |c_j||(u_i^{(j)})^{-1}\delta_i^{(j)}| + 4|\psi'(v_i)||w_i^{(t)} y_i| \left(\sum_{j\in[k]} \frac{c_j \delta_i^{(j)}}{u_i^{(j)} + t\cdot\delta_i^{(j)}}\right)^2 \\
&\le 8|\psi'(v_i)|w_i|\eta_i| \sum_{j\in[k]} |c_j||(u_i^{(j)})^{-1}\delta_i^{(j)}| + 8|\psi'(v_i)|\|c\|_1\|v\|_\infty \sum_{j\in[k]} |c_j|((u_i^{(j)})^{-1}\delta_i^{(j)})^2
\end{aligned}$$

To bound (62) we compute that

$$\begin{aligned}
\left|\psi'(v_i^{(t)}) w_i^{(t)} y_i \sum_{j\in[k]} \frac{c_j (\delta_i^{(j)})^2}{(u_i^{(j)} + t\cdot\delta_i^{(j)})^2}\right| &\le 2|\psi'(v_i)||w_i^{(t)} y_i| \sum_{j\in[k]} \frac{|c_j|(\delta_i^{(j)})^2}{(u_i^{(j)} + t\cdot\delta_i^{(j)})^2} \\
&\le 4|\psi'(v_i)|\|v\|_\infty \sum_{j\in[k]} |c_j|((u_i^{(j)})^{-1}\delta_i^{(j)})^2.
\end{aligned}$$

Note that

$$f'(0) = \psi'(v_i) w_i \eta_i + \psi'(v_i) v_i \sum_{j\in[k]} c_j (u_i^{(j)})^{-1}\delta_i^{(j)}.$$

Summing the previous bounds over all i and using that $f(1) \le f(0) + f'(0) + \frac{1}{2}\max_{\zeta\in[0,1]} f''(\zeta)$ gives us that

$$\begin{aligned}
\Psi(v^{(\text{new})}) &\le \Psi(v^{(\text{new})}) + \psi'(v)^\top \left(\mathbf{W}\eta + \mathbf{V}\sum_{j\in[k]} c_j (\mathbf{U}^{(j)})^{-1}\delta^{(j)}\right) \\
&+ 8\sum_{i=1}^m \psi''(v_i)(w_i\eta_i)^2 + 8\|c\|_1\|v\|_\infty^2 \sum_{i=1}^m \psi''(v_i) \sum_{j\in[k]} |c_j|((u_i^{(j)})^{-1}\delta_i^{(j)})^2
\end{aligned}$$

$$
\begin{aligned}
&+ 8\sum_{i=1}^{m} \psi''(v_i) w_i |\eta_i| \sum_{j\in[k]} |c_j| |(u_i^{(j)})^{-1}\delta_i^{(j)}| + 8(1+\|c\|_1)\|v\|_\infty \sum_{i=1}^{m} |\psi'(v_i)| \sum_{j\in[k]} |c_j| ((u_i^{(j)})^{-1}\delta_i^{(j)})^2 \\
&= \Psi(v^{(\text{new})}) + \psi'(v)^\top \left(\mathbf{W}\eta + \mathbf{V} \sum_{j\in[k]} c_j (\mathbf{U}^{(j)})^{-1}\delta^{(j)} \right) \\
&+ 8\|\mathbf{W}\eta\|_{\psi''(v)}^2 + 8\|c\|_1 \|v\|_\infty^2 \sum_{j\in[k]} |c_j| \|(\mathbf{U}^{(j)})^{-1}\delta^{(j)}\|_{\psi''(v)}^2 \\
&+ 8\left\langle \mathbf{W}|\eta|, \sum_{j\in[k]} |c_j| |(\mathbf{U}^{(j)})^{-1}| |\delta^{(j)}| \right\rangle_{|\psi'(v)|} + 8(1+\|c\|_1)\|v\|_\infty \sum_j |c_j| \|(\mathbf{U}^{(j)})^{-1}\delta^{(j)}\|_{|\psi'(v)|} \\
&\le \Psi(v^{(\text{new})}) + \psi'(v)^\top \left(\mathbf{W}\eta + \mathbf{V} \sum_{j\in[k]} c_j (\mathbf{U}^{(j)})^{-1}\delta^{(j)} \right) \\
&+ 8\|\mathbf{W}\eta\|_{\psi''(v)}^2 + 8(1+\|c\|_1)\|v\|_\infty^2 \sum_{j\in[k]} |c_j| \|(\mathbf{U}^{(j)})^{-1}\delta^{(j)}\|_{\psi''(v)}^2 \\
&+ 8\|\mathbf{W}\eta\|_{|\psi'(v)|} \sum_j |c_j| \|(\mathbf{U}^{(j)})^{-1}\delta^{(j)}\|_{|\psi'(v)|} + 8(1+\|c\|_1)\|v\|_\infty \sum_{j\in[k]} |c_j| \|(\mathbf{U}^{(j)})^{-1}\delta^{(j)}\|_{|\psi'(v)|}^2
\end{aligned}
$$

by the Cauchy-Schwarz inequality as desired. □

A.2 Leverage Scores and Fundamental Matrix Proofs

The analysis and notation throughout this section is broadly based on that of [LS14].

Lemma 4.18 (Alternate definition of regularized Lewis weights)**.** *For all non-negative c*

$$w(c) = \operatorname{argmin}_{w\in\mathbb{R}^m_{>0}} f(w,c)$$

where

$$f(w,c) \stackrel{\text{def}}{=} -\frac{1}{1-\frac{2}{p}} \log\det(\mathbf{A}^\top \mathbf{C}\mathbf{W}^{1-\frac{2}{p}}\mathbf{C}\mathbf{A}) + \sum_{i=1}^{m} w_i - \sum_{i=1}^{m} v_i \log w_i.$$

Proof. By [LS19, Lemma 23] we can compute that

$$\nabla_w f(w,c) = -\mathbf{W}^{-1}\sigma(\mathbf{W}^{\frac{1}{2}-\frac{1}{p}}\mathbf{C}\mathbf{A}) + \vec{1} - \mathbf{W}^{-1}v.$$

Therefore, we have that

$$-\mathbf{W}_c^{-1}\sigma(\mathbf{W}_c^{\frac{1}{2}-\frac{1}{p}}\mathbf{C}\mathbf{A}) + \vec{1} - \mathbf{W}_c^{-1}v = 0,$$

which is equivalent to the desired bound of $w(c) = \sigma(\mathbf{W}_c^{\frac{1}{2}-\frac{1}{p}}\mathbf{C}\mathbf{A}) + v$. □

Lemma 4.19 (Jacobian of Regularized Lewis weight)**.** *For a fixed vector v, we have that*

$$\mathbf{J}_c = 2\mathbf{W}_c \left(\mathbf{W}_c - \left(1-\frac{2}{p}\right)\mathbf{\Lambda}_c \right)^{-1} \mathbf{\Lambda}_c \mathbf{C}^{-1}.$$

Proof. By Lemma 4.18 we have that

$$\nabla_w f(w(c), c) = 0.$$

Differentiating with respect to c gives us

$$\nabla_{wc}^2 f(w(c),c) + \nabla_{ww}^2 f(w(c),c)\mathbf{J}_c = 0.$$

As in the proof of [LS19, Lemma 24] we have that

$$\nabla^2_{wc} f(w,c) = -2\mathbf{W}^{-1}\mathbf{\Lambda}_c\mathbf{C}^{-1}$$

and

$$\begin{aligned}\nabla^2_{ww} f(w,c) &= \mathbf{W}^{-1}\left(\mathbf{\Sigma}_c - \left(1-\frac{2}{p}\right)\mathbf{\Lambda}_c\right)\mathbf{W}^{-1} + \mathbf{W}^{-1}\mathbf{V}\mathbf{W}^{-1} \\ &= \mathbf{W}^{-1}\left(\mathbf{\Sigma}_c + \mathbf{V} - \left(1-\frac{2}{p}\right)\mathbf{\Lambda}_c\right)\mathbf{W}^{-1}.\end{aligned}$$

Now we can solve to get

$$\begin{aligned}\mathbf{J}_c &= 2\mathbf{W}_c\left(\mathbf{\Sigma}_c + \mathbf{V} - \left(1-\frac{2}{p}\right)\mathbf{\Lambda}_c\right)^{-1}\mathbf{\Lambda}_c\mathbf{C}^{-1} \\ &= 2\mathbf{W}_c\left(\mathbf{W}_c - \left(1-\frac{2}{p}\right)\mathbf{\Lambda}_c\right)^{-1}\mathbf{\Lambda}_c\mathbf{C}^{-1}\end{aligned}$$

where we have used that $\mathbf{W}_c = \mathbf{\Sigma}_c + \mathbf{V}$ by definition. □

Before continuing, we collect various properties of projection matrices, proven in [LS19, Lemma 47].

Lemma A.1 (Facts related to $\mathbf{T}^{-1}\mathbf{P}^{(2)}$)**.** *We have the following for a vector $h \in \mathbb{R}^m$, where $\mathbf{P}$ is an orthogonal projection matrix and $\mathbf{T}$ is a diagonal matrix with the same diagonal as $\mathbf{P}$.*

1. $\mathbf{P}^{(2)} \preceq \mathbf{T}$.
2. $\|\mathbf{T}^{-1}\mathbf{P}^{(2)}\|_\infty \le 1$.
3. $\|\mathbf{T}^{-1}\mathbf{P}^{(2)}h\|_\tau \le \|h\|_{\mathbf{P}^{(2)}} \le \|h\|_\tau$.
4. $\|\mathbf{T}^{-1}\mathbf{P}^{(2)}h\|_\infty \le \|h\|_\tau$.

We now show that the operator $\mathbf{W}_c^{-1}\mathbf{J}_c\mathbf{C}$ is bounded in the $\tau+\infty$ norm.

Lemma A.2 (Facts related to $\mathbf{W}_c^{-1}\mathbf{J}_c\mathbf{C}$)**.** *Define $\|g\|_{w(c)+\infty} \stackrel{\text{def}}{=} \|g\|_\infty + C_{\text{norm}}\|g\|_{w(c)}$. If $p \in [2/3, 1]$ then for all vectors h we have*

- $\|\mathbf{W}_c^{-1}\mathbf{J}_c\mathbf{C}h\|_{w(c)} \le p\|h\|_{w(c)}$.
- $\|\mathbf{W}_c^{-1}\mathbf{J}_c\mathbf{C}h\|_\infty \le p\|h\|_\infty + 2\|h\|_{w(c)}$.
- $\|\mathbf{W}_c^{-1}\mathbf{J}_c\mathbf{C}h\|_{w(c)+\infty} \le p(1+3/C_{\text{norm}})\|h\|_{w(c)+\infty}$.

Proof. Define $\overline{\mathbf{\Lambda}}_c = \mathbf{W}_c^{-\frac{1}{2}}\mathbf{\Lambda}_c\mathbf{W}_c^{-\frac{1}{2}}$. Compute that

$$\begin{aligned}\mathbf{W}_c^{-\frac{1}{2}}\mathbf{J}_c\mathbf{C} &= 2\mathbf{W}_c^{\frac{1}{2}}\left(\mathbf{W}_c - \left(1-\frac{2}{p}\right)\mathbf{\Lambda}_c\right)^{-1}\mathbf{\Lambda}_c \\ &= 2\left(\mathbf{I} - \left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}}_c\right)^{-1}\mathbf{W}_c^{-\frac{1}{2}}\mathbf{\Lambda}_c \\ &= 2\left(\mathbf{I} - \left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}}_c\right)^{-1}\overline{\mathbf{\Lambda}}_c\mathbf{W}_c^{\frac{1}{2}}.\end{aligned}$$

Recall that $\mathbf{\Lambda}_c \preceq \mathbf{\Sigma}_c \preceq \mathbf{W}_c$, so $0 \preceq \overline{\mathbf{\Lambda}}_c \preceq \mathbf{I}$. Therefore $\left(\mathbf{I} - \left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}}_c\right)^{-1}\overline{\mathbf{\Lambda}}_c$ is a positive semidefinite matrix with eigenvalues at most

$$\max_{0\le\lambda\le 1}\frac{\lambda}{1-\left(1-\frac{2}{p}\right)\lambda} = \frac{p}{2}.$$

Thus, we have

$$\|\mathbf{W}_c^{-1}\mathbf{J}_c\mathbf{C}h\|_{w(c)} = \|\mathbf{W}_c^{-\frac{1}{2}}\mathbf{J}_c\mathbf{C}h\|_2 \\ \leq 2\left\|\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}}_c\right)^{-1}\overline{\mathbf{\Lambda}}_c\right\|_2 \|\mathbf{W}_c^{\frac{1}{2}}h\|_2 \leq p\|h\|_{w(c)}.$$

Define $\mathbf{S}_c = \mathbf{W}_c^{-\frac{1}{2}}\mathbf{\Sigma}_c\mathbf{W}_c^{-\frac{1}{2}}$. Note that $\mathbf{S}_c$ is diagonal and $0 \preceq \mathbf{S}_c \preceq \mathbf{I}$. Define $\mathbf{Q}_c = \mathbf{W}_c^{-\frac{1}{2}}\mathbf{P}_c^{(2)}\mathbf{W}_c^{-\frac{1}{2}}$. By definition, $\overline{\mathbf{\Lambda}_c} = \mathbf{S}_c - \mathbf{Q}_c$. Define $\mathbf{D}_c = \frac{2\mathbf{S}_c}{\mathbf{I}+\left(\frac{2}{p}-1\right)\mathbf{S}_c}$, and note that $\mathbf{D}_c$ is diagonal and $0 \preceq \mathbf{D}_c \preceq p\mathbf{I}$. Note from the above formula that

$$\mathbf{W}_c^{-1}\mathbf{J}_c\mathbf{C}h - \mathbf{D}_c h = 2\mathbf{W}_c^{-\frac{1}{2}}\overline{\mathbf{\Lambda}}_c\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}}_c\right)^{-1}\mathbf{W}_c^{\frac{1}{2}}h - \mathbf{W}_c^{-\frac{1}{2}}\mathbf{D}_c\mathbf{W}_c^{\frac{1}{2}}h \tag{63}$$

$$= \mathbf{W}_c^{-\frac{1}{2}}\left(2\overline{\mathbf{\Lambda}}_c - \mathbf{D}_c\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}}_c\right)\right)\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}}_c\right)^{-1}\mathbf{W}_c^{\frac{1}{2}}h$$

$$= \mathbf{W}_c^{-\frac{1}{2}}\left(2\mathbf{S}_c - 2\mathbf{Q}_c - \frac{2\mathbf{S}_c}{\mathbf{I}+\left(\frac{2}{p}-1\right)\mathbf{S}_c}\left(\mathbf{I}-\left(1-\frac{2}{p}\right)(\mathbf{S}_c-\mathbf{Q}_c)\right)\right)\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}}_c\right)^{-1}\mathbf{W}_c^{\frac{1}{2}}h$$

$$= -2\mathbf{W}_c^{-\frac{1}{2}}\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\mathbf{S}_c\right)^{-1}\mathbf{Q}_c\left(\mathbf{I}\left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}}_c\right)^{-1}\mathbf{W}_c^{\frac{1}{2}}h$$

$$= -2\mathbf{W}_c^{-\frac{1}{2}}\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\mathbf{S}_c\right)^{-1}\mathbf{W}_c^{-\frac{1}{2}}\mathbf{P}_c^{(2)}\mathbf{W}_c^{-\frac{1}{2}}\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}}_c\right)^{-1}\mathbf{W}_c^{\frac{1}{2}}h. \tag{64}$$

By Lemma A.1 and $\mathbf{\Sigma}_c \preceq \mathbf{W}_c$ and both are diagonal matrices, we have that

$$\|\mathbf{W}_c^{-1}\mathbf{P}_c^{(2)}x\|_\infty \leq \|x\|_{\mathbf{\Sigma}_c} \leq \|x\|_{w(c)}.$$

Also, $\mathbf{S}_c$ is non-negative and diagonal. Therefore,

$$\begin{aligned}
&\|\mathbf{W}_c^{-1}\mathbf{J}_c\mathbf{C}h\|_\infty \\
&\leq \|\mathbf{D}_c h\|_\infty + \|2\mathbf{W}_c^{-\frac{1}{2}}\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\mathbf{S}_c\right)^{-1}\mathbf{W}_c^{-\frac{1}{2}}\mathbf{P}_c^{(2)}\mathbf{W}_c^{-\frac{1}{2}}\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}}_c\right)^{-1}\mathbf{W}_c^{\frac{1}{2}}h\|_\infty \\
&\leq \|\mathbf{D}_c h\|_\infty + \|2\mathbf{W}_c^{-1}\mathbf{P}_c^{(2)}\mathbf{W}_c^{-\frac{1}{2}}\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}}_c\right)^{-1}\mathbf{W}_c^{\frac{1}{2}}h\|_\infty \\
&\leq p\|h\|_\infty + 2\|\mathbf{W}_c^{-\frac{1}{2}}\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}}_c\right)^{-1}\mathbf{W}_c^{\frac{1}{2}}h\|_{w(c)} \\
&\leq p\|h\|_\infty + 2\left\|\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}}_c\right)^{-1}\right\|_2 \|\mathbf{W}_c^{\frac{1}{2}}h\|_2 \\
&\leq p\|h\|_\infty + 2\|h\|_{w(c)}.
\end{aligned}$$

At the end we have used that $\overline{\mathbf{\Lambda}}_c$ is a positive semidefinite matrix so $\left(\mathbf{I}+\left(\frac{2}{p}-1\right)\overline{\mathbf{\Lambda}}_c\right)$ has all eigenvalues at least 1, hence $0 \preceq \left(\mathbf{I}+\left(\frac{2}{p}-1\right)\overline{\mathbf{\Lambda}}_c\right)^{-1} \preceq \mathbf{I}$.

Finally, note that

$$\begin{aligned}
\|\mathbf{W}_c^{-1}\mathbf{J}_c\mathbf{C}h\|_{w(c)+\infty} &\leq p\|h\|_\infty + 2\|h\|_{w(c)} + C_{\text{norm}}p\|h\|_{w(c)} \\
&\leq p\left(1+2/(C_{\text{norm}}p)\right)\|h\|_{w(c)+\infty} \\
&\leq p(1+3/C_{\text{norm}})\|h\|_{w(c)+\infty}
\end{aligned}$$

as $p \geq 2/3$. □

We will need one additional bound on the ∞ norm of a matrix that appears in the expression for the Jacobian in Lemma 4.19.

Lemma A.3 (Matrix ∞-norm bound)**.** *In the notation of Lemma A.2 we have that for any vector h that*

$$\left\| \mathbf{W}_c^{-\frac{1}{2}} \left(\mathbf{I} - \left(1 - \frac{2}{p}\right) \overline{\mathbf{\Lambda}}_c \right)^{-1} \mathbf{W}_c^{\frac{1}{2}} h \right\|_\infty \leq 3 \|h\|_\infty.$$

Proof. By Lemma A.1 we know $\|\mathbf{\Sigma}_c^{-1} \mathbf{P}_c^{(2)}\|_\infty \leq 1$. Define $\mathbf{M}_c = \frac{p}{2}\left(\mathbf{W}_c + (\frac{2}{p} - 1)\mathbf{\Sigma}_c\right) \succeq \mathbf{\Sigma}_c$. Let $\ell = \frac{p}{2}\left(\frac{2}{p} - 1\right)$. We calculate

$$\begin{aligned}
\left\| \mathbf{W}_c^{-\frac{1}{2}} \left(\mathbf{I} - \left(1 - \frac{2}{p}\right) \overline{\mathbf{\Lambda}}_c \right)^{-1} \mathbf{W}_c^{\frac{1}{2}} \right\|_\infty &= \left\| \left(\mathbf{W}_c - \left(1 - \frac{2}{p}\right) \mathbf{\Lambda}_c \right)^{-1} \mathbf{W}_c \right\|_\infty \\
&= \left\| \left(\frac{2}{p} \mathbf{M}_c - \left(\frac{2}{p} - 1\right) \mathbf{P}_c^{(2)} \right)^{-1} \mathbf{W}_c \right\|_\infty \\
&= \frac{p}{2} \left\| \mathbf{M}_c^{-\frac{1}{2}} \left(\mathbf{I} - \ell \mathbf{M}_c^{-\frac{1}{2}} \mathbf{P}_c^{(2)} \mathbf{M}_c^{\frac{1}{2}} \right)^{-1} \mathbf{M}_c^{-\frac{1}{2}} \mathbf{W}_c \right\|_\infty \\
&= \frac{p}{2} \left\| \mathbf{M}_c^{-\frac{1}{2}} \sum_{i \geq 0} \ell^i (\mathbf{M}_c^{-\frac{1}{2}} \mathbf{P}_c^{(2)} \mathbf{M}_c^{-\frac{1}{2}})^i \mathbf{M}_c^{-\frac{1}{2}} \mathbf{W}_c \right\|_\infty \\
&= \frac{p}{2} \left\| \sum_{i \geq 0} \ell^i (\mathbf{M}_c^{-1} \mathbf{P}_c^{(2)})^i \mathbf{M}_c^{-1} \mathbf{W}_c \right\|_\infty \\
&\leq \frac{p}{2} \sum_{i \geq 0} \ell^i \left\| \mathbf{\Sigma}_c^{-1} \mathbf{P}_c^{(2)} \right\|_\infty^i \left\| \mathbf{M}^{-1} \mathbf{W}_c \right\|_\infty \\
&\leq \left\| \mathbf{M}_c^{-1} \mathbf{W}_c \right\|_\infty \leq \frac{2}{p} \leq 3
\end{aligned}$$

where we have used that if $\mathbf{X}$ is a symmetric matrix with $\|\mathbf{X}\|_2 < 1$ then $(\mathbf{I} - \mathbf{X})^{-1} = \sum_{i \geq 0} \mathbf{X}^i$, and $\mathbf{M}_c \succeq \frac{p}{2} \mathbf{W}_c$ as diagonal matrices. ☐

Lemma 4.20 (Decomposition of $\mathbf{J}_c$)**.** *For any vector $c \in \mathbb{R}^m_{>0}$, there is a diagonal matrix $\mathbf{0} \preceq \mathbf{D}_c \preceq \mathbf{I}$ such that for $\mathbf{K}_c = \mathbf{W}_c^{-1} \mathbf{J}_c \mathbf{C} - \mathbf{D}_c$, we have for all vectors h that*

- $\|\mathbf{K}_c h\|_\infty \lesssim \|h\|_\infty$.
- $\|\mathbf{K}_c h\|_{w(c)} \lesssim |||h|||_{\mathbf{P}_c^{(2)}}$.

Proof. We follow the notation of the proof of Lemma A.2. We then have by (63) and (64) that

$$\mathbf{K}_c = \mathbf{W}_c^{-1} \mathbf{J}_c \mathbf{C} - \mathbf{D}_c = 2 \mathbf{W}_c^{-\frac{1}{2}} \left(\mathbf{I} - \left(1 - \frac{2}{p}\right) \mathbf{S}_c \right)^{-1} \mathbf{W}_c^{-\frac{1}{2}} \mathbf{P}_c^{(2)} \mathbf{W}_c^{-\frac{1}{2}} \left(\mathbf{I} - \left(1 - \frac{2}{p}\right) \overline{\mathbf{\Lambda}}_c \right)^{-1} \mathbf{W}_c^{\frac{1}{2}}.$$

We first bound the ∞-norm of $\mathbf{K}_c$. By Lemma A.1 and Lemma A.3 we get

$$\begin{aligned}
&2 \left\| \mathbf{W}_c^{-\frac{1}{2}} \left(\mathbf{I} - \left(1 - \frac{2}{p}\right) \mathbf{S}_c \right)^{-1} \mathbf{W}_c^{-\frac{1}{2}} \mathbf{P}_c^{(2)} \mathbf{W}_c^{-\frac{1}{2}} \left(\mathbf{I} - \left(1 - \frac{2}{p}\right) \overline{\mathbf{\Lambda}}_c \right)^{-1} \mathbf{W}_c^{\frac{1}{2}} \right\|_\infty \\
&\leq 2 \| \mathbf{W}_c^{-1} \mathbf{P}_c^{(2)} \mathbf{W}_c^{-\frac{1}{2}} \left(\mathbf{I} - \left(1 - \frac{2}{p}\right) \overline{\mathbf{\Lambda}}_c \right)^{-1} \mathbf{W}_c^{\frac{1}{2}} \|_\infty \lesssim 1.
\end{aligned}$$

For the τ-norm, we use that $\mathbf{W}_c \succeq \mathbf{P}_c^{(2)}$ and bound

$$\begin{aligned}
&2\left\|\mathbf{W}_c^{-\frac{1}{2}}\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\mathbf{S}_c\right)^{-1}\mathbf{W}_c^{-\frac{1}{2}}\mathbf{P}_c^{(2)}\mathbf{W}_c^{-\frac{1}{2}}\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}}_c\right)^{-1}\mathbf{W}_c^{\frac{1}{2}}h\right\|_\tau \\
&=2\left\|\mathbf{W}_c^{-\frac{1}{2}}\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}}_c\right)^{-1}\mathbf{W}_c^{-\frac{1}{2}}\mathbf{P}_c^{(2)}\mathbf{W}_c^{-\frac{1}{2}}\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\mathbf{S}_c\right)^{-1}\mathbf{W}_c^{\frac{1}{2}}h\right\|_\tau \\
&\le 2\left\|\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}}_c\right)^{-1}\mathbf{W}_c^{-\frac{1}{2}}\mathbf{P}_c^{(2)}\mathbf{W}_c^{-\frac{1}{2}}\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\mathbf{S}_c\right)^{-1}\mathbf{W}_c^{\frac{1}{2}}h\right\|_2 \\
&\le 2\left\|\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\mathbf{S}_c\right)^{-1}h\right\|_{\mathbf{P}_c^{(2)}\mathbf{W}_c^{-1}\mathbf{P}_c^{(2)}} \\
&\le 2\left\|\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\mathbf{S}_c\right)^{-1}h\right\|_{\mathbf{P}_c^{(2)}} \le 2\left|\left|\left|h\right|\right|\right|_{\mathbf{P}_c^{(2)}}.
\end{aligned}$$

In the third line (the equality) we have used that $\mathbf{W}_c^{\frac{1}{2}}\mathbf{K}_c\mathbf{W}_c^{-\frac{1}{2}}$ is a symmetric matrix, as both $\mathbf{W}_c^{-\frac{1}{2}}\mathbf{J}_c\mathbf{C}\mathbf{W}_c^{-\frac{1}{2}}$ and $\mathbf{D}_c$ are, so it equals its transpose. □

Lemma 4.21 (Alternate decomposition)**.** *In the notation of Lemma 4.20, there is a diagonal matrix $\mathbf{D}'_c$ and matrix $\mathbf{K}'_c$ such that $\mathbf{0} \preceq \mathbf{D}'_c \preceq \mathbf{I}$,*

$$\mathbf{W}_c^{-\frac{1}{2}}\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}}_c\right)^{-1}\mathbf{W}_c^{\frac{1}{2}} = \mathbf{D}'_c + \mathbf{K}'_c,$$

and for all vectors h,

- $\|\mathbf{K}'_c h\|_{w(c)} \lesssim \left|\left|\left|h\right|\right|\right|_{\mathbf{P}_c^{(2)}}$
- $\|\mathbf{K}'_c h\|_\infty \lesssim \|h\|_\infty$.

Proof. By Lemma 4.20 we can write

$$\begin{aligned}
\mathbf{W}_c^{-\frac{1}{2}}\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}}_c\right)^{-1}\mathbf{W}_c^{\frac{1}{2}} &= \mathbf{I}+\left(1-\frac{2}{p}\right)\mathbf{W}_c^{-\frac{1}{2}}\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}}_c\right)^{-1}\overline{\mathbf{\Lambda}}_c\mathbf{W}_c^{\frac{1}{2}} \\
&= \mathbf{I}+\left(\frac{1}{2}-\frac{1}{p}\right)\mathbf{W}_c^{-1}\mathbf{J}_c\mathbf{C} \\
&= \mathbf{I}+\left(\frac{1}{2}-\frac{1}{p}\right)\mathbf{D}_c+\left(\frac{1}{2}-\frac{1}{p}\right)\mathbf{K}_c.
\end{aligned}$$

We set $\mathbf{D}'_c = \mathbf{I}+\left(1-\frac{2}{p}\right)\mathbf{D}_c$ and $\mathbf{K}'_c = \left(\frac{1}{2}-\frac{1}{p}\right)\mathbf{K}_c$. This immediately implies the two claims about the τ and ∞ norms of $\mathbf{K}'_c h$. Finally, as in the proof of Lemma A.2 we know that

$$\mathbf{I}+\left(\frac{1}{2}-\frac{1}{p}\right)\mathbf{D}_c = \mathbf{I}+\left(1-\frac{2}{p}\right)\frac{\mathbf{S}_c}{\mathbf{I}+\left(\frac{2}{p}-1\right)\mathbf{S}_c} = \left(\mathbf{I}+\left(\frac{2}{p}-1\right)\mathbf{S}_c\right)^{-1}.$$

Therefore, $\mathbf{D}'_c$ is diagonal and $0 \preceq \mathbf{D}'_c \preceq \mathbf{I}$ as desired. □

Lemma 4.22 (Lewis weight approximation)**.** *Let $p \in (0,4)$. If $\overline{\mathbf{C}} \approx_\varepsilon \mathbf{C}$ then $w_p(\mathbf{C}\mathbf{A}) \approx_{4\varepsilon} w_p(\overline{\mathbf{C}}\mathbf{A})$.*

Proof. Define $w_0 = w_p(\mathbf{C}\mathbf{A})$, and

$$w_{i+1} = \mathrm{Iter}(w_i, \overline{\mathbf{C}}) \quad \text{for} \quad \mathrm{Iter}(u,\mathbf{C})_k \stackrel{\text{def}}{=} \left(c_k^2 a_k^\top (\mathbf{A}^\top \mathbf{C}\mathbf{U}^{1-\frac{2}{p}}\mathbf{C}\mathbf{A})^{-1}a_k + u_k^{\frac{2}{p}-1}v_k\right)^{\frac{p}{2}}.$$

Note that if $\overline{\mathbf{C}} \approx_\varepsilon \mathbf{C}$ then $\mathrm{Iter}(u, \mathbf{C}) \approx_{2p\varepsilon} \mathrm{Iter}(u, \overline{\mathbf{C}})$ for all vectors u. Also, if $u \approx_\varepsilon u'$ then $\mathrm{Iter}(u, \mathbf{C}) \approx_{(1-\frac{p}{2})\varepsilon} \mathrm{Iter}(u', \mathbf{C})$ for all diagonal matrices $\mathbf{C}$. Now

$$w_0 = \mathrm{Iter}(w_0, \mathbf{C}) \approx_{2p\varepsilon} \mathrm{Iter}(w_0, \overline{\mathbf{C}}) = w_1.$$

If $w_i \approx_\delta w_{i+1}$ then

$$w_{i+1} = \mathrm{Iter}(w_i, \overline{\mathbf{C}}) \approx_{(1-\frac{p}{2})\delta} \mathrm{Iter}(w_{i+1}, \overline{\mathbf{C}}) = w_{i+2}.$$

Therefore

$$w_0 \approx_{\sum_{i\geq 0} 2(1-\frac{p}{2})^i p\varepsilon} \lim_{k\to\infty} w_k = w_p(\overline{\mathbf{C}}\mathbf{A})$$

and

$$2\sum_{i\geq 0}\left(1-\frac{p}{2}\right)^i p\varepsilon = 4\varepsilon$$

as desired. □

Matrices such as $\mathbf{P}\circ(\mathbf{PXP})$ appears in the derivative of $\mathbf{P}^{(2)}$, and can be bounded as follows.

Lemma A.4 (Projection matrix facts)**.** *Let $\mathbf{P}$ be a projection matrix with diagonal given by $\mathbf{T}$. For all vectors x, v and $\mathbf{X} = \mathbf{diag}(x), \mathbf{V} = \mathbf{diag}(v)$ we have that*

$$|\mathbf{P}\circ(\mathbf{PXP})v| \leq \frac{1}{2}\mathbf{P}^{(2)}(x^2) + \frac{1}{2}\mathbf{P}^{(2)}(v^2)$$

as vectors coordinate-wise. In particular, we also have that

- $\left\|\mathbf{T}^{-1}\mathbf{P}\circ(\mathbf{PXP})v\right\|_\infty \leq \frac{1}{2}(\|x\|_\infty^2 + \|v\|_\infty^2)$.
- $\left\|\mathbf{T}^{-1}\mathbf{P}\circ(\mathbf{PXP})v\right\|_\tau \leq \frac{1}{2}(\|x^2\|_{\mathbf{P}^{(2)}} + \|v^2\|_{\mathbf{P}^{(2)}}) \leq \frac{1}{2}(\|x^2\|_\tau + \|v^2\|_\tau)$.

Proof. By the inequality $|a^\top b| \leq \frac{1}{2}a^\top a + \frac{1}{2}b^\top b$ we have

$$\begin{aligned}
|e_i^\top \mathbf{P}\circ(\mathbf{PXP})v| &= \left|e_i^\top \mathbf{PXPVP}e_i\right| \\
&\leq \frac{1}{2}e_i^\top \mathbf{PXPXP}e_i + \frac{1}{2}e_i^\top \mathbf{PVPVP}e_i \\
&\leq \frac{1}{2}e_i^\top \mathbf{PX}^2\mathbf{P}e_i + \frac{1}{2}e_i^\top \mathbf{PV}^2\mathbf{P}e_i \\
&= \frac{1}{2}e_i^\top \mathbf{P}^{(2)}(x^2) + \frac{1}{2}e_i^\top \mathbf{P}^{(2)}(v^2).
\end{aligned}$$

Also, we then can use Lemma A.1 that $\|\mathbf{T}^{-1}\mathbf{P}^{(2)}\|_\infty \leq 1$ to get

$$\left\|\mathbf{T}^{-1}\mathbf{P}\circ(\mathbf{PXP})v\right\|_\infty \leq \frac{1}{2}\|\mathbf{T}^{-1}\mathbf{P}^{(2)}(x^2)\|_\infty + \frac{1}{2}\|\mathbf{T}^{-1}\mathbf{P}^{(2)}(v^2)\|_\infty \leq \frac{1}{2}(\|x\|_\infty^2 + \|v\|_\infty^2).$$

Finally, we have that

$$\begin{aligned}
\left\|\mathbf{T}^{-1}\mathbf{P}\circ(\mathbf{PXP})v\right\|_\tau &\leq \frac{1}{2}\|\mathbf{T}^{-1}\mathbf{P}^{(2)}(x^2)\|_\tau + \frac{1}{2}\|\mathbf{T}^{-1}\mathbf{P}^{(2)}(v^2)\|_\tau \\
&= \frac{1}{2}(\|x^2\|_{\mathbf{P}^{(2)}\mathbf{T}^{-1}\mathbf{P}^{(2)}} + \|v^2\|_{\mathbf{P}^{(2)}\mathbf{T}^{-1}\mathbf{P}^{(2)}}) \\
&\leq \frac{1}{2}(\|x^2\|_{\mathbf{P}^{(2)}} + \|v^2\|_{\mathbf{P}^{(2)}})
\end{aligned}$$

as $\mathbf{P}^{(2)} \preceq \mathbf{T}$, so $\mathbf{T}^{-1} \preceq (\mathbf{P}^{(2)})^{-1}$. □

Lemma 4.26 (Sharper bound on changes in τ part 2)**.** *Let $\delta_c = c^{(\text{new})} - c$ be a γ-bounded change, and let $\tau^{(\text{new})} = \tau_1$, and $\delta_\tau = \tau^{(\text{new})} - \tau$. For $\mathbf{J}$ as defined in Lemma 4.19, we have*

$$\left\|\mathbf{T}^{-1}(\mathbb{E}[\delta_\tau] - \mathbf{J}\mathbb{E}[\delta_c])\right\|_{\tau+\infty} \lesssim \gamma^2.$$

Proof. We adopt the same notation as Lemma 4.24, and also define $\mathbf{N}_t = 2\left(\mathbf{I} - \left(1 - \frac{2}{p}\right)\overline{\mathbf{\Lambda}_t}\right)^{-1}\overline{\mathbf{\Lambda}_t}$. We have

$$\mathbf{T}^{-1}(\mathbb{E}[\delta_\tau] - \mathbf{J}\mathbb{E}[\delta_c]) = \int_0^1 (1-t)\mathbf{T}^{-1}\mathbb{E}\left[\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{J}_t\delta_c\right]. \tag{65}$$

Recall that $\mathbf{J}_t = \mathbf{T}_t^{\frac{1}{2}}\mathbf{N}_t\mathbf{T}_t^{\frac{1}{2}}\mathbf{C}_t^{-1}$, so

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{J}_t = &\left(\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{T}_t^{\frac{1}{2}}\right)\mathbf{N}_t\mathbf{T}_t^{\frac{1}{2}}\mathbf{C}_t^{-1} + \mathbf{T}_t^{\frac{1}{2}}\left(\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{N}_t\right)\mathbf{T}_t^{\frac{1}{2}}\mathbf{C}_t^{-1}\\ &+ \mathbf{T}_t^{\frac{1}{2}}\mathbf{N}_t\left(\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{T}_t^{\frac{1}{2}}\right)\mathbf{C}_t^{-1} + \mathbf{T}_t^{\frac{1}{2}}\mathbf{N}_t\mathbf{T}_t^{\frac{1}{2}}\left(\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{C}_t^{-1}\right).\end{aligned}$$

We will bound all four terms separately. Define $\Delta_{\tau_t} = \mathbf{diag}(\delta_{\tau_t})$, and note that $\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{T}_t = \Delta_{\tau_t}$. Therefore, we may use Lemma 4.24 to bound the first term with

$$\begin{aligned}\left\|\mathbb{E}\mathbf{T}^{-1}\left(\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{T}_t^{\frac{1}{2}}\right)\mathbf{N}_t\mathbf{T}_t^{\frac{1}{2}}\mathbf{C}_t^{-1}\delta_c\right\|_{\tau+\infty} &= \frac{1}{2}\left\|\mathbb{E}\mathbf{T}^{-1}\mathbf{T}_t^{-1}\Delta_{\tau_t}\mathbf{T}_t^{\frac{1}{2}}\mathbf{N}_t\mathbf{T}_t^{\frac{1}{2}}\mathbf{C}_t^{-1}\delta_c\right\|_{\tau+\infty}\\ &= .5\left\|\mathbb{E}[\mathbf{T}^{-1}\mathbf{T}_t^{-1}\Delta_{\tau_t}\mathbf{J}_t\delta_c]\right\|_{\tau+\infty}\\ &\lesssim \left\|\mathbb{E}[(\mathbf{T}^{-1}\delta_{\tau_t})^2]\right\|_{\tau+\infty} \lesssim \gamma^2.\end{aligned}$$

For the third term we first define $v_t = \mathbf{T}_t^{-1}\Delta_{\tau_t}\mathbf{C}_t^{-1}\delta_c$ and use Lemma 4.20 to get

$$\begin{aligned}\left\|\mathbb{E}\mathbf{T}^{-1}\mathbf{T}_t^{\frac{1}{2}}\mathbf{N}_t\left(\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{T}_t^{\frac{1}{2}}\right)\mathbf{C}_t^{-1}\delta_c\right\|_{\tau+\infty} &\lesssim \left\|\mathbb{E}\mathbf{T}^{-1}\mathbf{T}_t\mathbf{T}_t^{-1}\mathbf{T}_t^{\frac{1}{2}}\mathbf{N}_t\mathbf{T}_t^{\frac{1}{2}}\mathbf{T}_t^{-1}\Delta_{\tau_t}\mathbf{C}_t^{-1}\delta_c\right\|_{\tau+\infty}\\ &= \left\|\mathbb{E}\mathbf{T}^{-1}\mathbf{T}_t\mathbf{T}_t^{-1}\mathbf{J}_t\mathbf{C}_t v_t\right\|_{\tau+\infty}\\ &\le \|\mathbb{E}\mathbf{T}^{-1}\mathbf{T}_t\mathbf{D}_t v_t\|_{\tau+\infty} + \|\mathbb{E}\mathbf{T}^{-1}\mathbf{T}_t\mathbf{K}_t v_t\|_{\tau+\infty}.\end{aligned}$$

Note that

$$|v_t| \lesssim (\mathbf{T}_t^{-1}\delta_{\tau_t})^2 + (\mathbf{C}_t^{-1}\delta_c)^2.$$

Therefore, we use Lemma 4.24 and γ-boundedness (Definition 4.23 (17)) to bound

$$\|\mathbb{E}\mathbf{T}^{-1}\mathbf{T}_t\mathbf{D}_t v_t\|_{\tau+\infty} \lesssim \|\mathbb{E}(\mathbf{T}_t^{-1}\delta_{\tau_t})^2\|_{\tau+\infty} + \|\mathbb{E}(\mathbf{C}_t^{-1}\delta_c)^2\|_{\tau+\infty} \lesssim \gamma^2.$$

For the term with $\mathbf{K}_t$, we use Lemma 4.20 to first bound

$$\|\mathbf{T}^{-1}\mathbf{T}_t\mathbf{K}_t v_t\|_\infty \lesssim \|\mathbf{K}_t v_t\|_\infty \lesssim \|\mathbf{T}_t^{-1}\delta_{\tau_t}\|_\infty\|\mathbf{C}_t^{-1}\delta_c\|_\infty \lesssim \gamma^2.$$

Also, we can use Lemma 4.20, Lemma 4.24, and γ-boundedness (Definition 4.23 (18)) to get

$$\begin{aligned}\mathbb{E}\|\mathbf{T}^{-1}\mathbf{T}_t\mathbf{K}_t v_t\|_\tau &\lesssim \mathbb{E}\|\mathbf{K}_t v_t\|_\tau\\ &\le \mathbb{E}\||\mathbf{T}_t^{-1}\Delta_{\tau_t}\mathbf{C}_t^{-1}\delta_c|\|_{\mathbf{P}_t^{(2)}}\\ &\lesssim \gamma\mathbb{E}\|\mathbf{C}^{-1}|\delta_c|\|_{\mathbf{P}_t^{(2)}} \lesssim \gamma^2/C_{\text{norm}}.\end{aligned}$$

Therefore, the total contribution from the third term is $O(\gamma^2)$. For the fourth term, we use Lemma 4.20 to write

$$\left\|\mathbb{E}\mathbf{T}^{-1}\mathbf{T}_t^{\frac{1}{2}}\mathbf{N}_t\mathbf{T}_t^{\frac{1}{2}}\left(\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{C}_t^{-1}\right)\delta_c\right\|_{\tau+\infty} = \left\|\mathbb{E}\mathbf{T}^{-1}\mathbf{T}_t\mathbf{T}_t^{-1}\mathbf{J}_t\mathbf{C}_t(\mathbf{C}_t^{-1}\delta_c)^2\right\|_{\tau+\infty}$$
$$\leq \|\mathbb{E}\mathbf{T}^{-1}\mathbf{T}_t\mathbf{D}_t(\mathbf{C}_t^{-1}\delta_c)^2\|_{\tau+\infty} + \|\mathbb{E}\mathbf{T}^{-1}\mathbf{T}_t\mathbf{K}_t(\mathbf{C}_t^{-1}\delta_c)^2\|_{\tau+\infty}.$$

For the first piece, use γ-boundedness (Definition 4.23 (17)) to bound

$$\|\mathbb{E}\mathbf{T}^{-1}\mathbf{T}_t\mathbf{D}_t(\mathbf{C}_t^{-1}\delta_c)^2\|_{\tau+\infty} \lesssim \|\mathbb{E}(\mathbf{C}_t^{-1}\delta_c)^2\|_{\tau+\infty} \lesssim \gamma^2.$$

For the second piece, we use Lemma 4.20 and γ-boundedness (Definition 4.23 (17)) to first bound

$$\|\mathbb{E}\mathbf{T}^{-1}\mathbf{T}_t\mathbf{K}_t(\mathbf{C}_t^{-1}\delta_c)^2\|_{\infty} \lesssim \|(\mathbf{C}_t^{-1}\delta_c)^2\|_{\infty} \lesssim \gamma^2.$$

For the τ-norm we can use Lemma 4.20, γ-boundedness (Definition 4.23 (17) and (18)) to bound

$$\mathbb{E}\|\mathbf{T}^{-1}\mathbf{T}_t\mathbf{K}_t(\mathbf{C}_t^{-1}\delta_c)^2\|_{\tau} \lesssim \mathbb{E}\|(\mathbf{C}_t^{-1}\delta_c)^2\|_{\mathbf{P}_t^{(2)}} \lesssim \gamma^2/C_{\mathrm{norm}}.$$

Therefore, the total contribution from this case is at most $O(\gamma^2)$.

Now we bound the contribution from the $\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{N}_t$ term.

Analysis of $\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{N}_t$. By the chain rule, we have that

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{N}_t &= 2\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}_t}\right)^{-1}\frac{\mathrm{d}}{\mathrm{d}t}\overline{\mathbf{\Lambda}_t}\\ &\quad + 2\left(1-\frac{2}{p}\right)\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}_t}\right)^{-1}\frac{\mathrm{d}}{\mathrm{d}t}\overline{\mathbf{\Lambda}_t}\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}_t}\right)^{-1}\overline{\mathbf{\Lambda}_t}\\ &= 2\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}_t}\right)^{-1}\frac{\mathrm{d}}{\mathrm{d}t}\overline{\mathbf{\Lambda}_t}\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}_t}\right)^{-1}.\end{aligned}$$

Recall that $\overline{\mathbf{\Lambda}_t} = \mathbf{T}_t^{-1}\mathbf{\Sigma}_t - \mathbf{T}_t^{-\frac{1}{2}}\mathbf{P}_t^{(2)}\mathbf{T}_t^{-\frac{1}{2}}$. Define the vectors $z_t = \mathbf{T}_t^{-1}\sigma_t, \ell_t = \mathbf{T}_t^{\frac{1}{2}-\frac{1}{p}}c_t$, and the diagonal matrices $\mathbf{Z}_t = \mathbf{diag}(z_t) = \mathbf{T}_t^{-1}\mathbf{\Sigma}_t$, $\mathbf{L}_t = \mathbf{T}_t^{\frac{1}{2}-\frac{1}{p}}\mathbf{C}_t$, so that $\mathbf{P}_t = \mathbf{P}(\mathbf{L}_t\mathbf{A}) = \mathbf{A}^\top\mathbf{L}_t(\mathbf{A}^\top\mathbf{L}_t^2\mathbf{A})^{-1}\mathbf{L}_t\mathbf{A}$. In this way, $\overline{\mathbf{\Lambda}_t} = \mathbf{Z}_t - \mathbf{T}_t^{-\frac{1}{2}}\mathbf{P}(\mathbf{L}_t\mathbf{A})^{(2)}\mathbf{T}_t^{-\frac{1}{2}}$.

Define $\delta_{z_t} = \frac{\mathrm{d}}{\mathrm{d}t}z_t, \delta_{\ell_t} = \frac{\mathrm{d}}{\mathrm{d}t}\ell_t$, and $\Delta_{z_t} = \mathbf{diag}(\delta_{z_t})$, $\Delta_{\ell_t} = \mathbf{diag}(\delta_{\ell_t})$. We have

$$\Delta_{z_t} = -\mathbf{T}_t^{-2}\mathbf{\Sigma}_t\Delta_{\tau_t} + \mathbf{T}_t^{-1}\Delta_{\tau_t} = \mathbf{T}_t^{-1}\Delta_{\tau_t}(\mathbf{I}-\mathbf{T}_t^{-1}\mathbf{\Sigma}_t).$$

and

$$\Delta_{\ell_t} = \left(\frac{1}{2}-\frac{1}{p}\right)\mathbf{T}_t^{-\frac{1}{2}-\frac{1}{p}}\mathbf{C}_t\Delta_{\tau_t} + \mathbf{T}_t^{\frac{1}{2}-\frac{1}{p}}\delta_c$$

so

$$\mathbf{L}_t^{-1}\Delta_{\ell_t} = \left(\frac{1}{2}-\frac{1}{p}\right)\mathbf{T}_t^{-1}\Delta_{\tau_t} + \mathbf{C}_t^{-1}\delta_c.$$

Therefore, using Lemma 4.24, γ-boundedness (Definition 4.23 (16), (17), (18)) we get that

$$\|\delta_{z_t}\|_\infty \lesssim \gamma \quad \text{and} \quad \|\mathbb{E}[\delta_{z_t}^2]\|_{\tau+\infty} \lesssim \gamma^2 \quad \text{and} \quad \mathbb{E}\|\|\delta_{z_t}\|\|_{\mathbf{P}_t^{(2)}} \lesssim \gamma/C_{\mathrm{norm}} \tag{66}$$

and

$$\|\mathbf{L}_t^{-1}\delta_{\ell_t}\|_\infty \lesssim \gamma \quad \text{and} \quad \|\mathbb{E}[(\mathbf{L}_t^{-1}\delta_{\ell_t})^2]\|_{\tau+\infty} \lesssim \gamma^2 \quad \text{and} \quad \mathbb{E}\|\mathbf{L}_t^{-1}|\delta_{\ell_t}|\|_{\mathbf{P}_t^{(2)}} \lesssim \gamma/C_{\mathrm{norm}}. \tag{67}$$

A direct calculation gives that

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\overline{\mathbf{\Lambda}_t} &= \Delta_{z_t} + \frac{1}{2}\mathbf{T}_t^{-\frac{3}{2}}\Delta_{\tau_t}\mathbf{P}_t^{(2)}\mathbf{T}_t^{-\frac{1}{2}} + \frac{1}{2}\mathbf{T}_t^{-\frac{1}{2}}\mathbf{P}_t^{(2)}\mathbf{T}_t^{-\frac{3}{2}}\Delta_{\tau_t} - 2\mathbf{T}_t^{-\frac{1}{2}}\mathbf{P}(\mathbf{L}_t\mathbf{A})\circ\left(\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{P}(\mathbf{L}_t\mathbf{A})\right)\mathbf{T}_t^{-\frac{1}{2}} \\ &= \Delta_{z_t} + \frac{1}{2}\mathbf{T}_t^{-\frac{3}{2}}\Delta_{\tau_t}\mathbf{P}_t^{(2)}\mathbf{T}_t^{-\frac{1}{2}} + \frac{1}{2}\mathbf{T}_t^{-\frac{1}{2}}\mathbf{P}_t^{(2)}\mathbf{T}_t^{-\frac{3}{2}}\Delta_{\tau_t} \qquad (68)\\ &- 2\mathbf{T}_t^{-\frac{1}{2}}\mathbf{P}_t^{(2)}\mathbf{L}_t^{-1}\Delta_{\ell_t}\mathbf{T}_t^{-\frac{1}{2}} - 2\mathbf{T}_t^{-\frac{1}{2}}\mathbf{L}_t^{-1}\Delta_{\ell_t}\mathbf{P}_t^{(2)}\mathbf{T}_t^{-\frac{1}{2}} + 4\mathbf{T}_t^{-\frac{1}{2}}(\mathbf{P}_t\circ(\mathbf{P}_t\mathbf{L}_t^{-1}\Delta_{\ell_t}\mathbf{P}_t))\mathbf{T}_t^{-\frac{1}{2}}. \qquad (69)\end{aligned}$$

There are six terms here and we analyze them one by one. At a high level, for each of the six terms, we will use Lemma 4.21 to handle the $\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}_t}\right)^{-1}$ terms in $\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{N}_t$. For simplicity, we will omit the factor of 2. For the Δ_{z_t} term in (68) we get

$$\begin{aligned}&\mathbf{T}^{-1}\mathbf{T}_t^{\frac{1}{2}}\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}_t}\right)^{-1}\Delta_{z_t}\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}_t}\right)^{-1}\mathbf{T}_t^{\frac{1}{2}}\mathbf{C}_t^{-1}\delta_c \\ &= \mathbf{T}^{-1}\mathbf{T}_t(\mathbf{D}_t'+\mathbf{K}_t')\Delta_{z_t}(\mathbf{D}_t'+\mathbf{K}_t')\mathbf{C}_t^{-1}\delta_c \\ &= \mathbf{T}^{-1}\mathbf{T}_t\mathbf{D}_t'\Delta_{z_t}\mathbf{D}_t'\mathbf{C}_t^{-1}\delta_c + \mathbf{T}^{-1}\mathbf{T}_t\mathbf{K}_t'\Delta_{z_t}\mathbf{D}_t'\mathbf{C}_t^{-1}\delta_c \qquad (70)\\ &+ \mathbf{T}^{-1}\mathbf{T}_t\mathbf{D}_t'\Delta_{z_t}\mathbf{K}_t'\mathbf{C}_t^{-1}\delta_c + \mathbf{T}^{-1}\mathbf{T}_t\mathbf{K}_t'\Delta_{z_t}\mathbf{K}_t'\mathbf{C}_t^{-1}\delta_c. \qquad (71)\end{aligned}$$

We know by (66) and γ-boundedness (Definition 4.23 (16))

$$\left\|\mathbf{T}^{-1}\mathbf{T}_t^{\frac{1}{2}}\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}_t}\right)^{-1}\Delta_{z_t}\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}_t}\right)^{-1}\mathbf{T}_t^{\frac{1}{2}}\mathbf{C}_t^{-1}\delta_c\right\|_\infty \lesssim \gamma^2,$$

so we focus on the τ-norm. First we use (66) and γ-boundedness (Definition 4.23 (17)) to get

$$\|\mathbb{E}[\mathbf{T}^{-1}\mathbf{T}_t\mathbf{D}_t'\Delta_{z_t}\mathbf{D}_t'\mathbf{C}_t^{-1}\delta_c]\|_\tau \le \|\mathbb{E}[\delta_{z_t}^2]\|_\tau + \|\mathbb{E}[(\mathbf{C}_t^{-1}\delta_c)^2]\|_\tau \lesssim \gamma^2/C_{\mathrm{norm}}.$$

For the remaining three terms in (70), (71) we can use Lemma 4.21 to bound

$$\|\mathbb{E}[\mathbf{T}^{-1}\mathbf{T}_t\mathbf{K}_t'\Delta_{z_t}\mathbf{D}_t'\mathbf{C}_t^{-1}\delta_c]\|_\tau \lesssim \|\delta_{z_t}\|_\infty\mathbb{E}\|\mathbf{C}_t^{-1}|\delta_c|\|_{\mathbf{P}_t^{(2)}} \lesssim \gamma^2/C_{\mathrm{norm}}$$

and similar for the other two terms. Therefore, the total contribution from (70), (71) in $\tau+\infty$ norm is $O(\gamma^2)$.

For the $\frac{1}{2}\mathbf{T}_t^{-\frac{3}{2}}\Delta_{\tau_t}\mathbf{P}_t^{(2)}\mathbf{T}_t^{-\frac{1}{2}}$ term in (68) we can use Lemma 4.21 to write (omitting the $\frac{1}{2}$)

$$\begin{aligned}&\mathbf{T}^{-1}\mathbf{T}_t^{\frac{1}{2}}\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}_t}\right)^{-1}\mathbf{T}_t^{-\frac{3}{2}}\Delta_{\tau_t}\mathbf{P}_t^{(2)}\mathbf{T}_t^{-\frac{1}{2}}\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}_t}\right)^{-1}\mathbf{T}_t^{\frac{1}{2}}\mathbf{C}_t^{-1}\delta_c \\ &= \mathbf{T}^{-1}\mathbf{T}_t(\mathbf{D}_t'+\mathbf{K}_t')\mathbf{T}_t^{-2}\Delta_{\tau_t}\mathbf{P}_t^{(2)}(\mathbf{D}_t'+\mathbf{K}_t')\mathbf{C}_t^{-1}\delta_c \\ &= \mathbf{T}^{-1}\mathbf{T}_t\mathbf{D}_t'\mathbf{T}_t^{-2}\Delta_{\tau_t}\mathbf{P}_t^{(2)}\mathbf{D}_t'\mathbf{C}_t^{-1}\delta_c + \mathbf{T}^{-1}\mathbf{T}_t\mathbf{K}_t'\mathbf{T}_t^{-2}\Delta_{\tau_t}\mathbf{P}_t^{(2)}\mathbf{D}_t'\mathbf{C}_t^{-1}\delta_c \qquad (72)\\ &+ \mathbf{T}^{-1}\mathbf{T}_t\mathbf{D}_t'\mathbf{T}_t^{-2}\Delta_{\tau_t}\mathbf{P}_t^{(2)}\mathbf{K}_t'\mathbf{C}_t^{-1}\delta_c + \mathbf{T}^{-1}\mathbf{T}_t\mathbf{K}_t'\mathbf{T}_t^{-2}\Delta_{\tau_t}\mathbf{P}_t^{(2)}\mathbf{K}_t'\mathbf{C}_t^{-1}\delta_c. \qquad (73)\end{aligned}$$

We can bound using $\|\mathbf{T}_t^{-1}\mathbf{P}_t^{(2)}\|_\infty \le 1$ (Lemma A.1)

$$\left\|\mathbf{T}^{-1}\mathbf{T}_t^{\frac{1}{2}}\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}_t}\right)^{-1}\mathbf{T}_t^{-\frac{3}{2}}\Delta_{\tau_t}\mathbf{P}_t^{(2)}\mathbf{T}_t^{-\frac{1}{2}}\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}_t}\right)^{-1}\mathbf{T}_t^{\frac{1}{2}}\mathbf{C}_t^{-1}\delta_c\right\|_\infty \lesssim \gamma^2.$$

To bound the τ-norm we can bound the first term in (72) with

$$\|\mathbb{E}[\mathbf{T}^{-1}\mathbf{T}_t\mathbf{D}_t'\mathbf{T}_t^{-2}\Delta_{\tau_t}\mathbf{P}_t^{(2)}\mathbf{D}_t'\mathbf{C}_t^{-1}\delta_c]\|_\tau \lesssim \|\mathbf{T}_t^{-1}\delta_{\tau_t}\|_\infty\mathbb{E}\|\mathbf{C}_t^{-1}|\delta_c|\|_{\mathbf{P}_t^{(2)}} \lesssim \gamma^2/C_{\mathrm{norm}}.$$

The other terms in (72), (73) follow similarly, e.g. using $\|\mathbf{T}_t^{-1}\mathbf{P}_t^{(2)}\|_\tau \le 1$

$$\begin{aligned}\mathbb{E}[\|\mathbf{T}^{-1}\mathbf{T}_t\mathbf{D}_t'\mathbf{T}_t^{-2}\Delta_{\tau_t}\mathbf{P}_t^{(2)}\mathbf{K}_t'\mathbf{C}_t^{-1}\delta_c\|_\tau] &\lesssim \mathbb{E}[\|\mathbf{T}_t^{-1}\Delta_{\tau_t}\mathbf{T}_t^{-1}\mathbf{P}_t^{(2)}\mathbf{K}_t'\mathbf{C}_t^{-1}\delta_c\|_\tau]\\ &\lesssim \|\mathbf{T}_t^{-1}\delta_{\tau_t}\|_\infty\mathbb{E}[\|\mathbf{K}_t'\mathbf{C}_t^{-1}\delta_c\|_\tau]\\ &\lesssim \gamma\mathbb{E}[\||\mathbf{C}_t^{-1}|\delta_c|\|_{\mathbf{P}_t^{(2)}}]\\ &\lesssim \gamma^2/C_{\text{norm}}.\end{aligned}$$

Therefore, the total contribution from (72), (73) in $\tau+\infty$ norm is $O(\gamma^2)$. The $\frac{1}{2}\mathbf{T}_t^{-\frac{1}{2}}\mathbf{P}_t^{(2)}\mathbf{T}_t^{-\frac{3}{2}}\Delta_{\tau_t}$ term in (68) can be handled equivalently.

We turn to the $2\mathbf{T}_t^{-\frac{1}{2}}\mathbf{P}_t^{(2)}\mathbf{L}_t^{-1}\Delta_{\ell_t}\mathbf{T}_t^{-\frac{1}{2}}$ in (69), which can be handled similarly to previous bounds. We can use Lemma 4.21 to write (omitting the 2)

$$\begin{aligned}&\mathbf{T}^{-1}\mathbf{T}_t^{\frac{1}{2}}\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}_t}\right)^{-1}\mathbf{T}_t^{-\frac{1}{2}}\mathbf{P}_t^{(2)}\mathbf{L}_t^{-1}\Delta_{\ell_t}\mathbf{T}_t^{-\frac{1}{2}}\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}_t}\right)^{-1}\mathbf{T}_t^{\frac{1}{2}}\mathbf{C}_t^{-1}\delta_c\\ &=\mathbf{T}^{-1}\mathbf{T}_t(\mathbf{D}_t'+\mathbf{K}_t')\mathbf{T}_t^{-1}\mathbf{P}_t^{(2)}\mathbf{L}_t^{-1}\Delta_{\ell_t}(\mathbf{D}_t'+\mathbf{K}_t')\mathbf{C}_t^{-1}\delta_c\\ &=\mathbf{T}^{-1}\mathbf{T}_t\mathbf{D}_t'\mathbf{T}_t^{-1}\mathbf{P}_t^{(2)}\mathbf{L}_t^{-1}\Delta_{\ell_t}\mathbf{D}_t'\mathbf{C}_t^{-1}\delta_c+\mathbf{T}^{-1}\mathbf{T}_t\mathbf{K}_t'\mathbf{T}_t^{-1}\mathbf{P}_t^{(2)}\mathbf{L}_t^{-1}\Delta_{\ell_t}\mathbf{D}_t'\mathbf{C}_t^{-1}\delta_c \quad (74)\\ &+\mathbf{T}^{-1}\mathbf{T}_t\mathbf{D}_t'\mathbf{T}_t^{-1}\mathbf{P}_t^{(2)}\mathbf{L}_t^{-1}\Delta_{\ell_t}\mathbf{K}_t'\mathbf{C}_t^{-1}\delta_c+\mathbf{T}^{-1}\mathbf{T}_t\mathbf{K}_t'\mathbf{T}_t^{-1}\mathbf{P}_t^{(2)}\mathbf{L}_t^{-1}\Delta_{\ell_t}\mathbf{K}_t'\mathbf{C}_t^{-1}\delta_c. \quad (75)\end{aligned}$$

We can bound using $\|\mathbf{T}_t^{-1}\mathbf{P}_t^{(2)}\|_\infty \le 1$ (Lemma A.1)

$$\left\|\mathbf{T}^{-1}\mathbf{T}_t^{\frac{1}{2}}\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}_t}\right)^{-1}\mathbf{T}_t^{-\frac{1}{2}}\mathbf{P}_t^{(2)}\mathbf{L}_t^{-1}\Delta_{\ell_t}\mathbf{T}_t^{-\frac{1}{2}}\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}_t}\right)^{-1}\mathbf{T}_t^{\frac{1}{2}}\mathbf{C}_t^{-1}\delta_c\right\|_\infty \lesssim \gamma^2.$$

To bound the τ-norm we can bound the first term in (74) with

$$\|\mathbb{E}[\mathbf{T}^{-1}\mathbf{T}_t\mathbf{D}_t'\mathbf{T}_t^{-1}\mathbf{P}_t^{(2)}\mathbf{L}_t^{-1}\Delta_{\ell_t}\mathbf{D}_t'\mathbf{C}_t^{-1}\delta_c]\|_\tau \lesssim \|\mathbf{L}_t^{-1}\delta_{\ell_t}\|_\infty\mathbb{E}\||\mathbf{C}_t^{-1}|\delta_c|\|_{\mathbf{P}_t^{(2)}} \lesssim \gamma^2/C_{\text{norm}}.$$

The other terms in (72), (73) follow similarly, e.g. using $\|\mathbf{T}_t^{-1}\mathbf{P}_t^{(2)}\|_\tau \le 1$

$$\begin{aligned}\mathbb{E}\|\mathbf{T}^{-1}\mathbf{T}_t\mathbf{D}_t'\mathbf{T}_t^{-1}\mathbf{P}_t^{(2)}\mathbf{L}_t^{-1}\Delta_{\ell_t}\mathbf{K}_t'\mathbf{C}_t^{-1}\delta_c\|_\tau &\lesssim \mathbb{E}\|\mathbf{T}_t^{-1}\mathbf{P}_t^{(2)}\mathbf{L}_t^{-1}\Delta_{\ell_t}\mathbf{K}_t'\mathbf{C}_t^{-1}\delta_c\|_\tau\\ &\lesssim \mathbb{E}\|\mathbf{L}_t^{-1}\Delta_{\ell_t}\mathbf{K}_t'\mathbf{C}_t^{-1}\delta_c\|_\tau\\ &\lesssim \|\mathbf{L}_t^{-1}\delta_{\ell_t}\|_\infty\mathbb{E}\|\mathbf{K}_t'\mathbf{C}_t^{-1}\delta_c\|_\tau\\ &\lesssim \gamma\mathbb{E}\||\mathbf{C}_t^{-1}|\delta_c|\|_{\mathbf{P}_t^{(2)}} \lesssim \gamma^2/C_{\text{norm}}.\end{aligned}$$

Therefore, the total contribution from the $2\mathbf{T}_t^{-\frac{1}{2}}\mathbf{P}_t^{(2)}\mathbf{L}_t^{-1}\Delta_{\ell_t}\mathbf{T}_t^{-\frac{1}{2}}$ term in (69) $O(\gamma^2)$. The $2\mathbf{T}_t^{-\frac{1}{2}}\mathbf{L}_t^{-1}\Delta_{\ell_t}\mathbf{P}_t^{(2)}\mathbf{T}_t^{-\frac{1}{2}}$ term in (69) can be handled equivalently.

Finally we bound the $4\mathbf{T}_t^{-\frac{1}{2}}(\mathbf{P}_t\circ(\mathbf{P}_t\mathbf{L}_t^{-1}\Delta_{\ell_t}\mathbf{P}_t))\mathbf{T}_t^{-\frac{1}{2}}$ term in (69). We start by using Lemma 4.21 to write (omitting the factor of 4)

$$\begin{aligned}&\mathbf{T}^{-1}\mathbf{T}_t^{\frac{1}{2}}\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}_t}\right)^{-1}\mathbf{T}_t^{-\frac{1}{2}}(\mathbf{P}_t\circ(\mathbf{P}_t\mathbf{L}_t^{-1}\Delta_{\ell_t}\mathbf{P}_t))\mathbf{T}_t^{-\frac{1}{2}}\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}_t}\right)^{-1}\mathbf{T}_t^{\frac{1}{2}}\mathbf{C}_t^{-1}\delta_c\\ &=\mathbf{T}^{-1}\mathbf{T}_t(\mathbf{D}_t'+\mathbf{K}_t')\mathbf{T}_t^{-1}(\mathbf{P}_t\circ(\mathbf{P}_t\mathbf{L}_t^{-1}\Delta_{\ell_t}\mathbf{P}_t))(\mathbf{D}_t'+\mathbf{K}_t')\mathbf{C}_t^{-1}\delta_c\\ &=\mathbf{T}^{-1}\mathbf{T}_t\mathbf{D}_t'\mathbf{T}_t^{-1}(\mathbf{P}_t\circ(\mathbf{P}_t\mathbf{L}_t^{-1}\Delta_{\ell_t}\mathbf{P}_t))\mathbf{D}_t'\mathbf{C}_t^{-1}\delta_c+\mathbf{T}^{-1}\mathbf{T}_t\mathbf{K}_t'\mathbf{T}_t^{-1}(\mathbf{P}_t\circ(\mathbf{P}_t\mathbf{L}_t^{-1}\Delta_{\ell_t}\mathbf{P}_t))\mathbf{D}_t'\mathbf{C}_t^{-1}\delta_c \quad (76)\\ &+\mathbf{T}^{-1}\mathbf{T}_t\mathbf{D}_t'\mathbf{T}_t^{-1}(\mathbf{P}_t\circ(\mathbf{P}_t\mathbf{L}_t^{-1}\Delta_{\ell_t}\mathbf{P}_t))\mathbf{K}_t'\mathbf{C}_t^{-1}\delta_c+\mathbf{T}^{-1}\mathbf{T}_t\mathbf{K}_t'\mathbf{T}_t^{-1}(\mathbf{P}_t\circ(\mathbf{P}_t\mathbf{L}_t^{-1}\Delta_{\ell_t}\mathbf{P}_t))\mathbf{K}_t'\mathbf{C}_t^{-1}\delta_c. \quad (77)\end{aligned}$$

Using Lemma A.3 and A.4 we get that

$$\begin{aligned}
&\left\|\mathbf{T}^{-1}\mathbf{T}_t^{\frac{1}{2}}\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}_t}\right)^{-1}\mathbf{T}_t^{-\frac{1}{2}}(\mathbf{P}_t\circ(\mathbf{P}_t\mathbf{L}_t^{-1}\Delta_{\ell_t}\mathbf{P}_t))\mathbf{T}_t^{-\frac{1}{2}}\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}_t}\right)^{-1}\mathbf{T}_t^{\frac{1}{2}}\mathbf{C}_t^{-1}\delta_c\right\|_\infty \\
\lesssim &\left\|\mathbf{T}_t^{-1}(\mathbf{P}_t\circ(\mathbf{P}_t\mathbf{L}_t^{-1}\Delta_{\ell_t}\mathbf{P}_t))\mathbf{T}_t^{-\frac{1}{2}}\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}_t}\right)^{-1}\mathbf{T}_t^{\frac{1}{2}}\mathbf{C}_t^{-1}\delta_c\right\|_\infty \\
\lesssim &\|\mathbf{L}_t^{-1}\Delta_{\ell_t}\|_\infty^2+\left\|\mathbf{T}_t^{-\frac{1}{2}}\left(\mathbf{I}-\left(1-\frac{2}{p}\right)\overline{\mathbf{\Lambda}_t}\right)^{-1}\mathbf{T}_t^{\frac{1}{2}}\mathbf{C}_t^{-1}\delta_c\right\|_\infty^2\lesssim\gamma^2
\end{aligned}$$

by (67) and γ-boundedness (Definition 4.23 (16)). We now bound the τ-norm of all four terms in (76) and (77). For the first term in (76) we bound using Lemma A.4 and (67)

$$\begin{aligned}
&\mathbb{E}\|\mathbf{T}^{-1}\mathbf{T}_t\mathbf{D}_t'\mathbf{T}_t^{-1}(\mathbf{P}_t\circ(\mathbf{P}_t\mathbf{L}_t^{-1}\Delta_{\ell_t}\mathbf{P}_t))\mathbf{D}_t'\mathbf{C}_t^{-1}\delta_c\|_\tau \\
\lesssim\ &\mathbb{E}\|\mathbf{T}_t^{-1}(\mathbf{P}_t\circ(\mathbf{P}_t\mathbf{L}_t^{-1}\Delta_{\ell_t}\mathbf{P}_t))\mathbf{D}_t'\mathbf{C}_t^{-1}\delta_c\|_\tau &(78)\\
\lesssim\ &\mathbb{E}\|(\mathbf{L}_t^{-1}\delta_{\ell_t})^2\|_{\mathbf{P}_t^{(2)}}+\mathbb{E}\|(\mathbf{D}_t'\mathbf{C}_t^{-1}\delta_c)^2\|_{\mathbf{P}_t^{(2)}} \\
\lesssim\ &\|\mathbf{L}_t^{-1}\delta_{\ell_t}\|_\infty\mathbb{E}\|\mathbf{L}_t^{-1}|\delta_{\ell_t}|\|_{\mathbf{P}_t^{(2)}}+\|\mathbf{C}_t^{-1}\delta_c\|_\infty\mathbb{E}\|\mathbf{C}_t^{-1}|\delta_c|\|_{\mathbf{P}_t^{(2)}} \\
\lesssim\ &\gamma^2/C_{\text{norm}} &(79)
\end{aligned}$$

For the second term in (76) we also get

$$\begin{aligned}
&\mathbb{E}[\|\mathbf{T}^{-1}\mathbf{T}_t\mathbf{K}_t'\mathbf{T}_t^{-1}(\mathbf{P}_t\circ(\mathbf{P}_t\mathbf{L}_t^{-1}\Delta_{\ell_t}\mathbf{P}_t))\mathbf{D}_t'\mathbf{C}_t^{-1}\delta_c\|_\tau] \\
\lesssim\ &\mathbb{E}[\|\mathbf{T}_t^{-1}(\mathbf{P}_t\circ(\mathbf{P}_t\mathbf{L}_t^{-1}\Delta_{\ell_t}\mathbf{P}_t))\mathbf{D}_t'\mathbf{C}_t^{-1}\delta_c\|_\tau]\lesssim\gamma^2/C_{\text{norm}}
\end{aligned}$$

exactly as from (78) to (79). For the first term in (77) we use Lemma A.4 and (67) to bound

$$\begin{aligned}
&\mathbb{E}\|\mathbf{T}^{-1}\mathbf{T}_t\mathbf{D}_t'\mathbf{T}_t^{-1}(\mathbf{P}_t\circ(\mathbf{P}_t\mathbf{L}_t^{-1}\Delta_{\ell_t}\mathbf{P}_t))\mathbf{K}_t'\mathbf{C}_t^{-1}\delta_c\|_\tau \\
\lesssim\ &\mathbb{E}\|\mathbf{T}_t^{-1}(\mathbf{P}_t\circ(\mathbf{P}_t\mathbf{L}_t^{-1}\Delta_{\ell_t}\mathbf{P}_t))\mathbf{K}_t'\mathbf{C}_t^{-1}\delta_c\|_\tau &(80)\\
\lesssim\ &\mathbb{E}\|(\mathbf{L}_t^{-1}\delta_{\ell_t})^2\|_{\mathbf{P}_t^{(2)}}+\mathbb{E}\|(\mathbf{K}_t'\mathbf{C}_t^{-1}\delta_c)^2\|_{\mathbf{P}_t^{(2)}} \\
\le\ &\|\mathbf{L}_t^{-1}\delta_{\ell_t}\|_\infty\mathbb{E}\|\mathbf{L}_t^{-1}\delta_{\ell_t}\|_{\mathbf{P}_t^{(2)}}+\mathbb{E}\|(\mathbf{K}_t'\mathbf{C}_t^{-1}\delta_c)^2\|_\tau \\
\le\ &\gamma^2/C_{\text{norm}}+\|\mathbf{K}_t'\mathbf{C}_t^{-1}\delta_c\|_\infty\mathbb{E}\|\mathbf{K}_t'\mathbf{C}_t^{-1}\delta_c\|_\tau \\
\le\ &\gamma^2/C_{\text{norm}}+\gamma\cdot\mathbb{E}\|\mathbf{C}_t^{-1}|\delta_c|\|_{\mathbf{P}_t^{(2)}}\lesssim\gamma^2/C_{\text{norm}} &(81)
\end{aligned}$$

where we have used $\|\mathbf{K}_t'\|_\infty\lesssim 1$ and Lemma 4.25. For the second term in (77), we exactly follow (80) and (81) to bound

$$\begin{aligned}
&\mathbb{E}[\|\mathbf{T}^{-1}\mathbf{T}_t\mathbf{K}_t'\mathbf{T}_t^{-1}(\mathbf{P}_t\circ(\mathbf{P}_t\mathbf{L}_t^{-1}\Delta_{\ell_t}\mathbf{P}_t))\mathbf{K}_t'\mathbf{C}_t^{-1}\delta_c\|_\tau] \\
\lesssim\ &\mathbb{E}[\|\mathbf{T}_t^{-1}(\mathbf{P}_t\circ(\mathbf{P}_t\mathbf{L}_t^{-1}\Delta_{\ell_t}\mathbf{P}_t))\mathbf{K}_t'\mathbf{C}_t^{-1}\delta_c\|_\tau]\lesssim\gamma^2/C_{\text{norm}}.
\end{aligned}$$

Therefore, the total contribution from the $\tau+\infty$-norm from the $4\mathbf{T}_t^{-\frac{1}{2}}(\mathbf{P}_t\circ(\mathbf{P}_t\mathbf{L}_t^{-1}\Delta_{\ell_t}\mathbf{P}_t))\mathbf{T}_t^{-\frac{1}{2}}$ term in (69) is at most $O(\gamma^2)$. Therefore, the $\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{N}_t$ term has total contribution of $O(\gamma^2)$. Combining everything, this gives our desired bound. □

A.3 Initial and Final Point

We will require some basic properties of self-concordant functions.

Lemma A.5 (Theorem 4.1.7, Lemma 4.2.4 in [Nes98])**.** *Let ϕ be a ν-self-concordant function on the domain $\mathcal{X}$. Then for any $x,y\in\mathcal{X}$ we have that*

- $\phi(x)^\top(y-x) \le \nu$.
- $(\nabla\phi(y)-\nabla\phi(x))^\top(y-x) \ge \frac{\|y-x\|^2_{\nabla^2\phi(x)}}{1+\|y-x\|_{\nabla^2\phi(x)}}$.

Lemma 4.11 (Final point)**.** *Given an ε-centered point (x,s,μ) where $\varepsilon \le 1/80$, we can compute a point $(x^{(\text{final})}, s^{(\text{final})})$ satisfying*

1. $\mathbf{A}^\top x^{(\text{final})} = b$, $s^{(\text{final})} = \mathbf{A}y + c$ *for some* y.
2. $c^\top x^{(\text{final})} - \min_{\substack{\mathbf{A}^\top x = b \\ \ell_i \le x_i \le u_i \forall i}} c^\top x \lesssim n\mu$.

The algorithm takes $O(\text{nnz}(\mathbf{A}))$ time plus the time for solving a linear system on $\mathbf{A}^\top\mathbf{D}\mathbf{A}$ where $\mathbf{D}$ is a diagonal matrix.

Proof. Set $s^{(\text{final})} = s$ and $x^{(\text{final})} = x - \overline{\mathbf{T}}^{-1}\Phi''(x)^{-1}\mathbf{A}(\mathbf{A}^\top\overline{\mathbf{T}}^{-1}\Phi''(x)^{-1}\mathbf{A})^{-1}(b-\mathbf{A}^\top x)$. This obviously satisfies the first point. For the second point, there are two steps: We first prove the claim below, and then use the claim to prove the second point.

Claim A.6. $\|\Phi''(x)^{\frac12}(x^{(\text{final})}-x)\|_\infty \lesssim \varepsilon$ *and* $\left\|\frac{s^{(\text{final})}+\mu\tau(x)\phi'(x^{(\text{final})})}{\mu\tau(x)\sqrt{\phi''(x^{(\text{final})})}}\right\|_\infty \lesssim \varepsilon$.

Proof. We start by using Lemma 4.28 to get that

$$\begin{aligned}\|\Phi''(x)^{\frac12}(x^{(\text{final})}-x)\|_\infty &\le \|\overline{\mathbf{T}}^{-1}\Phi''(x)^{-\frac12}\mathbf{A}(\mathbf{A}^\top\overline{\mathbf{T}}^{-1}\Phi''(x)^{-1}\mathbf{A})^{-1}(b-\mathbf{A}^\top x)\|_\infty \\ &\lesssim \|\mathbf{A}^\top x - b\|_{(\mathbf{A}^\top\mathbf{T}^{-1}\Phi''(x)^{-1}\mathbf{A})^{-1}} \\ &\le \varepsilon\gamma/C_{\text{norm}} \le \varepsilon.\end{aligned}$$

1-self-concordance gives us that

$$\left\|\Phi''(x)^{-\frac12}\left(\phi'(x^{(\text{final})})-\phi'(x)\right)\right\|_\infty \lesssim \|\Phi''(x)^{-\frac12}\Phi''(x)(x^{(\text{final})}-x)\|_\infty \lesssim \varepsilon.$$

Also, we know that $\phi''(x) \approx_2 \phi''(x^{(\text{final})})$, hence

$$\left\|\frac{s^{(\text{final})}+\mu\tau(x)\phi'(x^{(\text{final})})}{\mu\tau(x)\sqrt{\phi''(x^{(\text{final})})}}\right\|_\infty \lesssim \left\|\frac{s^{(\text{final})}+\mu\tau(x)\phi'(x^{(\text{final})})}{\mu\tau(x)\sqrt{\phi''(x)}}\right\|_\infty \lesssim \varepsilon.$$

□

Now, we are ready to prove the second point. Define $x^* \stackrel{\text{def}}{=} \operatorname{argmin}_{\substack{\mathbf{A}^\top x = b \\ \ell_i \le x_i \le u_i\forall i}} c^\top x$, and $x_t = tx^{(\text{final})} + (1-t)x^*$ for $t \in [0,1]$. Let $v = \frac{s^{(\text{final})}+\mu\tau(x)\phi'(x^{(\text{final})})}{\mu\tau(x)\sqrt{\phi''(x^{(\text{final})})}}$, so that $\|v\|_\infty \lesssim \varepsilon$. We will use that $\|v\|_\infty \le 1$. This gives us

$$\begin{aligned}c^\top(x^{(\text{final})}-x^*) &= (s^{(\text{final})}-\mathbf{A}y)^\top(x^{(\text{final})}-x^*) \\ &= (s^{(\text{final})})^\top(x^{(\text{final})}-x^*) \\ &= \mu(\sqrt{\phi''(x^{(\text{final})})}v - \phi'(x^{(\text{final})}))^\top\mathbf{T}(x)(x^{(\text{final})}-x^*) \\ &\le \mu\sum_{i\in[m]}\tau(x)_i\left|\sqrt{\phi_i''(x_i^{(\text{final})})}(x_i^{(\text{final})}-x_i^*)\right| - \mu\phi'(x^{(\text{final})})^\top\mathbf{T}(x)(x^{(\text{final})}-x^*).\end{aligned} \tag{82}$$

By Lemma A.5 above coordinate-wise on the 1-self-concordant functions ϕ_i, we can bound

$$\phi'(x^{(\text{final})})^\top\mathbf{T}(x)(x^{(\text{final})}-x^*)$$

$$
\begin{aligned}
&= 2\phi'(x^{(\text{final})})^\top \mathbf{T}(x)(x^{(\text{final})} - x_{1/2}) \\
&= 2(\phi'(x^{(\text{final})}) - \phi'(x_{1/2}))^\top \mathbf{T}(x)(x^{(\text{final})} - x_{1/2}) + 2\phi'(x_{1/2})^\top \mathbf{T}(x)(x^{(\text{final})} - x_{1/2}) \\
&= 2(\phi'(x^{(\text{final})}) - \phi'(x_{1/2}))^\top \mathbf{T}(x)(x^{(\text{final})} - x_{1/2}) + 2\phi'(x_{1/2})^\top \mathbf{T}(x)(x_{1/2} - x^*) \\
&\geq 2\sum_{i\in[m]} \tau(x)_i \frac{\left|\sqrt{\phi_i''(x_i^{(\text{final})})}(x_i^{(\text{final})} - (x_{1/2})_i)\right|^2}{1 + \left|\sqrt{\phi_i''(x_i^{(\text{final})})}(x_i^{(\text{final})} - (x_{1/2})_i)\right|} - 2\sum_{i\in[m]} \tau(x)_i \\
&= \sum_{i\in[m]} \tau(x)_i \frac{\left|\sqrt{\phi_i''(x_i^{(\text{final})})}(x_i^{(\text{final})} - x_i^*)\right|^2}{2 + \left|\sqrt{\phi_i''(x_i^{(\text{final})})}(x_i^{(\text{final})} - x_i^*)\right|} - 2\sum_{i\in[m]} \tau(x)_i
\end{aligned}
$$

Applying this to the expression in (82) we get that

$$
\begin{aligned}
&c^\top(x^{(\text{final})} - x^*) \leq (82) \\
&\leq \mu\left(\sum_{i\in[m]} \tau(x)_i \left|\sqrt{\phi_i''(x_i^{(\text{final})})}(x_i^{(\text{final})} - x_i^*)\right| - \sum_i \tau(x)_i \frac{\left|\sqrt{\phi_i''(x_i^{(\text{final})})}(x_i^{(\text{final})} - x_i^*)\right|^2}{2 + \left|\sqrt{\phi_i''(x_i^{(\text{final})})}(x_i^{(\text{final})} - x_i^*)\right|} + 2\sum_{i\in[m]} \tau(x)_i\right) \\
&= \mu\left(\sum_{i\in[m]} \tau(x)_i \frac{2\left|\sqrt{\phi_i''(x_i^{(\text{final})})}(x_i^{(\text{final})} - x_i^*)\right|}{2 + \left|\sqrt{\phi_i''(x_i^{(\text{final})})}(x_i^{(\text{final})} - x_i^*)\right|} + 2\sum_{i\in[m]} \tau(x)_i\right) \leq 4\mu \sum_{i\in[m]} \tau(x)_i \lesssim n\mu.
\end{aligned}
$$

□

A.4 Sampling Schemes

Lemma 4.41 (Independent sampling)**.** *Let vector $q \in \mathbb{R}^m_{\geq 0}$ satisfy*

$$
q_i \geq C_{\text{valid}}^2 \gamma^{-1} |(\delta_r)_i| + C_{\text{sample}} \sigma(\overline{\mathbf{T}}^{-\frac{1}{2}} \Phi''(\overline{x})^{-\frac{1}{2}} \mathbf{A})_i \log(m) \gamma^{-2}
$$

for sufficiently large C_{sample}. Then picking $\mathbf{R}_{ii} = 1/\min(q_i, 1)$ with probability $\min(q_i, 1)$ and 0 otherwise is a C_{valid}-valid (Definition 4.13).

Proof. The (Expectation) condition is clear by definition, and (Covariance) follows by independence. The (Matrix approximation) condition follows by [CLM+15, Lemma 4] and that $q_i \geq C_{\text{sample}} \log(m) \gamma^{-2} \sigma(\overline{\mathbf{A}})$.

For the (Variance) condition, note that the variance is 0 if $q_i = 1$. Otherwise, $q_i \geq C_{\text{valid}}^2 \gamma^{-1} |(\delta_r)_i|$ so we have

$$
\text{Var}[\mathbf{R}_{ii}(\delta_r)_i] \leq \frac{(\delta_r)_i^2}{q_i} \leq \frac{\gamma |(\delta_r)_i|}{C_{\text{valid}}^2}
$$

Also, $q_i \geq C_{\text{sample}} \sigma(\overline{\mathbf{T}}^{-\frac{1}{2}} \Phi''(\overline{x})^{-\frac{1}{2}} \mathbf{A})_i \log(m) \gamma^{-2}$ so $\mathbb{E}[\mathbf{R}_{ii}^2] \leq 2\sigma(\overline{\mathbf{A}})^{-1}$.

For the (Maximum) condition, we perform cases on q_i. It is trivial for $q_i = 1$. Otherwise, $q_i \geq C_{\text{valid}}^2 \gamma^{-1} |(\delta_r)_i|$, so $|q_i^{-1}(\delta_r)_i| \leq \frac{\gamma}{C_{\text{valid}}^2}$. □

Lemma 4.42 (Proportional sampling)**.** *Let vector $q \in \mathbb{R}^m_{\geq 0}$ satisfy*

$$
q_i \geq |(\delta_r)_i| + \sigma(\overline{\mathbf{T}}^{-\frac{1}{2}} \Phi''(\overline{x})^{-\frac{1}{2}} \mathbf{A})_i.
$$

Let $S \geq \sum_i q_i$. Let X be a random variable which equals $q_i^{-1} e_i$ (e_i is the standard basis vector) with probability q_i/S for all i, and $\vec{0}$ otherwise. For $C_0 = 100 C_{\text{valid}}^4 \gamma^{-2} \log(m)$ let $\mathbf{R} = C_0^{-1} \sum_{j=1}^{C_0 S} X_j$, where X_j are i.i.d. copies of X. Then $\mathbf{R}$ is a C_{valid}-valid distribution (Definition 4.13).

Proof. The (Expectation) condition follows directly. For the (Covariance) condition, note that

$$\mathbb{E}[\mathbf{R}_{ii}\mathbf{R}_{jj}] = \sum_{1 \leq t_1, t_2 \leq C_0 S} \frac{1}{C_0^2 q_i q_j} \Pr[X_{t_1} \text{ picks } i] \Pr[X_{t_2} \text{ picks } j] \leq C_0^2 S^2 \frac{1}{C_0^2 q_i q_j} \frac{q_i q_j}{S^2} = 1.$$

Because variance is additive over independent samples, we use that $q_i \geq |(\delta_r)_i|$ to get

$$\text{Var}[\mathbf{R}_{ii}(\delta_r)_i] \leq \frac{(\delta_r)_i^2}{C_{\text{valid}}^2 \gamma^{-2} q_i} \leq \frac{\gamma^2 |(\delta_r)_i|}{C_{\text{valid}}^2} \leq \frac{\gamma |(\delta_r)_i|}{C_{\text{valid}}^2}.$$

Also, $q_i \geq \sigma(\overline{\mathbf{A}})_i$, so we get that

$$\mathbb{E}[\mathbf{R}_{ii}^2] \leq 1 + \text{Var}[\mathbf{R}_{ii}] \leq 1 + q_i^{-1} \leq 2\sigma(\overline{\mathbf{A}})^{-1}.$$

To show the (Maximum) condition, we recall that $\mathbf{R} = \sum_{j=1}^{C_0 S} C_0^{-1} X_j$, and use Bernstein's inequality. Note that the maximum possible value of $C_0^{-1} X_j$ is $(C_0 q_i)^{-1}$, and that $\text{Var}[\mathbf{R}_{ii}] \leq (C_0 q_i)^{-1}$. Therefore, by Bernstein's inequality we know that

$$\Pr[|\mathbf{R}_{ii} - 1| \geq t] \leq \exp\left(- \frac{t^2/2}{\frac{1}{C_0 q_i} + \frac{t}{3 C_0 q_i}} \right).$$

For $t = \frac{\gamma}{C_{\text{valid}}^2 |(\delta_r)_i|}$ we have that

$$\exp\left(- \frac{t^2/2}{\frac{1}{C_0 q_i} + \frac{t}{3 C_0 q_i}} \right) \leq \exp\left(- \frac{1}{\frac{(\delta_r)_i^2}{50 q_i \log(m)} + \frac{|(\delta_r)_i| \gamma}{150 C_{\text{valid}}^2 q_i \log(m)}} \right).$$

Now, because $q_i \geq |(\delta_r)_i|$, we have that

$$\frac{(\delta_r)_i^2}{50 q_i \log(m)} \leq \frac{|(\delta_r)_i|}{50 \log(m)} \leq \frac{1}{50 \log(m)}$$

as $\|\delta_r\|_\infty \lesssim \gamma$ by Lemma 4.29. From the definition of q_i, we also calculate that

$$\frac{|(\delta_r)_i| \gamma}{150 C_{\text{valid}}^2 q_i \log(m)} \leq \frac{1}{150 C_{\text{valid}}^2 \log(m)}.$$

Therefore, we have that

$$\begin{aligned} & \exp\left(- \frac{1}{\frac{(\delta_r)_i^2}{50 q_i \log(m)} + \frac{(\delta_r)_i \gamma}{150 C_{\text{norm}} C_{\text{valid}} q_i \log(m)}} \right) \\ \leq \; & \exp\left(\frac{1}{\frac{1}{50 \log(m)} + \frac{1}{150 C_{\text{valid}}^2 \log(m)}} \right) \\ \leq \; & \exp(-25 \log(m)) \leq m^{-10}. \end{aligned}$$

Finally, to show the (Matrix concentration) result, we will use the matrix Freedman Inequality [Tro11]. Let a_i be the rows of the matrix $\overline{\mathbf{A}}$, so we have that

$$(\overline{\mathbf{A}}^\top \overline{\mathbf{A}})^{-1/2} \overline{\mathbf{A}}^\top \mathbf{R} \overline{\mathbf{A}} (\overline{\mathbf{A}}^\top \overline{\mathbf{A}})^{-1/2} = \sum_{j=1}^{C_0 S} (\overline{\mathbf{A}}^\top \overline{\mathbf{A}})^{-1/2} C_0^{-1} q_{i_j}^{-1} a_{i_j} a_{i_j}^\top (\overline{\mathbf{A}}^\top \overline{\mathbf{A}})^{-1/2},$$

where i_j is the i-th selected nonzero entry for $\mathbf{R}$. We now bound the maximum and the variance. For the maximum, we know that

$$\left\|(\overline{\mathbf{A}}^\top\overline{\mathbf{A}})^{-1/2}C_0^{-1}q_{i_j}^{-1}a_{i_j}a_{i_j}^\top(\overline{\mathbf{A}}^\top\overline{\mathbf{A}})^{-1/2}\right\|_2 \le C_0^{-1}q_{i_j}^{-1}\sigma(\overline{\mathbf{A}})_{i_j} \le C_0^{-1},$$

as $q_i \ge \sigma(\overline{\mathbf{A}})_i$ for all $i \in [m]$. For the variance, by the above calculation using $q_i \ge \sigma(\overline{\mathbf{A}})_i$ for all i, we know that

$$\begin{aligned}
&\mathbb{E}_{i_j}\left[\left((\overline{\mathbf{A}}^\top\overline{\mathbf{A}})^{-1/2}C_0^{-1}q_{i_j}^{-1}a_{i_j}a_{i_j}^\top(\overline{\mathbf{A}}^\top\overline{\mathbf{A}})^{-1/2}\right)\right]\\
&= \sum_j C_0^{-2}q_i^{-2}\cdot q_i/S\cdot\left((\overline{\mathbf{A}}^\top\overline{\mathbf{A}})^{-1/2}a_ja_j^\top(\overline{\mathbf{A}}^\top\overline{\mathbf{A}})^{-1/2}\right)^2\\
&\preceq C_0^{-2}S^{-1}\sum_j(\overline{\mathbf{A}}^\top\overline{\mathbf{A}})^{-1/2}a_ja_j^\top(\overline{\mathbf{A}}^\top\overline{\mathbf{A}})^{-1/2} = C_0^{-2}S^{-1}\mathbf{I}.
\end{aligned}$$

As there are C_0S total terms, the variance is bounded by $C_0^{-1}\mathbf{I}$. Hence, the matrix Freedman inequality tells us that

$$\Pr\left[\overline{\mathbf{A}}^\top\mathbf{R}\overline{\mathbf{A}} \not\approx_\gamma \overline{\mathbf{A}}^\top\overline{\mathbf{A}}\right] \le m\cdot\exp\left(-\frac{\gamma^2/2}{C_0^{-1}+\gamma C_0/3}\right) \le m\cdot\exp(-25\log(m)) \le m^{-10}$$

as desired, by the choice of C_0. □

Corollary 4.43 (Sampling by a mixture of ℓ_2 and uniform)**.** *Let C_1, C_2, C_3 be constants such that $C_3 \ge 4C_{\text{sample}}$ and $C_1C_2 \ge C_{\text{valid}}^4\gamma^{-2}$ and*

$$p_i = C_1\sqrt{n}(\delta_r)_i^2 + C_2/\sqrt{n} + C_3\tau_i\gamma^{-2}\log m.$$

Then $p_i \ge q_i$ in each of Lemma 4.41 and 4.42. Hence replacing q_i with p_i in Lemma 4.41 and 4.42 and sampling acoordingly gives a valid distribution (Definition 4.13). Additionally

$$\sum_{i\in[m]} p_i \le \left((C_1+C_2)\frac{m}{\sqrt{n}} + C_3n\gamma^{-2}\log m\right). \tag{25}$$

Proof. Note that by the choice of $\alpha = \frac{1}{4\log(4m/n)}$ we have that

$$\sigma(\overline{\mathbf{T}}^{-\frac12}\Phi''(\overline{x})^{-\frac12}\mathbf{A})_i \approx_1 \sigma(\overline{\mathbf{T}}^{-\frac12-\frac{1}{1-\alpha}}\Phi''(\overline{x})^{-\frac12}\mathbf{A})_i = \tau(\overline{x})_i,$$

which gives the bound for the part with τ_i. For the other piece, note by the AM-GM inequality

$$C_1\sqrt{n}(\delta_r)_i^2 + C_2/\sqrt{n} \ge 2\sqrt{C_1C_2}|\delta_r|_i \ge C_{\text{valid}}^2\gamma^{-1}|\delta_r|_i.$$

Now we bound $\sum_i p_i$. By Lemma 4.29 we have that

$$\|\delta_r\|_2^2 \le \frac{m}{n}\|\delta_r\|_\tau^2 \le 2m\gamma^2/n \le m/n$$

because $\gamma \le 1/2$. Therefore, we have that

$$\begin{aligned}
\sum_{i\in[m]} p_i &= \sum_{i\in[m]} C_1\sqrt{n}(\delta_r)_i^2 + C_2/\sqrt{n} + C_3\tau_i\gamma^{-2}\log m\\
&= C_1\sqrt{n}\|\delta_r\|_2^2 + C_2\frac{m}{\sqrt{n}} + C_3n\gamma^{-2}\log m\\
&\le (C_1+C_2)\frac{m}{\sqrt{n}} + C_3n\gamma^{-2}\log m.
\end{aligned}$$

□

A.5 Additional IPM Properties

Lemma 4.44 (Nearby stability of x)**.** *Suppose that* $\mathbf{R}$ *is sampled from a* C_{valid}*-valid distribution for* $C_{\text{valid}} \geq \beta^{-2}\log(mT)$ *where* $\beta \in (0,\gamma)$*. Let* $(x^{(k)}, s^{(k)})$ *for* $k \in [T]$ *be the sequence of points found by Algorithm 1. With probability* $1 - m^{-10}$*, there is a sequence of points* $\widehat{x}^{(k)}$ *from* $1 \leq k \leq T$ *such that*

- $\|\Phi''(x^{(k)})^{\frac{1}{2}}(\widehat{x}^{(k)} - x^{(k)})\|_\infty \leq \beta/2$.
- $\|\Phi''(\widehat{x}^{(k)})^{\frac{1}{2}}(\widehat{x}^{(k)} - x^{(k)})\|_\infty \leq \beta$.
- $\|\Phi''(x^{(k)})^{\frac{1}{2}}(\widehat{x}^{(k+1)} - \widehat{x}^{(k)})\|_{\tau(\widehat{x}^{(k)})+\infty} \leq 2\gamma$.

Proof. Define $\widehat{x}^{(1)} = x^{(1)}$. Define the *stability potential*, analogous to the centrality potential in Definition 4.8, as

$$\Psi_{\text{stab}}(x,\widehat{x}) \stackrel{\text{def}}{=} \sum_{i\in[m]} \cosh\left(\lambda_{\text{stab}}\phi''(\widehat{x}_i^{(k)})_i^{\frac{1}{2}}(\widehat{x}_i^{(k)} - x_i^{(k)})\right)$$

for $\lambda_{\text{stab}} = C\log(mT)/\beta$ for sufficiently large constant C. We will choose $\widehat{x}^{(k+1)}$ using gradient descent against the potential, and will analyze the procedure using Lemma 4.15. Precisely, fix $\varepsilon_{\text{stab}} = \frac{1}{C\lambda_{\text{stab}}}$ for C as the same constant as in Algorithm 1. Chosen this way, we can see that $\gamma\beta \leq \varepsilon_{\text{stab}}$ because $\varepsilon_{\text{stab}} = \frac{\beta}{C^2\log(mT)}$ and $\gamma = \varepsilon/(\lambda C) \leq 1/(C^3\log m)$ by the choice of parameters in Algorithm 1.

Define $\delta_{\widehat{x}} = \varepsilon_{\text{stab}}\nabla_{\widehat{x}}\Psi(x^{(k)},\widehat{x}^{(k)})^{\flat(\tau(\widehat{x}^{(k)}))}$ and

$$\widehat{x}^{(k+1)} = \widehat{x}^{(k)} - \mathbb{E}[x^{(k+1)} - x^{(k)}] - \Phi''(x^{(k)})^{-\frac{1}{2}}\delta_{\widehat{x}} = \widehat{x}^{(k)} - \mathbb{E}[\overline{\delta}_x] - \Phi''(x^{(k)})^{-\frac{1}{2}}\delta_{\widehat{x}}$$

where $\overline{\delta}_x$ is defined in Algorithm 1 line 12. We will now verify the conditions of Lemma 4.15 and apply it. Here, we will choose $y = \widehat{x}^{(k)} - x^{(k)}$ and $u^{(1)} = \Phi''(x^{(k)})^{\frac{1}{2}}$. In the notation of Lemma 4.15, for simplicity we will just write $c = u^{(1)}$, and $\delta_c = \delta^{(1)}$, as in the notation of Section 4.4. By Lemma 4.31 we know that $\|\mathbf{C}^{-1}\delta_c\|_\infty \leq 2\gamma \leq 1/100$.

Throughout the proof, we will use that $\tau(\widehat{x}^{(k)}) \approx_{O(\beta)} \tau(x^{(k)})$ because $\Phi''(\widehat{x}^{(k)}) \approx_{O(\beta)} \Phi''(x^{(k)})$ by induction and Lemma 4.22. In particular, for any vector h we have that $\|h\|_{\tau(\widehat{x}^{(k)})+\infty} \approx_{0.1} \|h\|_{\tau+\infty}$.

By induction, we know that $\|\Phi''(x^{(k)})^{\frac{1}{2}}(\widehat{x}^{(k)} - x^{(k)})\|_\infty \leq \beta/2 \leq 1/50$, and thus

$$\|\mathbf{C}^{-1}\delta_c\|_\infty\|\Phi''(x^{(k)})^{\frac{1}{2}}(\widehat{x}^{(k)} - x^{(k)})\|_\infty \leq \beta\gamma \leq \frac{1}{100\lambda_{\text{stab}}}$$

by the choice of γ. Also, we know that

$$\eta = \widehat{x}^{(k+1)} - \widehat{x}^{(k)} - (x^{(k+1)} - x^{(k)}) - \Phi''(x^{(k)})^{-\frac{1}{2}}\delta_{\widehat{x}} = \Phi''(x^{(k)})^{-\frac{1}{2}}\left(\mathbf{R}\delta_r - \delta_r - \delta_{\widehat{x}}\right).$$

Therefore, by the (Maximum) condition of Definition 4.13 and $\|\delta_{\widehat{x}}\|_{\tau(\widehat{x}^{(k)})+\infty} \leq \varepsilon_{\text{stab}}$ we know that

$$\|\mathbf{W}\eta\|_\infty \leq \|\Phi''(x^{(k)})^{\frac{1}{2}}(\widehat{x}^{(k+1)} - \widehat{x}^{(k)})\|_\infty \leq \frac{\gamma}{C_{\text{valid}}^2} + \varepsilon_{\text{stab}} \leq \gamma\beta + \varepsilon_{\text{stab}} \leq \frac{1}{100\lambda_{\text{stab}}},$$

so all the conditions of Lemma 4.15 are satisfied. Now we bound the terms of (13), (14), (15) in expectation over $\mathbf{R}$. To bound (13), note that

$$\mathbb{E}[\psi'(v)^\top\mathbf{W}\eta] = \psi'(v)^\top\delta_{\widehat{x}} = -\varepsilon_{\text{stab}}\|\psi'(v)\|^*_{\tau(\widehat{x}^{(k)})+\infty}$$

and by Lemma 4.31

$$\mathbb{E}[\psi'(v)^\top\mathbf{V}\mathbf{C}^{-1}\delta_c] \leq \|v\|_\infty\|\psi'(v)\|^*_{\tau(\widehat{x}^{(k)})+\infty}\|\mathbf{C}^{-1}\mathbb{E}[\delta_c]\|_{\tau(\widehat{x}^{(k)})+\infty}$$

$$\leq \beta\gamma\|\psi'(v)\|^*_{\tau(\widehat{x}^{(k)})+\infty} \leq \frac{1}{4}\varepsilon_{\text{stab}}\|\psi'(v)\|^*_{\tau(\widehat{x}^{(k)})+\infty}.$$

For (14) we have that

$$\begin{aligned}
\mathbb{E}[8\|\mathbf{W}\eta\|^2_{\psi''(v)}] &\lesssim \|\text{Var}(\delta_r)^{1/2}\|^2_{\psi''(v)} + \|\delta^2_{\widehat{x}}\|^2_{\psi''(v)} \\
&\leq \gamma\beta^2\|\delta_r\|_{\tau(\widehat{x}^{(k)})+\infty}\|\psi''(v)\|^*_{\tau(\widehat{x}^{(k)})+\infty} + \|\delta^2_{\widehat{x}}\|_{\tau(\widehat{x}^{(k)})+\infty}\|\psi''(v)\|^*_{\tau(\widehat{x}^{(k)})+\infty} \\
&\lesssim (\gamma^2\beta^2 + \varepsilon^2_{\text{stab}})\|\psi''(v)\|^*_{\tau(\widehat{x}^{(k)})+\infty} \\
&\lesssim \varepsilon^2_{\text{stab}}\|\psi''(v)\|^*_{\tau(\widehat{x}^{(k)})+\infty}.
\end{aligned}$$

where the first step is via the triangle inequality and the definition of η, the second step is by the (Variance) condition of Definition 4.13 for $C_{\text{valid}} \geq \beta^{-2}$ and the definition of $\|\cdot\|^*_{\tau+\infty}$, the third step is from Lemma 4.30 Part 2 and $\|\delta_{\widehat{x}}\|_{\tau(\widehat{x}^{(k)})+\infty} \leq \varepsilon_{\text{stab}}$ by Definition, and the final step is by $\gamma\beta \leq \varepsilon_{\text{stab}}$.

Also, we can bound that

$$\begin{aligned}
&\mathbb{E}[8(1+\|c\|_1)\|v\|^2_\infty \sum_{j\in[k]} |c_j|\|(\mathbf{U}^{(j)})^{-1}\delta^{(j)}\|^2_{\psi''(v)}] \\
\lesssim\ & \beta^2\mathbb{E}[\|\mathbf{C}^{-1}\delta_c\|^2_{\psi''(v)}] \\
=\ & \beta^2\|\mathbb{E}[\mathbf{C}^{-2}\delta_c^2]^{1/2}\|^2_{\psi''(v)} \\
\leq\ & \beta^2\|\mathbb{E}[\mathbf{C}^{-2}\delta_c^2]\|_{\tau(\widehat{x}^{(k)})+\infty}\|\psi''(v)\|^*_{\tau(\widehat{x}^{(k)})+\infty} \\
\lesssim\ & \beta^2\gamma^2\|\psi''(v)\|^*_{\tau(\widehat{x}^{(k)})+\infty} \\
\leq\ & \varepsilon^2_{\text{stab}}\|\psi''(v)\|^*_{\tau(\widehat{x}^{(k)})+\infty}.
\end{aligned}$$

where the first step follows from $\|v\|_\infty \leq \beta$ by induction, $|c_j| = O(1)$ for all j, and the definition of $\delta^{(j)}$, the third step follows from the definition of the dual norm $\|\cdot\|^*_{\tau+\infty}$, the fourth step follows from Lemma 4.31 Part 3, and the final step follows from $\beta\gamma \leq \varepsilon_{\text{stab}}$.

For (15) we can bound using the Cauchy-Schwarz inequality and the above computations that

$$\begin{aligned}
&\mathbb{E}[8\|\mathbf{W}\eta\|_{|\psi'(v)|}\sum_{j\in[k]}|c_j|\|(\mathbf{U}^{(j)})^{-1}\delta^{(j)}\|_{|\psi'(v)|}] \\
\lesssim\ & \mathbb{E}[\|\mathbf{W}\eta\|^2_{|\psi'(v)|}]^{1/2}\mathbb{E}\left[\left(\sum_{j\in[k]}|c_j|\|(\mathbf{U}^{(j)})^{-1}\delta^{(j)}\|_{|\psi'(v)|}\right)^2\right]^{1/2} \\
\lesssim\ & (\varepsilon^2_{\text{stab}}\beta\gamma^2)^{1/2}\|\psi'(v)\|^*_{\tau(\widehat{x}^{(k)})+\infty}. \qquad (83)
\end{aligned}$$

Also, we have that

$$\begin{aligned}
&\mathbb{E}\left[8(1+\|c\|_1)\|v\|_\infty\sum_{j\in[k]}|c_j|\|(\mathbf{U}^{(j)})^{-1}\delta^{(j)}\|^2_{|\psi'(v)|}\right] \\
\lesssim\ & \beta\|\mathbb{E}[\mathbf{C}^{-2}\delta_c^2]^{1/2}\|_|\psi'(v)|^2 \\
\leq\ & \beta\|\mathbb{E}[\mathbf{C}^{-2}\delta_c^2]\|_{\tau(\widehat{x}^{(k)})+\infty}\|\psi'(v)\|^*_{\tau(\widehat{x}^{(k)})+\infty} \\
\lesssim\ & \beta\gamma^2\|\psi'(v)\|^*_{\tau(\widehat{x}^{(k)})+\infty}. \qquad (84)
\end{aligned}$$

For sufficiently small choice of γ, we have that even with the suppressed constants in (83), (84) that

$$O\left(\beta\gamma^2 + (\varepsilon^2_{\text{stab}}\beta\gamma^2)^{1/2}\right) \leq \frac{1}{4}\varepsilon_{\text{stab}}.$$

Therefore combining everything, we get that

$$\mathbb{E}[\Psi(x^{(k+1)},\widehat{x}^{(k+1)})] \le \Psi(x^{(k)},\widehat{x}^{(k)}) - \frac{1}{4}\varepsilon_{\text{stab}}\|\psi'(v)\|^*_{\tau(\widehat{x}^{(k)})+\infty} + O(\varepsilon^2_{\text{stab}})\|\psi''(v)\|^*_{\tau(\widehat{x}^{(k)})+\infty}.$$

As in the proof of Lemma 4.40, by [BLN+20, Lemma 4.36], and the fact that $\|1\|_{\tau(\widehat{x}^{(k)})+\infty} \le 4C_{\text{norm}}\sqrt{n}$, we get that

$$\begin{aligned}\mathbb{E}[\Psi(x^{(k+1)},\widehat{x}^{(k+1)})] &\le \left(1 - \frac{\lambda_{\text{stab}}\varepsilon_{\text{stab}}}{4C_{\text{norm}}\sqrt{n}}\right)\Psi(x^{(k)},\widehat{x}^{(k)}) + m\\ &\le \left(1 - \frac{1}{4C_{\text{norm}}C\sqrt{n}}\right)\Psi(x^{(k)},\widehat{x}^{(k)}) + m.\end{aligned}$$

As $\Psi(x^{(1)},\widehat{x}^{(1)}) = m$, we have by induction that $\mathbb{E}[\Psi(x^{(k)},\widehat{x}^{(k)})] \le 4C_{\text{norm}}Cm\sqrt{n} \le m^2$ by induction for all k. Therefore, with probability $1 - m^{-12}$ we have that $\Psi(x^{(k)},\widehat{x}^{(k)}) \le m^{14}$ for all k. By the choice of λ_{stab} this implies that $\|\Phi''(x^{(k)})^{\frac{1}{2}}(\widehat{x}^{(k)} - x^{(k)})\|_\infty \le \beta/2$ as desired.

To finish we must verify the other two conditions. The second item follows from self-concordance and the first. To check the third condition, we have that

$$\|\Phi''(x^{(k)})^{\frac{1}{2}}(\widehat{x}^{(k+1)} - \widehat{x}^{(k)})\|_{\tau(\widehat{x}^{(k)})+\infty} = \left\|\Phi''(x^{(k)})^{\frac{1}{2}}\mathbb{E}[\overline{\delta}_x] - \delta_{\widehat{x}}\right\|_{\tau(\widehat{x}^{(k)})+\infty} \le 1.1\gamma + \varepsilon_{\text{stab}} \le \gamma$$

where the first inequality follows by the triangle inequality and Lemma 4.29 Part 1. as $\varepsilon \le \frac{\beta}{C\log(mT)} \le 0.1\gamma$ as $\beta \le \gamma$. □

Lemma 4.45 (Nearby stability of ϕ'' and τ)**.** *In the setup of Lemma 4.44, and* $\widehat{w}^{(k)} = \Phi''(\widehat{x}^{(k)})^{-\frac{1}{2}}\tau(\widehat{x}^{(k)})^{\frac{1}{2}-\frac{1}{p}}$*, we have the following.*

- $\|\Phi''(\widehat{x}^{(k)})^{\frac{1}{2}}(\phi''(\widehat{x}^{(k)})^{-\frac{1}{2}} - \phi''(x^{(k)})^{-\frac{1}{2}})\|_\infty \le \beta.$
- $\|\Phi''(\widehat{x}^{(k)})^{\frac{1}{2}}(\phi''(\widehat{x}^{(k+1)})^{-\frac{1}{2}} - \phi''(\widehat{x}^{(k)})^{-\frac{1}{2}})\|_{\tau(\widehat{x}^{(k)})+\infty} \lesssim \gamma.$
- $\|\mathbf{T}(\widehat{x}^{(k)})^{-1}(\tau(\widehat{x}^{(k+1)}) - \tau(\widehat{x}^{(k)}))\|_{\tau(\widehat{x}^{(k)})+\infty} \lesssim \gamma.$
- $\|(\widehat{\mathbf{W}}^{(k)})^{-1}(\widehat{w}^{(k+1)} - \widehat{w}^{(k)})\|_{\tau(\widehat{x}^{(k)})+\infty} \lesssim \gamma.$

Proof. The first two items directly follow from 1-self-concordance, specifically that

$$|\phi''(\widehat{x}^{(k)})^{-\frac{1}{2}} - \phi''(x^{(k)})^{-\frac{1}{2}}| \le |\widehat{x}^{(k)} - x^{(k)}|$$

and Lemma 4.44.

For the third item, define $p_0 = \phi''(x^{(k)})^{-\frac{1}{2}}$, $p_1 = \phi''(x^{(k+1)})^{-\frac{1}{2}}$, and $\delta_p = p_1 - p_0$. Define $p_t = p_0 + t\cdot\delta_p$ and $\tau_t = w(p_t)$. Note that $\tau_t \approx_{0.04} \tau_0 = \tau(\widehat{x}^{(k)})$ for all $t \in [0,1]$ as $p_t \approx_{0.01} p_0$ by the second item of this lemma (Lemma 4.45), and Lemma 4.22. Therefore we can compute

$$\begin{aligned}\|\mathbf{T}(\widehat{x}^{(k)})^{-1}(\tau(\widehat{x}^{(k+1)}) - \tau(\widehat{x}^{(k)}))\|_{\tau(\widehat{x}^{(k)})} &= \left\|\int_0^1 \mathbf{T}(\widehat{x}^{(k)})^{-1}\mathbf{J}_{w(p_t)}\delta_p\right\|_{\tau(\widehat{x}^{(k)})+\infty}\\ &\lesssim \int_0^1 \left\|\mathbf{T}_t^{-1}\mathbf{J}_{w(p_t)}\delta_p\right\|_{\tau_t+\infty}\\ &\lesssim \|\mathbf{P}_t^{-1}\delta_p\|_{\tau_t+\infty}\\ &\lesssim \|\mathbf{P}^{-1}\delta_p\|_{\tau(\widehat{x}^{(k)})+\infty} \lesssim \gamma\end{aligned}$$

where we have used Lemma A.2.

For the fourth item, we once again use that $\tau_1 \approx_{0.04} \tau_0$ and $p_t \approx_{0.01} p_0$. This gives us

$$\|(\widehat{\mathbf{W}}^{(k)})^{-1}(\widehat{w}^{(k+1)} - \widehat{w}^{(k)})\|_{\tau(\widehat{x}^{(k)})+\infty}$$

$$
\begin{aligned}
&= \|\Phi''(\widehat{x}^{(k)})^{\frac{1}{2}}\mathbf{T}(\widehat{x}^{(k)})^{\frac{1}{p}-\frac{1}{2}}\left(\phi''(\widehat{x}^{(k+1)})^{-\frac{1}{2}}\tau(\widehat{x}^{(k+1)})^{\frac{1}{2}-\frac{1}{p}} - \phi''(\widehat{x}^{(k)})^{-\frac{1}{2}}\tau(\widehat{x}^{(k)})^{\frac{1}{2}-\frac{1}{p}}\right)\|_{\tau(\widehat{x}^{(k)})+\infty} \\
&\leq \|\Phi''(\widehat{x}^{(k)})^{\frac{1}{2}}\mathbf{T}(\widehat{x}^{(k)})^{\frac{1}{p}-\frac{1}{2}}\left(\phi''(\widehat{x}^{(k+1)})^{-\frac{1}{2}} - \phi''(\widehat{x}^{(k)})^{-\frac{1}{2}}\right)\tau(\widehat{x}^{(k+1)})^{\frac{1}{2}-\frac{1}{p}}\|_{\tau(\widehat{x}^{(k)})+\infty} \\
&+ \|\Phi''(\widehat{x}^{(k)})^{\frac{1}{2}}\mathbf{T}(\widehat{x}^{(k)})^{\frac{1}{p}-\frac{1}{2}}\phi''(\widehat{x}^{(k)})^{-\frac{1}{2}}\left(\tau(\widehat{x}^{(k+1)})^{\frac{1}{2}-\frac{1}{p}} - \tau(\widehat{x}^{(k)})^{\frac{1}{2}-\frac{1}{p}}\right)\|_{\tau(\widehat{x}^{(k)})+\infty} \\
&\lesssim \|\Phi''(\widehat{x}^{(k)})^{\frac{1}{2}}\left(\phi''(\widehat{x}^{(k+1)})^{-\frac{1}{2}} - \phi''(\widehat{x}^{(k)})^{-\frac{1}{2}}\right)\|_{\tau(\widehat{x}^{(k)})+\infty} \\
&+ \|\mathbf{T}(\widehat{x}^{(k)})^{\frac{1}{p}-\frac{1}{2}}\left(\tau(\widehat{x}^{(k+1)})^{\frac{1}{2}-\frac{1}{p}} - \tau(\widehat{x}^{(k)})^{\frac{1}{2}-\frac{1}{p}}\right)\|_{\tau(\widehat{x}^{(k)})+\infty} \\
&\lesssim \gamma + \|(\tau(\widehat{x}^{(k+1)})/\tau(\widehat{x}^{(k)}))^{\frac{1}{2}-\frac{1}{p}} - 1\|_{\tau(\widehat{x}^{(k)})+\infty} \\
&\leq \gamma + \|(\tau(\widehat{x}^{(k+1)})/\tau(\widehat{x}^{(k)})) - 1\|_{\tau(\widehat{x}^{(k)})+\infty} \\
&= \gamma + \|\mathbf{T}(\widehat{x}^{(k)})^{-1}(\tau(\widehat{x}^{(k+1)}) - \tau(\widehat{x}^{(k)}))\|_{\tau(\widehat{x}^{(k)})+\infty} \lesssim \gamma.
\end{aligned}
$$

Here we have used that $|x^{\frac{1}{2}-\frac{1}{p}} - 1| \leq |x-1|$ for all $x \in [0.9, 1.1]$, and items two and three of this lemma (Lemma 4.45). □

Lemma 4.46 (Parameter changes along central path)**.** *For $\mathbf{A} \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^n, c \in \mathbb{R}^m$ and $\ell, u \in \mathbb{R}^m$ assume that the point $x^{(\text{init})} = (\ell+u)/2$ is feasible, i.e. $\mathbf{A}^\top x^{(\text{init})} = b$. Let W be the ratio of the largest to smallest entry of $\phi''(x^{(\text{init})})^{1/2}$, and let W' be the ratio of the largest to smallest entry of $\phi''(x)^{1/2}$ encountered in Algorithm 2. Then*

$$\log W' = \widetilde{O}\left(\log W + \log(1/\mu^{(\text{final})}) + \log\|c\|_\infty\right).$$

Proof. Note that the smallest possible entry of W and W' is at least $\min_i(u_i-\ell_i)^{-1}$, as $\phi_i''(x) \geq \frac{1}{(u_i-\ell_i)^2}$ for all $x \in (\ell_i, u_i)$. By Lemma 4.11 and Claim A.6, for any ε-centered (x, s, μ) encountered in the algorithm, we can find a point $x^{(\text{final})}$ such that $(x^{(\text{final})}, s, \mu)$ is also ε-centered, $\Phi''(x^{(\text{final})}) \approx_1 \Phi''(x)$, and $\mathbf{A}^\top x^{(\text{final})} = b$, i.e. $x^{(\text{final})}$ is exactly feasible. Thus, it suffices to control the largest entry of $\Phi''(x^{(\text{final})})$ to bound $\log W'$.

Let $v = \frac{s+\mu\tau\phi'(x^{(\text{final})})}{\mu\tau\phi''(x^{(\text{final})})^{1/2}}$, so that $\|v\|_\infty \lesssim \varepsilon$. We will use that $\|v\|_\infty \leq 1/100$ say. Also, we know that $s = \mathbf{A}z + c$ for some $z \in \mathbb{R}^n$. This gives us that

$$\mathbf{A}z + c + \mu\tau\phi'(x^{(\text{final})}) - v\mu\tau\phi''(x^{(\text{final})})^{1/2} = 0.$$

Computing an inner product with $(x^{(\text{final})} - x^{(\text{init})})$ gives that

$$
\begin{aligned}
0 &= (x^{(\text{final})} - x^{(\text{init})})^\top \mathbf{A}z + c^\top(x^{(\text{final})} - x^{(\text{init})}) \qquad (85)\\
&+ \mu\sum_i \tau_i\left(\phi_i'(x_i^{(\text{final})}) - v_i\phi_i''(x_i^{(\text{final})})^{1/2}\right)(x_i^{(\text{final})} - x_i^{(\text{init})}) \\
&= c^\top(x^{(\text{final})} - x^{(\text{init})}) + \mu\sum_i \tau_i\left(\phi_i'(x_i^{(\text{final})}) - v_i\phi_i''(x_i^{(\text{final})})^{1/2}\right)(x_i^{(\text{final})} - x_i^{(\text{init})}). \qquad (86)
\end{aligned}
$$

We claim that $\left(\phi_i'(x_i^{(\text{final})}) - v_i\phi_i''(x_i^{(\text{final})})^{1/2}\right)(x_i^{(\text{final})} - x_i^{(\text{init})}) \geq -1$ for all i. To show this, we without loss of genearlity assume that $x_i^{(\text{final})} \geq x_i^{(\text{init})}$. Note that $\phi_i'(x_i^{(\text{final})})(x_i^{(\text{final})} - x_i^{(\text{init})})$ by our choice of $\phi_i(x) = -\log(u_i - x) - \log(x-\ell_i)$ and $x^{(\text{init})} = (\ell+u)/2$. If $x_i^{(\text{final})} \geq (x_i^{(\text{init})} + u_i)/2$, then note that

$$\phi_i'(x_i^{(\text{final})}) - v_i\phi_i''(x_i^{(\text{final})})^{1/2} \geq \frac{1}{2(u_i - x_i^{(\text{final})})} - v_i\sqrt{\frac{2}{(u_i - x_i^{(\text{final})})^2}} \geq \frac{1}{4(u_i - x_i^{(\text{final})})} > 0,$$

so the claim is trivially true. So the remaining case if $x_i^{(\text{init})} \leq x_i^{(\text{final})} \leq (x_i^{(\text{init})} + u_i)/2$. In this case, we have that

$$\left(\phi_i'(x_i^{(\text{final})}) - v_i\phi_i''(x_i^{(\text{final})})^{1/2}\right)(x_i^{(\text{final})} - x_i^{(\text{init})}) \geq -\frac{1}{100}|\phi_i''(x_i^{(\text{final})})^{1/2}(x_i^{(\text{final})} - x_i^{(\text{init})})|$$

$$\geq -\frac{1}{10}\sqrt{\frac{1}{(u_i-\ell_i)^2}}(u_i-\ell_i) \geq -1.$$

Going back to (86), we have for all $j \in [m]$ that

$$\begin{aligned}
&\mu\tau_j\left(\phi_j'(x_j^{(\text{final})}) - v_j\phi_j''(x_j^{(\text{final})})^{1/2}\right)(x_j^{(\text{final})} - x_j^{(\text{init})}) \\
&= -c^\top(x^{(\text{final})} - x^{(\text{init})}) - \mu\sum_{i\in[m]\setminus\{i\}}\tau_i\left(\phi_i'(x_i^{(\text{final})}) - v_i\phi_i''(x_i^{(\text{final})})^{1/2}\right)(x_i^{(\text{final})} - x_i^{(\text{init})}) \\
&\leq m\|c\|_\infty\|u-\ell\|_\infty + \mu\sum_j\tau_j \leq m\|c\|_\infty\|u-\ell\|_\infty + \mu n.
\end{aligned} \tag{87}$$

We now will bound $\phi_j''(x_j^{(\text{final})})$. We will assume without loss of generality that $x_j^{(\text{final})} \geq x_j^{(\text{init})}$. If $x_j^{(\text{final})} \leq (x_j^{(\text{init})} + u_j)/2$, then we know that $\phi_j''(x_j^{(\text{final})}) \lesssim (u_i-\ell_i)^{-2}$, as desired. Otherwise, the above arguments give that

$$\left(\phi_j'(x_j^{(\text{final})}) - v_j\phi_j''(x_j^{(\text{final})})^{1/2}\right) \geq \frac{1}{4(u_j - x_j^{(\text{final})})}.$$

Therefore, we get that

$$\begin{aligned}
(u_j - x_j^{(\text{final})})^{-1} &\lesssim \mu^{-1}\tau_j^{-1}(x_j^{(\text{final})} - x_j^{(\text{init})})^{-1}(m\|c\|_\infty\|u-\ell\|_\infty + \mu n) \\
&\leq \mu^{-1}\frac{m}{n}(u_j-\ell_j)^{-1}(m\|c\|_\infty\|u-\ell\|_\infty + \mu n).
\end{aligned}$$

Using that $\phi''(x_j^{(\text{final})}) \lesssim (u_j - x_j^{(\text{final})})^{-2}$ completes the proof. □

B Matrix Data Structures

In this section we provide the existence of the required data structure for solving general linear programs. We start by citing the HeavyHitter- (Definition 3.1) and InverseMaintenance-data structure (Definition 6.2) proven in [BLSS20]. We then prove the existence of a HeavySampler-data structure (Definition 6.3).

Lemma B.1 ([BLSS20, Section 6.1]). *There exists a (P, c, Q)-HeavyHitter data structure (Definition 3.1) with $P = \widetilde{O}(\text{nnz}(\mathbf{A}))$, $c_i = \widetilde{O}(n)$ for all $i \in [m]$, and $Q = \widetilde{O}(n)$.*

Lemma B.2 ([BLSS20, Theorem 9]). *There exists a (P, c, Q)-InverseMaintenance data structure (Definition 6.2) with $P = \widetilde{O}(\text{nnz}(\mathbf{A}) + n^\omega)$, $c_i = O(1)$ for all $i \in [m]$, and $Q = \widetilde{O}(n^2 + n^{\omega-1/2})$.*

We now construct a data structure for the HeavySampler-problem (Definition 6.3). We first provide the data structure Algorithm 6 with guarantees given by the following, Lemma B.3. We then show in Corollary B.6 how the data structure of Lemma B.3 can be used to solve the HeavySampler-problem.

Lemma B.3. *There exists a data structure (Algorithm 6) that supports the following operations.*

- *Initialize($\mathbf{A} \in \mathbb{R}^{m\times n}, v \in \mathbb{R}^m_{\geq 0}, g \in \mathbb{R}^m$): The data structure is given a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, additive vector $v \in \mathbb{R}^m_{\geq 0}$, and a scaling vector $g \in \mathbb{R}^m$. It initializes in $\widetilde{O}(\text{nnz}(\mathbf{A}))$ time.*
- *Scale($i \in [m], s \geq 0, b \geq 0$): Update $g_i \leftarrow s$ and $v_i \leftarrow b$ in $\widetilde{O}(\text{nnz}(a_i))$ time.*
- *Sample($h \in \mathbb{R}^n, U \in \mathbb{R}_{>0}$): If $U \geq e^4\|\mathbf{GA}h\|_2^2$ then, with high probability, in $\widetilde{O}(n)$ time, the data structure returns a random $i \in [m]$ with $\mathbb{P}[i = j] = (\mathbf{GA}h)_j^2/U$, and with probability $1 - \|\mathbf{GA}h\|_2^2/U$, returns nothing. Each random index returned by a call to Sample is independent from the previous call.*

Algorithm 6: Sampler data structure.

`// [M]^{l,j} denotes the matrix formed by rows` $(j-1)m/2^l+1, \cdots, jm/2^l$ `of` $\mathbf{M}$.

members

$\mathbf{A} \in \mathbb{R}^{m \times n}$, $g \in \mathbb{R}^m$ `// Assume` m `is a power of 2 for code simplicity`

$c = 1/\log(4m)$, $k = O(c^{-2} \log m)$

for $l \in \{0, 1, \cdots, \log m\}$ **do**

  for $j \in [2^l]$ **do**

    $\mathbf{J}^{l,j} \in \mathbb{R}^{k \times m/2^l}$ `//` $\mathbf{J}^{l,j} = \mathrm{JL}(c, m/2^l)$ `in Lemma B.4`

    $\mathbf{Q}^{l,j} \in \mathbb{R}^{k \times d}$ `//` $\mathbf{Q}^{l,j} = \mathbf{J}^{l,j} \cdot [\mathbf{GA}]^{l,j}$

procedure *Initialize*$(\mathbf{A} \in \mathbb{R}^{m \times n}, g \in \mathbb{R}^m)$

  $\mathbf{A} \leftarrow \mathbf{A}$, $g \leftarrow g$, $v \leftarrow v$

  for $l \in \{0, 1, \cdots, \log m\}$ **do**

    for $j \in [2^l]$ **do**

      $\mathbf{J}^{l,j} \leftarrow \mathrm{JL}(c, m/2^l)$ `// See Lemma B.4`

      $\mathbf{Q}^{l,j} \leftarrow \mathbf{J}^{l,j} \cdot [\mathbf{GA}]^{l,j}$

procedure *Scale*$(i \in [m], s \geq 0)$

  for $l \in \{0, 1, \cdots, \log m\}$ **do**

    $j = \lceil i \cdot 2^l / m \rceil$

    $\mathbf{Q}^{l,j} \leftarrow \mathbf{Q}^{l,j} + (s - g_i)\mathbf{J}^{l,j} \cdot [\mathbf{1}_i \mathbf{1}_i^\top \mathbf{A}]^{l,j}$

  $g_i \leftarrow s$

procedure *Sample*$(h \in \mathbb{R}^n, U \in \mathbb{R}_{>0})$

  return SampleInternal$(h, 1, 1, 0, U)$

procedure *SampleInternal*$(h \in \mathbb{R}^n, Z \geq 0, j, \ell, U \in \mathbb{R}_{>0})$

  if $\ell = \log m$ **then**

    With probability $Z^{-1} \cdot \frac{(\mathbf{GA} \cdot h)_j^2}{U}$ **return** j, otherwise **return** $\emptyset$

  else

    $r_1 \leftarrow \|\mathbf{Q}^{l+1,2j-1} h\|_2^2$, $r_2 \leftarrow \|\mathbf{Q}^{l+1,2j} h\|_2^2$

    With probability $r_1/(r_1 + r_2)$,

      return SampleInternal$(h, Z r_1/(r_1 + r_2), 2j - 1, \ell + 1)$

    otherwise

      return SampleInternal$(h, Z r_2/(r_1 + r_2), 2j, \ell + 1)$

First, we state a lemma used in the proof of Lemma B.3.

Lemma B.4 (Johnson–Lindenstrauss (JL) [JL84])**.** *There exists a function* $\mathrm{JL}(\epsilon, m)$ *that given* $\epsilon > 0$ *returns a matrix* $\mathbf{J} \in \mathbb{R}^{k \times m}$ *with* $k = O(\epsilon^{-2} \log m)$ *in* $O(km)$ *time. For any* $v \in \mathbb{R}^m$ *this matrix* $\mathbf{J}$ *satisfies with high probability in* m *that* $\|\mathbf{J}v\|_2 \approx_\epsilon \|v\|_2$.

Using Lemma B.4 allows us to give an algorithm for sampling coordinates proportional to their ℓ_2 weight (Lemma B.5). First, we show how to use this to prove Lemma B.3 by analyzing the runtime costs and showing how to apply a Scale operation. Then we show Lemma B.5, where the high-level approach is to build a binary tree and walk down the tree using the JL lemma to sample a node proportional to the ℓ_2 norm of its coordinates.

Lemma B.5. *A call to* *SampleInternal*$(h, 1, 1, 0, U)$ *for* $e^4 \|\mathbf{GA}h\|_2^2 \leq U$ *randomly returns a single index* $i \in [m]$*, or no index at all. Index* $i \in [m]$ *is returned with probability*

$$p_i = \frac{(\mathbf{GA}h)_i^2}{U}.$$

Proof of Lemma B.3. The implementation of our data structure is given in Algorithm 6. We now analyze the correctness and efficiency of the operations.

Initialize. We bound the running time of INITIALIZE. For each $l \in 0, 1, \cdots \log m$ and $j \in [2^l]$ we instantiate a matrix $\mathbf{J}^{l,j} = \mathrm{JL}(\epsilon, m/2^l)$ with $\epsilon = 1/\log m$, so each matrix $\mathbf{J}^{l,j}$ has $k = O(c^{-2} \log m)$ rows. Creating a $k \times m/2^l$ sized JL matrix takes $O(km/2^l)$ time (Lemma B.4), so in total creating the JL matrices takes

$$\sum_{l=0}^{\log m} 2^l \cdot O(km/2^l) = O(km \log^3 m) = \widetilde{O}(m).$$

We use $[\mathbf{GA}]^{l,j}$ to denote the submatrix of $\mathbf{GA}$ consisting of the $\{(j-1)m/2^l + 1, \cdots, j \cdot m/2^l\}$-th rows of $\mathbf{GA}$. We also use $V^{l,j}$ to denote the sum of v_i for $i \in \{(j-1)m/2^l + 1, \cdots, j \cdot m/2^l\}$ which can be computed in $\widetilde{O}(m)$ time. For any fixed l, computing the matrices $\mathbf{Q}^{l,j} = \mathbf{J}^{l,j} \cdot [\mathbf{GA}]^{l,j} \in \mathbb{R}^{k \times n}$ for all $j \in [m/2^l]$ takes $O(k \cdot \mathrm{nnz}(\mathbf{A}))$ time since $\mathbf{J}^{l,j} \in \mathbb{R}^{k \times m/2^l}$ and the $[\mathbf{GA}]^{l,j} \in \mathbb{R}^{m/2^l \times n}$ are just a decomposition of the matrix $\mathbf{GA}$. Since we create $\mathbf{Q}^{l,j}$ for every $l \in \{0, \cdots, \log m\}$ and $j \in [2^l]$, in total constructing all the $\mathbf{Q}$ matrices takes time

$$\sum_{l=0}^{\log m} O(k \cdot \mathrm{nnz}(\mathbf{A})) = O(k \, \mathrm{nnz}(\mathbf{A}) \cdot \log^3 m) = \widetilde{O}(\mathrm{nnz}(\mathbf{A})).$$

Thus initialization can be done in $\widetilde{O}(\mathrm{nnz}(\mathbf{A}))$ time.

Scale. We first prove $\mathbf{Q}^{l,j} = \mathbf{J}^{l,j} \cdot [\mathbf{GA}]^{l,j}$ is still satisfied after updating g_i to be s. For any $l \in \{0, \cdots, \log m\}$, we only need to update the $\mathbf{Q}^{l,j}$ with $j = \lceil \frac{i}{m/2^l} \rceil$ (since $[\mathbf{GA}]^{l,j}$ has rows of $\mathbf{GA}$ in set $\{(j-1)m/2^l + 1, \cdots, j \cdot m/2^l\}$. We have

$$\begin{aligned} \mathbf{Q}^{l,j} &= \mathbf{J}^{l,j} \cdot [\mathbf{GA}]^{l,j} + (s - g_i)\mathbf{J}^{l,j} \cdot [\mathbf{1}_i \mathbf{1}_i^\top \mathbf{A}]^{l,j} \\ &= \mathbf{J}^{l,j} \cdot [(\mathbf{G} + (s - g_i)\mathbf{1}_i \mathbf{1}_i^\top)\mathbf{A}]^{l,j}, \end{aligned}$$

where the first step follows from how we update $\mathbf{Q}^{l,j}$ (Line 16 of Algorithm 6), the second step follows from merging terms. And $(\mathbf{G} + (s - g_i)\mathbf{1}_i \mathbf{1}_i^\top)$ is indeed the updated scaling vector whose i-th coordinate is s. Note that we also update $V^{l,j}$ to be the partial sum of the v_i for $i \in \{(j-1)m/2^l + 1, \cdots, j \cdot m/2^l\}$.

Next we bound the running time of SCALE. We need to compute $\mathbf{J}^{l,j} \cdot [\mathbf{1}_i \mathbf{1}_i^\top \mathbf{A}]^{l,j}$ for $l \in \{0, \cdots, \log m\}$ and one j that depends on l, i. Since $\mathbf{J}^{l,j} \in \mathbb{R}^{k \times m/2^l}$ and $[\mathbf{1}_i \mathbf{1}_i^\top \mathbf{A}]^{l,j} \in \mathbb{R}^{m/2^l \times n}$ only has one non-zero row, consisting of $\mathrm{nnz}(a_i)$ non-zero entries, this multiplication takes $O(k \, \mathrm{nnz}(a_i))$ time. Thus in total computing the multiplication for all l takes time

$$O(\log n) \cdot O(k \, \mathrm{nnz}(a_i)) = \widetilde{O}(\mathrm{nnz}(a_i)),$$

which follows from $k = O(c^{-2} \log m)$.

Sample. By Line 19 and Lemma B.5 we return a randomly sampled index j according to the distribution $\mathbb{P}[j = i] = (\mathbf{GA}h)_i^2 / U$ for all $i \in [m]$.

A call to SAMPLEINTERNAL requires $\widetilde{O}(n)$ time, because we compute $O(\log m)$ norms which each require a matrix-vector-product with an $\widetilde{O}(1) \times n$ matrix in Line 24.

□

Proof of Lemma B.5. Consider the binarytree where nodes are labeled by sub-intervalls of $[m]$. The root is labeled by $[m]$. For each node labeled by some $[L, R]$, the left child is labeled by $[L, \lfloor (L+R)/2 \rfloor]$ and the right node is labeled by $[\lceil (L+R+1)/2 \rceil, R]$. We also identify the j-th node on level l via the tuple (l, j).

Note that the execution of SAMPLEINTERNAL can be seen as a path from the root of this tree to one of its leaves. For each node (i, j) it picks the left child with probability $r_1/(r_1 + r_2)$ and otherwise the right child.

We arrive at the index $j \in [m]$ on level $\ell = \log(m)$ (i.e. a leaf) with probability as follows, note that we are basically looking at the binary representation of j and multiplying together the probabilities that the half containing j is selected in each level.

$$\begin{aligned}
p &= \prod_{l=1}^{\log m} \frac{\|\mathbf{Q}^{l,\lceil j/(m/2^l)\rceil} \cdot h\|_2^2}{\|\mathbf{Q}^{l,2\lceil j/(m/2^{l-1})\rceil} \cdot h\|_2^2 + \|\mathbf{Q}^{l,2\lceil j/(m/2^{l-1})\rceil - 1} \cdot h\|_2^2} \\
&= \prod_{l=1}^{\log m} e^{\pm 4c} \frac{\|[\mathbf{GA}h]^{l,\lceil j/(m/2^l)\rceil}\|_2^2}{\|[\mathbf{GA}h]^{l,2\lceil j/(m/2^{l-1})\rceil}\|_2^2 + \|[\mathbf{GA}h]^{l,2\lceil j/(m/2^{l-1})\rceil - 1}\|_2^2} \\
&= e^{\pm 4c \cdot \log m} \prod_{l=1}^{\log m} \frac{\|[\mathbf{GA}h]^{l,\lceil j/(m/2^l)\rceil}\|_2^2}{\|[\mathbf{GA}h]^{l-1,\lceil j/(m/2^{l-1})\rceil}\|_2^2} \\
&= e^{\pm 4} \cdot \frac{\|[\mathbf{GA}h]^{\log m, j}\|_2^2}{\|[\mathbf{GA}h]^{0,1}\|_2^2 + V^{0,1}} = e^{\pm 4} \frac{(\mathbf{GA}h)_j^2 + v_j}{\|\mathbf{GA}h\|_2^2}
\end{aligned}$$

where the second step follows from the JL matrices guarantee that $\forall l$, $\forall j$, with high probability

$$e^{-2c}\|[\mathbf{GA}]^{l,j}h\|_2^2 \le \|\mathbf{Q}^{l,j} \cdot h\|_2^2 = \|\mathbf{J}^{l,j} \cdot [\mathbf{GA}]^{l,j}h\|_2^2 \le e^{2c}\|[\mathbf{GA}]^{l,j}h\|_2^2,$$

and the fourth step follows from $c = 1/\log(4m)$. The other steps simply follow the definition of $[\mathbf{GA}h]^{l,j}$. Note that the variable Z tracks exactly this probability so in Line 22 we have $Z = p$. This also means $(\mathbf{GA}h)_j^2/U \le e^{-4} \cdot (\mathbf{GA}h)_j^2/\|\mathbf{GA}h\|_2^2 \le Z$ so the rejection sampling defined in Line 22 is well defined. For any j, the probability of it being returned by SampleInternal$(h, 1, 1, 0)$ is thus $(\mathbf{GA}h)_j^2/U$ □

We now show how to combine the data structure of Lemma B.3 with an additional sampling step to obtain a HeavySampler data structure.

Corollary B.6. *There exists a (P, c, Q)-HeavySampler data structure for matrices $\mathbf{A} \in \mathbb{R}^{m \times n}$ with $P = \widetilde{O}(\mathrm{nnz}(\mathbf{A}))$, $c_i = \widetilde{O}(n)$ for all $i \in [m]$, and $Q = \widetilde{O}(n^2 + m\sqrt{n})$.*

Proof. By Lemma 4.42 and Corollary 4.43 we need to be able to efficiently construct the following distribution: For some constants C_1, C_2, C_3 as in Corollary 4.43, let $q \in \mathbb{R}^m$, $S \in \mathbb{R}$ such that

$$q_i \ge C_1\sqrt{n}(\mathbf{GA}h)_i^2 + C_2/\sqrt{n} + C_3\tau_i\gamma^2 \log m \tag{88}$$

and $S \ge \sum_{i=1}^m q_i$. Let X be a random variable with $X = q_i^{-1}\vec{e}_i$ with probability q_i/S for all i, and 0 otherwise.

Generating X We now describe how to efficiently generate this random X. Let $v \in \mathbb{R}^m_{\ge 0}$ with

$$v_i = \frac{1}{4}\left(\frac{1}{m + n^{1.5}} + 1.5\frac{\overline{\tau}_i}{\frac{m}{\sqrt{n}} + n}\right)$$

where $\overline{\tau}$ is the current approximation of the Lewis weights. Further define $U = m/n + \sqrt{n}$ and note that $\|\mathbf{GA}h\|_2^2 \le m/n < U$ by guarantee of Definition 6.3 (the definition of HeavySampler).

Flip a balanced coin and then either sample an index j with $P[i = j] = (\mathbf{GA}h)_i^2/U$ via Lemma B.3, or sample the index by $P[i = j] = v_i$. We then return

$$X = \vec{e}_i \cdot (C(\sqrt{n}(\mathbf{GA}h)_i^2 + \frac{1}{4\sqrt{n}} + 1.5\overline{\tau}_i/4))^{-1}$$

for $C := \max\{C_1, C_2, C_3\gamma^2 \log m\}$. Note that we might sample no index at all, in which case we return $X = 0$. This procedure can be implemented to take $\widetilde{O}(n)$ time by Lemma B.3.

Correctness We now prove that the X generated in the paragraph above satisfies that $X = q_i^{-1}\vec{e}_i$ with probability q_i/S. For that we define the following

$$q_i := C(\sqrt{n}(\mathbf{G}\mathbf{A}h)_i^2 + \frac{1}{4\sqrt{n}} + 1.5\overline{\tau}_i/4)$$
$$S := 2CU\sqrt{n}.$$

Here we clearly have (88) by definition of $C := \max\{C_1, C_2, C_3\gamma^2 \log m\}$. We also have $S \geq \sum_i q_i$ by

$$\sum_{i=1}^m q_i = \sum_{i=1}^m C(\sqrt{n}(\mathbf{G}\mathbf{A}h)_i^2 + \frac{1}{4\sqrt{n}} + 1.5\overline{\tau}_i/4) = C(\sqrt{n}\|\mathbf{G}\mathbf{A}h\|_2^2 + \frac{m}{4\sqrt{n}} + n/2) \leq 2C(\frac{m}{\sqrt{n}} + n) \leq S.$$

Note that the random X we generated in the previous paragraph is $X = q_i^{-1}\vec{e}_i$ with probability $1/2 \cdot ((\mathbf{G}\mathbf{A}h)_i^2/U + v_i)$. We now show that this probability is q_i/S:

$$\begin{aligned} q_i/S &= S^{-1} \cdot C(\sqrt{n}(\mathbf{G}\mathbf{A}h)_i^2 + \frac{1}{4\sqrt{n}} + 1.5\overline{\tau}_i/4) \\ &= 1/2 \cdot ((\mathbf{G}\mathbf{A}h)_i^2/U + \frac{1}{4Un} + 1.5\frac{\overline{\tau}_i}{4U\sqrt{n}}) \\ &= 1/2 \cdot ((\mathbf{G}\mathbf{A}h)_i^2/U + \frac{1}{4(m+n^{1.5})} + 1.5\frac{\overline{\tau}_i}{4(\frac{m}{\sqrt{n}}+n)}) \\ &= 1/2 \cdot ((\mathbf{G}\mathbf{A}h)_i^2/U + v_i), \end{aligned}$$

where the first step uses the definition of q_i, the second step uses the definition of S, the third step uses the definition of U and the last step uses the definition of v_i. Thus we indeed have $X = q_i^{-1}\vec{e}_i$ with probability q_i/S.

Final Complexity The complexity parameters $P := \widetilde{O}(\mathrm{nnz}(A))$ and $c_i := \widetilde{O}(n)$ stem from Lemma B.3. Generating one random X takes $\widetilde{O}(n)$ time and we must generate $\widetilde{O}(S) = \widetilde{O}(m/\sqrt{n}+n)$ many independent copies of X by Lemma 4.42, which results in a total sampling complexity of $Q := \widetilde{O}(m\sqrt{n} + n^2)$.

□

C Leverage Score

In this section we show how to efficiently maintain an approximation of the leverage scores $\sigma(\mathbf{V}\mathbf{A})$ under updates to $\mathbf{V} = \mathbf{Diag}(v)$. The data structure is obtained via reduction to a HeavyHitter-data structure.

Theorem C.1. *Assume there exists a (P, c, Q)-HeavyHitter data structure (Definition 3.1). Then there exists a Monte-Carlo data-structure (Algorithm 7), that works against an adaptive adversary, with the following procedures:*

- *Initialize($\mathbf{A} \in \mathbb{R}^{m\times n}, v \in \mathbb{R}^m, z \in \mathbb{R}^m, \epsilon \in (0,1)$): The data structure initializes on a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, scaling $v \in \mathbb{R}^m$, target accuracy $\epsilon > 0$, and regularization parameter $z \in \mathbb{R}^m$ with $z \geq n/m + nc/\|c\|_1$ (where c is the parameter of the heavy hitter data structure) and returns a vector $\overline{\sigma} \in \mathbb{R}^m$ with $\overline{\sigma} \approx_\epsilon \sigma(\mathbf{V}\mathbf{A}) + z$.*
- *Scale($i \in [m], c \in \mathbb{R}_{\geq 0}$): For given $c \approx_{0.25} v_i$, set $v_i \leftarrow c$.*
- *Query(): W.h.p. in n the data-structure outputs a vector $\overline{\sigma} \in \mathbb{R}^m$ such that $\overline{\sigma} \approx_\epsilon \sigma(\mathbf{V}\mathbf{A}) + z$. The vector $\overline{\sigma}$ is returned as a pointer and the data structure also returns a set $I \subset [m]$ of indices i where $\overline{\sigma}_i$ has changed compared to the last call to Query.*

The amortized complexities of our data structure depends on the parameters P, c, Q of the HeavyHitter data structure. Further, our complexity bounds require that some additional properties are satisfied. These properties and the resulting amortized complexities are stated in Theorem C.2.

Theorem C.2. *Consider the data structure of Theorem C.1 (Algorithm 7) and let P, c, Q be the parameters of the HeavyHitter data structure (Definition 3.1). Let $v^{(t)}$ be the vector v in Theorem 5.1 during the t-th call to Query (and $v^{(0)}$ during the initialization). Further, assume the following:*

1. *For any given $\overline{\mathbf{V}} \in \mathbb{R}^m_{\geq 0}$ we can solve linear systems in $(\mathbf{A}^\top \overline{\mathbf{V}} \mathbf{A})^{-1}$ with $\epsilon/(64n)$ accuracy (i.e. for input b we can output $\overline{\mathbf{H}} b$ for some $\overline{\mathbf{H}} \approx_{\epsilon/(64n)} (\mathbf{A}^\top \overline{\mathbf{V}} \mathbf{A})^{-1}$) in $\widetilde{O}(P+\Psi+\mathrm{nnz}(\overline{\mathbf{V}}\mathbf{A}))$ time. If*
$$\mathbf{A}^\top \overline{\mathbf{V}} \mathbf{A} \approx_{1/(64 \log n)} \mathbf{A}^\top (\mathbf{V}^{(t)})^2 \mathbf{A}$$
for some t, then the required time is only $\widetilde{O}(\Psi + \mathrm{nnz}(\overline{\mathbf{V}}\mathbf{A}))$.
2. *There exists a sequence $\widetilde{v}^{(t)}$ such that for all t*
$$v^{(t)} \in (1 \pm 1/(64 \log n))\widetilde{v}^{(t)} \tag{89}$$
$$\|(\widetilde{v}^{(t)})^{-1}(\widetilde{v}^{(t)} - \widetilde{v}^{(t+1)})\|_{\sigma(\widetilde{\mathbf{V}}^{(t)}\mathbf{A})} = O(1) \tag{90}$$
for all t.

Then the following time complexities hold:

- *Initialize takes $\widetilde{O}(P + \epsilon^{-2}(\Psi + \mathrm{nnz}(\mathbf{A})))$ time.*
- *Scale$(i, \cdot)$ takes $\widetilde{O}(\frac{\|c\|_1}{n\epsilon^4}\sigma(\mathbf{VA})_i + \frac{c_i}{\epsilon^2})$ amortized time.*
- *Query takes $\widetilde{O}(\Psi\epsilon^{-2} + \epsilon^{-4} n(\max_i \mathrm{nnz}(a_i)) + \epsilon^{-2}\sqrt{\frac{P\|c\|_1}{n}} + Q)$. amortized time.*

The high-level idea of Algorithm 7 (Theorem C.1) is as follows: For a set $I \subset [m]$, the method UpdateIndices(I) computes for all $i \in I$ an approximation $\overline{\sigma}'_i \approx \sigma(\mathbf{VA})_i + z_i$. If $\overline{\sigma}_i \not\approx \overline{\sigma}'_i$, then we set $\overline{\sigma}_i \leftarrow \overline{\sigma}'_i$. So for all $i \in I$ we have $\overline{\sigma}_i \approx \sigma(\mathbf{VA})_i + z_i$ after a call to UpdateIndices(I), as proven in Lemma C.7. Thus if the set I contains all indices where $\overline{\sigma}_i \not\approx \sigma(\mathbf{VA})_i + z_i$, then $\overline{\sigma}$ will be a valid approximation after the execution of UpdateIndices(I).

The set I is constructed in method FindIndices as follows: Fix some $T \in \mathbb{N}$. For $j = 0, ..., \log T$, the data structure checks every 2^j calls to Query, if $\sigma(\mathbf{VA})_i + z_i$ changed a lot compared to its value 2^j calls to Query in the past. This claim is proven in Lemma C.6 and we prove in Lemma C.9 that such a set I suffices to maintain $\overline{\sigma} \approx \sigma(\mathbf{VA}) + z$ for up to T iterations. After T iterations the data structure restarts.

C.1 Correctness

As the vector v for which we want to maintain $\sigma(\mathbf{VA})$ changes over time, we use the following notation during the proof of Theorem C.1 to specify which instance of v we refer to:

Definition C.3. *Write $v^{(t)}$ for the vector v during the t-th call to Query and $v^{(0)}$ for the vector v as given during the initialization. Likewise write $\overline{\sigma}^{(t)}$ for the vector $\overline{\sigma}$ returned by the t-th call to Query.*

So we want to prove that $\overline{\sigma}^{(t)} \approx_\epsilon \sigma(\mathbf{V}^{(t)}\mathbf{A}) + z$. We will prove this via induction, so let the following Proposition C.4 be the induction hypothesis. Note that for $t = 0$ the claim is true as seen by the initialization procedure (Line 5).

Proposition C.4. *During the t-th call to Query we have $\overline{\sigma}^{(t-1)} \approx_\epsilon \sigma(\mathbf{V}^{(t-1)}\mathbf{A}) + z$, i.e. the result of the* previous *call to Query was correct.*

Algorithm 7: Data structure for maintaining leverage scores

members
  $t, T, r \in \mathbb{N}$, $v \in \mathbb{R}^m$, $\Delta^{(j)} \in \mathbb{R}^m$ and $S_j, C_j \subset [m]$ for $j = 1, ..., 0.5 \log n$.
procedure *Initialize*($\mathbf{A}, v^{(\text{init})}, z, \epsilon$)
  $t \leftarrow 0$, $v \leftarrow v^{(\text{init})}$, $z \leftarrow z$, $T \leftarrow \lceil \epsilon^2 \sqrt{Pn/\|c\|_1} \rceil$
  Compute $\overline{\sigma} \approx_\epsilon \sigma(\mathbf{VA}) + z$
  Let $r = O(\log n)$ be such that a $r \times m$ JL-matrix yields a 1/2-approximation.
  for $j = 0, ..., T$ **do**
    D_j.Initialize($\mathbf{A}, v \cdot z^{-1/2}$)
    $S_j \leftarrow \emptyset$
    $C_j \leftarrow \emptyset$
    $\Delta^{(j)} \leftarrow \vec{0}_m$
  return $\overline{\sigma}$
procedure *FindIndices*($h \in \mathbb{R}^n$)
  $I \leftarrow \emptyset$
  for $j = \log T, ..., 0$ **do**
    if $2^j | t$ **then**
      $\mathbf{S}_{i,i} = 1$ for $i \in S_j \cup C_j$, and for other i we set $\mathbf{S}_{i,i} = 1/p_i$ with probability $p_i = \min(1,\ c\overline{\sigma}_i \epsilon^{-2} \log n \log\log n)$ for some large constant $c > 0$, and $\mathbf{S}_{i,i} = 0$ otherwise.
      $\mathbf{R} \leftarrow m \times r$ JL-matrix
      Let $\mathbf{M} \approx_{1/(64n)} (\mathbf{A}^\top (\mathbf{V} - \Delta^{(j)})^2 \mathbf{S}^2 \mathbf{A})^{-1}$ and $\mathbf{M} \approx_{1/(64n)} (\mathbf{A}^\top \mathbf{V}^2 \mathbf{S}^2 \mathbf{A})^{-1}$
      $\mathbf{H} \leftarrow (\mathbf{M}' \mathbf{A}^\top (\mathbf{V} - \Delta^{(j)}) \mathbf{SR} - \mathbf{MA}^\top \mathbf{VSR}$
      $I \leftarrow I \cup D_j$.HeavyQuery($\mathbf{H}\vec{e}_k, \epsilon/(48r \log n)$) for all $k \in [r]$
      $\Delta^{(j)} \leftarrow \vec{0}_m$
      D_j.Scale($i, v_i z_i^{-1/2}$) for $i \in S_j$
      $I \leftarrow I \cup S_j$, $S_j \leftarrow \emptyset$, $C_j \leftarrow \emptyset$
  return I
procedure *UpdateIndices*(I)
  $\mathbf{S}_{i,i} = 1/\sqrt{p_i}$ with probability $p_i = \min(1,\ c\epsilon^{-2} \overline{\sigma}_i \log n)$ for some large enough constant $c > 0$, and $\mathbf{S}_{i,i} = 0$ otherwise.
  $\mathbf{R} \leftarrow \exp(\pm\epsilon/16)$-accurate JL-matrix
  $\mathbf{H} \leftarrow \mathbf{MA}^\top \mathbf{VSR}$ for any $\mathbf{M} \approx_{\epsilon/(64 \log n)} (\mathbf{A}^\top \mathbf{V}^2 \mathbf{S}^2 \mathbf{A})^{-1}$
  $I' \leftarrow \emptyset$
  for $i \in I$ **do**
    if $\overline{\sigma}_i \not\approx_{3\epsilon/8} \|e_i^\top \mathbf{VAH}\|_2^2 + z_i$ **then**
      $\overline{\sigma}_i \leftarrow \|e_i^\top \mathbf{VAH}\|_2^2 + z_i$
      $C_j \leftarrow C_j \cup \{i\}$ for $j = 0, ..., \log n$
      $I' \leftarrow I' \cup \{i\}$
  return I'
procedure *Scale*(i, c)
  for $j = 0, ..., \log T$ **do**
    $S_j \leftarrow S_j \cup \{i\}$
    D_j.Scale($i, 0$)
    $\Delta^{(j)} \leftarrow \Delta^{(j)} + c - v_i$ `// Maintain` $v^{(t)} = v^{(t_j)} + \Delta^{(j)}$
  $v_i \leftarrow c$
procedure *Query*()
  if $t = T$ **then return** $[m]$, Initialize($\mathbf{A}, v, w, \epsilon$);
  $t \leftarrow t + 1$
  $I \leftarrow$ FindIndices()
  $I \leftarrow$ UpdateIndices(I)
  return I, $\overline{\sigma}$

During the t-th call to QUERY, the algorithm needs access to past $v^{(k)}$ for $k < t$. In order to not process an m-dimensional vector in every iteration, these vectors are maintained implicitly by the data structure as stated in Lemma C.5.

Lemma C.5. *When executing Line 20 we have* $v^{(t)} = v^{(t-2^j)} + \Delta^{(j)}$.

Proof. Let t_j be the last time we set $\Delta^{(j)} \leftarrow \vec{0}_m$. We set $\Delta^{(j)} \leftarrow \vec{0}_m$ either during initialization (so $t_j = 0$) or in Line 22 (FINDINDICES) when $2^j | t$, so right after executing Line 22 we have $t_j = t - 2^j$.

Further, whenever v_i is increased by some $\delta \in \mathbb{R}$, we add δ also to $\Delta_i^{(j)}$ for all j in Line 41, so we always have $v_i^{(t)} = v^{(t_j)} + \Delta^{(j)}$.

Thus in Line 20 we have $v_i^{(t)} = v^{(t-2^j)} + \Delta^{(j)}$. □

Next, we prove the previous claim of the data structure's outline regarding FINDINDICES. We claimed that every 2^j calls to FINDINDICES for any $j = 1, ..., \log T$, the function returns a set $I \subset [m]$ of indices i where $\sigma(\mathbf{V}^{(t)}\mathbf{A})_i + z_i$ changed a lot compared to $\sigma(\mathbf{V}^{(t-2^j)}\mathbf{A})_i + z_i$.

Lemma C.6. *Let $I \subset [m]$ be the set returned by FINDINDICES($h \in \mathbb{R}^n$). Then w.h.p set I contains all indices i where for any $0 \leq j \leq \log T$ we have $2^j | t$ and*

$$\sigma(\mathbf{V}^{(t)}\mathbf{A})_i + z_i \not\approx_{\epsilon/(4\log n)} \sigma(\mathbf{V}^{(t-2^j)})_i + z_i. \tag{91}$$

Proof. Note that for j with $w^j | t$ the set I contains all indices $i \in S_j$ by Line 24. So we only need to consider the remaining indices $i \notin S_j$. Let $\mathbf{Z} = \mathbf{Diag}(z)$ and let $\mathbf{F}$ be diagonal with $\mathbf{F}_{i,i} = 1$ for $i \notin S_j$. I contains all indices i where for some k we have $(\mathbf{F}\mathbf{Z}^{-1/2}\mathbf{V}^{(t-2^j)}\mathbf{A}\mathbf{H}\vec{e}_k)_i > \epsilon/(48r\log n)$. (because we called D_j.SCALE$(i, 0)$ whenever i was added to S_j Line 40, and otherwise we call D_j.SCALE$(i, v_i/\sqrt{z_i})$ for $i \in S_j$ in Line 23).

When $\mathbf{R}$ in Line 18 has $r = O(\log n)$ columns (and thus $\mathbf{H}$ has r columns), then if $\|\vec{e}_i^\top \mathbf{F}\mathbf{Z}^{-1/2}\mathbf{V}^{(t-2^j)}\mathbf{A}\mathbf{H}\|_2^2 > \epsilon^2/(48^2\log^2 n)$, then $|\vec{e}_i^\top \mathbf{F}\mathbf{Z}^{-1/2}\mathbf{V}^{(t-2^j)}\mathbf{A}\mathbf{H}\vec{e}_k| > \epsilon/(48r\log n)$ for some k, so i is added to set I in Line 21. So we must show that if (91) is satisfied, then $\|\vec{e}_i^\top \mathbf{F}\mathbf{Z}^{-1/2}\mathbf{V}^{(t-2^j)}\mathbf{A}\mathbf{H}\|_2^2 > \epsilon^2/(48^2\log^2 n)$.

By Line 20 we have

$$\begin{aligned}
\|\vec{e}_i^\top \mathbf{F}\mathbf{Z}^{-1/2}\mathbf{V}^{(t-2^j)}\mathbf{A}\mathbf{H}\|_2 &= \|\vec{e}_i^\top \mathbf{F}\mathbf{Z}^{-1/2}\mathbf{V}^{(t-2^j)}\mathbf{A}\left(\mathbf{M}'\mathbf{A}^\top\mathbf{V}^{(t-2^j)}\mathbf{S} - \mathbf{M}\mathbf{A}^\top\mathbf{V}^{(t)}\mathbf{S}\right)\mathbf{R}\|_2 \\
&\geq 0.5\|\vec{e}_i^\top \mathbf{F}\mathbf{Z}^{-1/2}\mathbf{V}^{(t-2^j)}\mathbf{A}\left(\mathbf{M}'\mathbf{A}^\top\mathbf{V}^{(t-2^j)}\mathbf{S} - \mathbf{M}\mathbf{A}^\top\mathbf{V}^{(t)}\mathbf{S}\right)\|_2 \\
&\geq 0.5 z_i^{-1/2}\Big|\ (\|\vec{e}_i^\top \mathbf{V}^{(t-2^j)}\mathbf{A}\mathbf{M}'\mathbf{A}^\top\mathbf{V}^{(t-2^j)}\mathbf{S}\|_2^2 + z_i)^{0.5} \\
&\quad - (\|\vec{e}_i^\top \mathbf{V}^{(t)}\mathbf{A}\mathbf{M}\mathbf{A}^\top\mathbf{V}^{(t)}\mathbf{S}\|_2^2 + z_i)^{0.5}\Big|
\end{aligned}$$

For the second step we used that $\mathbf{R}$ is a JL-matrix such that for any $w \in \mathbb{R}^m$ we have w.h.p $\|w^\top\mathbf{R}\|_2 \geq 0.5\|w\|_2$. The third step uses triangle inequality and that we only consider $i \notin S_j$, so $v_i^{(t)} = v_i^{(t-2^j)}$.

Note that set C_j contains all indices for which the leverage score has changed by Line 34. Thus for all $k \notin C_j$ we have that $\overline{\sigma}_k$ is a constant factor approximation of $\sigma(\mathbf{V}^{(t-1)}\mathbf{A})_k + z_k$ and $\sigma(\mathbf{V}^{(t-2^j)}\mathbf{A})_k + z_k$. Further, since $v^{(t)} \approx_1 v^{(t-1)}$ by assumption on SCALE (see Theorem C.1) we have that $\overline{\sigma}_k$ is also a constant factor approximation of $\sigma(\mathbf{V}^{(t)}\mathbf{A})_k + z_k$. Hence we have that $\mathbf{S}_{k,k} = 1/\sqrt{p_k}$ with probability p_k and 0 otherwise, where

$$p_k \geq \min\{1, \max\{\sigma(\mathbf{V}^{(t)}\mathbf{A})_k, \sigma(\mathbf{V}^{(t-2^j)}\mathbf{A})_k\} c\epsilon^{-2}\log n \log\log n\}$$

for some large enough constant $c > 0$. Thus with high probability we have

$$\mathbf{M} \approx_{1/(64\log n)} (\mathbf{A}^\top(\mathbf{V}^{(t)})^2\mathbf{S}^2\mathbf{A})^{-1} \approx_{\epsilon/(64\log n)} (\mathbf{A}^\top(\mathbf{V}^{(t)})^2\mathbf{A})^{-1}$$

and likewise $\mathbf{M}' \approx_{1/(32\log n)} (\mathbf{A}^\top(\mathbf{V}^{(t-2^j)})^2\mathbf{A})^{-1}$, because $\mathbf{S}$ is a valid leverage score sample for both matrices. Finally, this means that

$$\begin{aligned}\sigma(\mathbf{V}^{(t)}\mathbf{A})_i &= \|\vec{e}_i^\top \mathbf{V}^{(t)}\mathbf{A}(\mathbf{A}^\top(\mathbf{V}^{(t)})^2\mathbf{A})^{-1}\mathbf{A}^\top\mathbf{V}^{(t)}\|_2^2 \\ &\approx_{\epsilon/(16\log n)} \|\vec{e}_i^\top \mathbf{V}^{(t)}\mathbf{A}\mathbf{M}\mathbf{A}^\top\mathbf{V}^{(t)}\mathbf{S}\|_2^2\end{aligned}$$

and likewise for $t-2^j$. By $z_i \geq n/m$ we thus have

$$\begin{aligned}&\sigma(\mathbf{V}^{(t)}\mathbf{A})_i + z_i \not\approx_{\epsilon/(4\log n)} \sigma(\mathbf{V}^{(t-2^j)}\mathbf{A})_i + z_i \\ \Rightarrow\ &\|\vec{e}_i^\top \mathbf{V}^{(t)}\mathbf{A}(\mathbf{A}^\top(\mathbf{V}^{(t)})^2\mathbf{A})^{-1}\mathbf{A}^\top\mathbf{V}^{(t)}\|_2^2 + z_i \\ &\not\approx_{\epsilon/(4\log n)} \|\vec{e}_i^\top \mathbf{V}^{(t-2^j)}\mathbf{A}(\mathbf{A}^\top(\mathbf{V}^{(t-2^j)})^2\mathbf{A})^{-1}\mathbf{A}^\top\mathbf{V}^{(t-2^j)}\|_2^2 + z_i \\ \Rightarrow\ &\|\vec{e}_i^\top \mathbf{V}^{(t)}\mathbf{A}\mathbf{M}\mathbf{A}^\top\mathbf{V}^{(t)}\mathbf{S}\|_2^2 + z_i \\ &\not\approx_{\epsilon/(8\log n)} \|\vec{e}_i^\top \mathbf{V}^{(t-2^j)}\mathbf{A}\mathbf{M}'\mathbf{A}^\top\mathbf{V}^{(t-2^j)}\mathbf{S}\|_2^2 + z_i \\ \Rightarrow\ &(\|\vec{e}_i^\top \mathbf{V}^{(t)}\mathbf{A}\mathbf{M}\mathbf{A}^\top\mathbf{V}^{(t)}\mathbf{S}\|_2^2 + z_i)^{0.5} \\ &\not\approx_{\epsilon/(16\log n)} (\|\vec{e}_i^\top \mathbf{V}^{(t-2^j)}\mathbf{A}\mathbf{M}'\mathbf{A}^\top\mathbf{V}^{(t-2^j)}\mathbf{S}\|_2^2 + z_i)^{0.5} \\ \Rightarrow\ &|\ (\|\vec{e}_i^\top \mathbf{V}^{(t-2^j)}\mathbf{A}\mathbf{M}'\mathbf{A}^\top\mathbf{V}^{(t-2^j)}\mathbf{S}\|_2^2 + z_i)^{0.5} \\ &\quad - (\|\vec{e}_i^\top \mathbf{V}^{(t)}\mathbf{A}\mathbf{M}\mathbf{A}^\top\mathbf{V}^{(t)}\mathbf{S}\|_2^2 + z_i)^{0.5}\ | \geq \epsilon\sqrt{z_i}/(48\log n)\end{aligned}$$

so the index i is returned by FindIndices. □

Next, we prove the claim that calling UpdateIndices(I) indeed guarantees that $\overline{\sigma}_i \approx_{\epsilon/2} \sigma(\mathbf{V}^{(t)}\mathbf{A})_i + z_i$ for all $i \in I$.

Lemma C.7. *For $I \subset [m]$ after a call to UpdateIndices(I) we have $\overline{\sigma}_i \approx_{\epsilon/2} \sigma(\mathbf{V}^{(t)}\mathbf{A})_i + z_i$ for all $i \in I$ w.h.p in n.*

Proof. For $i \in I$ the method UpdateIndices computes

$$\begin{aligned}&\|\vec{e}_i^\top \mathbf{V}^{(t)}\mathbf{A}\mathbf{H}\|_2^2 \\ =\ &\|\vec{e}_i^\top \mathbf{V}^{(t)}\mathbf{A}\mathbf{M}\mathbf{A}^\top\mathbf{V}^{(t)}\mathbf{S}\mathbf{R}\|_2^2 \\ \approx_{\epsilon/8}\ &\|\vec{e}_i^\top \mathbf{V}^{(t)}\mathbf{A}\mathbf{M}\mathbf{A}^\top\mathbf{V}^{(t)}\|_2^2 \\ =\ &\sigma(\mathbf{V}^{(t)}\mathbf{A})_i\end{aligned}$$

where we used that $\mathbf{R}$ is a JL-matrix that yields an $\exp(\pm\epsilon/16)$-factor approximation of the norm and that $\overline{\sigma}_i \approx_\epsilon \sigma(\mathbf{V}^{(t-1)}\mathbf{A}) \approx_4 \sigma(\mathbf{V}^{(t)}\mathbf{A})$, so $\mathbf{S}^2$ is a leverage score sample and satisfies with high probability $\mathbf{A}^\top(\mathbf{V}^{(t)})^2\mathbf{S}^2\mathbf{A} \approx_{\epsilon/32} \mathbf{A}^\top(\mathbf{V}^{(t)})^2\mathbf{A}$ Thus $\mathbf{M} \approx_{\epsilon/16} (\mathbf{A}^\top(\mathbf{V}^{(t)})^2\mathbf{A})^{-1}$.

So if $\overline{\sigma}_i \approx_{3\epsilon/8} \|\vec{e}_i^\top \mathbf{V}^{(t)}\mathbf{A}\mathbf{H}\|_2^2 + z_i$, then $\overline{\sigma}_i \approx_{\epsilon/2} \sigma(\mathbf{V}^{(t)}\mathbf{A})_i + z_i$. On the other hand, if the difference is larger, then UpdateIndices sets $\overline{\sigma}_i \leftarrow \|\vec{e}_i^\top \mathbf{V}^{(t)}\mathbf{A}\mathbf{H}\|_2^2 + z_i$, so then $\overline{\sigma}_i \approx_{\epsilon/8} \sigma(\mathbf{V}^{(t)}\mathbf{A})_i + z_i$. □

At last, we combine the guarantees of FindIndices and UpdateIndices to show that we always maintain the desired approximation. For that we will use the following natural lemma about decomposing an interval into powers of two from [BLN+20].

Lemma C.8 ([BLN+20])**.** *Given any $\overline{t} < t$, there exists a sequence*

$$t = t_0 > t_1 > \ldots > t_k = \overline{t}$$

such that $k \leq 2\log t$ and $t_{z+1} = t_z - 2^{\ell_z}$ where ℓ_z satisfies $2^{\ell_z} | t_z$ for all $z = 0, \ldots, k-1$.

Lemma C.9. *After the t-th call to* Query *we have $\overline{\sigma}_i \approx_\epsilon \sigma(\mathbf{VA})_i + z_i$ w.h.p. in n.*

Proof. Fix some i. We want to prove that $\overline{\sigma}_i \approx_\epsilon \sigma(\mathbf{V}^{(t)}\mathbf{A})_i + z_i$.

Assume index i was last contained in set I when calling UpdateIndices(I) during the $\overline{t}$-th call to Query ($\overline{t} = 0$ if i was never in set I). By Lemma C.7 we have $\overline{\sigma}_i \approx_{\epsilon/2} \sigma(\mathbf{V}^{\overline{t}}\mathbf{A})_i + z_i$. If $\overline{t} = t$, then we are done, so let us focus on the case $\overline{t} < t$ instead. In that case we are left with showing $\sigma(\mathbf{V}^{\overline{t}}\mathbf{A})_i + z_i \approx_{\epsilon/2} \sigma(\mathbf{V}^{(t)}\mathbf{A})_i + z_i$.

By Lemma C.8 there exists a sequence $t = t_0 > t_1 > ... > t_k = \overline{t}$ with $k \leq 2\log t$ and $t_{z+1} = t_z - 2^{\ell_z}$ where ℓ_z satisfies $2^{\ell_z} | t_z$ for all $z = 0, ..., k-1$.

If during the t_z-th call to Query we had $\sigma(\mathbf{V}^{(t_z)}\mathbf{A})_i + z_i \not\approx_{\epsilon/(4\log n)} \sigma(\mathbf{V}^{(t_{z+1})})_i + z_i$, then by Lemma C.6 the index i would have been added to set I in FindIndices. Since by assumption i was not added to I since $\overline{t}$, we thus know $\sigma(\mathbf{V}^{(t_z)}\mathbf{A})_i + z_i \approx_{\epsilon/(4\log n)} \sigma(\mathbf{V}^{(t_{z+1})})_i + z_i$. As the sequence of t_z has length $2\log t$ and $t \leq \sqrt{n}$ (by restarting the data structure after $\sqrt{n}$ calls to Query) we have $\sigma(\mathbf{V}^{(\overline{t})}\mathbf{A})_i + z_i = \sigma(\mathbf{V}^{(t_k)}\mathbf{A})_i + z_i \approx_{\epsilon 2\log t/(4\log n)} \sigma(\mathbf{V}^{(t_0)})_i + z_i = \sigma(\mathbf{V}^{(t)})_i + z_i$, so $\sigma(\mathbf{V}^{(\overline{t})}\mathbf{A})_i \approx_{\epsilon/2} \sigma(\mathbf{V}^{(t)})_i + z_i$. ∎

C.2 Complexity

In this subsection we prove the amortized complexity guarantees of Theorem C.1. Remember that the idea of our algorithm was to detect whenever a leverage score changes a lot, and to then recompute the detected scores. Thus to bound the complexity, we must bound how many i are detected for which σ_i might have changed a lot. For that we will use the following Lemma C.15, which is proven in Appendix C.3.

Lemma C.15. *Let $\mathbf{A} \in \mathbb{R}^{m\times n}$. Let $v^{(0)}, v^{(1)}, \cdots, v^{(T)} \in \mathbb{R}^m_+$ be a sequence of vectors with $v^{(j)} \approx_{1/4} v^{(j-1)}$ for $j = 1, 2, \cdots, T$. Let $\mathbf{F}$ and $\mathbf{S}$ be diagonal matrices on $\mathbb{R}^{m\times m}$ such that $\|\mathbf{F}\|_2 \leq 1$,*

- $\mathbf{F}\mathbf{V}^{(j)} = \mathbf{F}\mathbf{V}^{(j-1)}$ *for $j \in \{1, 2, \cdots, T\}$,*
- $(\mathbf{V}^{(j)} - \mathbf{V}^{(j-1)})\mathbf{S} = \mathbf{V}^{(j)} - \mathbf{V}^{(j-1)}$ *for $j \in \{1, 2, \cdots, T\}$,*
- $\mathbf{A}^\top \mathbf{S}^2 \mathbf{V}^{(j)2}\mathbf{A} \approx_{1/2} \mathbf{A}^\top \mathbf{V}^{(j)2}\mathbf{A}$ *for $j \in \{0, 1, 2, \cdots, T\}$.*

Suppose that there is a sequence $\overline{v}^{(0)}, \overline{v}^{(1)}, \cdots, \overline{v}^{(T)} \in \mathbb{R}^m_+$ such that

$$\overline{v}^{(j)} \approx_{1/6} v^{(j)} \tag{95}$$

$$\|(\widetilde{\mathbf{V}}^{(j-1)})^{-1}(\overline{v}^{(j)} - \overline{v}^{(j-1)})\|_{\sigma(\widetilde{\mathbf{V}}^{(j-1)}\mathbf{A})+\infty} \leq \frac{1}{6}. \tag{96}$$

for $j = 1, 2, \cdots, T$. Then, for $\mathbf{P}^{(j)} \stackrel{\text{def}}{=} \mathbf{V}^{(j)}\mathbf{A}(\mathbf{A}^\top \mathbf{V}^{(j)2}\mathbf{S}^2\mathbf{A})^{-1}\mathbf{A}\mathbf{V}^{(j)}$, we have

$$\|\mathbf{F}(\mathbf{P}^{(T)} - \mathbf{P}^{(0)})\mathbf{S}\|_F^2 \lesssim T^2 + \sum_{j=1}^{T} \sum_{v_i^{(j)} \neq v_i^{(j-1)}} \sigma(\mathbf{V}^{(j)}\mathbf{A})_i.$$

As the complexity of UpdateIndices(I) depends on the size of set I, we first require a bound on the size of these sets. The following lemma bounds the total size of all sets I, when weighting each element $i \in I$ by z_i. This weighting is required because the runtime cost due to one $i \in I$ in UpdateIndices scales in c_i.

Lemma C.10. *Let $I^{(t)}$ be the set I returned by* FindIndices *during the t-th call to* Query*. Then*

$$\sum_{t=1}^{T} \sum_{i\in I^{(t)}} c_i \leq \widetilde{O}\left(T^2 \frac{\|c\|_1}{n\epsilon^2} + \sum_{t=0}^{T-1} \sum_{v_i^{(t)} \neq v_i^{(t+1)}} \left(c_i + \frac{\|c\|_1}{n\epsilon^2} \sigma(\mathbf{V}^{(t)}\mathbf{A})_i \right) \right) \tag{92}$$

Proof. An index i is added to I, if it is contained in set S_j for some $j = 0, ..., \log n$. Thus one call to SCALE$(i, \cdot)$ results in i being in $\log n$ many I. This is the $\sum_{t=0}^{T-1} \sum_{v_i^{(t)} \neq v_i^{(t+1)}} c_i$ term in (92).

Next, an index i is added to I, if it was returned by D_j.HEAVYQUERY in Line 21. The number of returned indices (weighted by c_i) is bounded by

$$\begin{aligned}
&\sum_{k\in[r]} \sum_{i\in[m]} c_i \cdot \mathbf{1}_{|(\mathbf{F}\mathbf{Z}^{-1/2}\mathbf{V}^{(t-2^j)}\mathbf{A}\mathbf{H}\vec{e}_k)_i| > \epsilon/(48r\log n)} \\
&= \widetilde{O}\left(\sum_{k\in[r]} \|\mathbf{F}\mathbf{Z}^{-1/2}\mathbf{V}^{(t-2^j)}\mathbf{A}\mathbf{H}\vec{e}_k\|_c^2 \epsilon^{-2} \right) \\
&= \widetilde{O}\left(\frac{\|c\|_1}{n} \|\mathbf{F}\mathbf{V}^{(t-2^j)}\mathbf{A}\left(\mathbf{M}'\mathbf{A}^\top\mathbf{V}^{(t-2^j)}\mathbf{S} - \mathbf{M}\mathbf{A}^\top\mathbf{V}^{(t)}\mathbf{S}\right)\mathbf{R}\|_F^2 \epsilon^{-2} \right) \\
&\leq \widetilde{O}\left(\frac{\|c\|_1}{n} \left(2^{2j} + \sum_{j=t-2^j}^{t} \sum_{v_i^{(j)} \neq v_i^{(j+1)}} \sigma(\mathbf{V}^{(j)}\mathbf{A})_i \right) \epsilon^{-2} \right), \qquad (93)
\end{aligned}$$

where the first step uses $n \cdot c/\|c\|_1 \leq z$ and the last step uses that $\mathbf{R}$ is a JL-matrix, that $\mathbf{M} \approx_{1/(64n)} (\mathbf{A}^\top\mathbf{V}^{(t)}\mathbf{S}^2\mathbf{A})^{-1}$ and $\mathbf{M}' \approx_{1/(64n)} (\mathbf{A}^\top\mathbf{V}^{(t-2^j)}\mathbf{S}^2\mathbf{A})^{-1}$, and Lemma C.15. Note that we can apply Lemma C.15, because (95) is satisfied by (89), (96) is satisfied by (90), the conditions on $\mathbf{F}$ are satisfied by definition, and the conditions on $\mathbf{S}$ are satisfied by $\mathbf{S}$ being a leverage score sample where we used probability $p_i = 1$ for $i \in S_j \cup C_j$. As D_j.HEAVYQUERY is performed once every j iterations for $j = 0, ..., T$, we obtain (92). □

Using Lemma C.10 we can now bound the total runtime of all calls to UPDATEINDICES.

Lemma C.11. *The amortized cost of the t-th call to UPDATEINDICES is*

$$\widetilde{O}\left(\Psi\epsilon^{-2} + \epsilon^{-4} n(\max_i \operatorname{nnz}(a_i)) + T\frac{\|c\|_1}{n\epsilon^4} + \sum_{v_i^{(t)} \neq v_i^{(t-1)}} \left(\frac{c_i}{\epsilon^2} + \frac{\|c\|_1}{n\epsilon^4} \sigma(\mathbf{V}^{(t-1)}\mathbf{A})_i \right) \right).$$

Proof. The matrix $\mathbf{S}$ has $O(\epsilon^{-2} n \log n)$ non-zero entries and the matrix $\mathbf{R}$ has $O(\epsilon^{-2}\log n)$ columns. Thus we can compute $\mathbf{H}$ in Line 29 in $\widetilde{O}(\Psi\epsilon^{-2} + \epsilon^{-4} n \max_i \operatorname{nnz}(a_i))$ time.

Next, we compute $\|\vec{e}_i^\top \mathbf{V}\mathbf{A}\mathbf{H}\|_2^2$ for all $i \in I$. The time required for that is

$$O\left(\sum_{i\in I} \operatorname{nnz}(a_i)\epsilon^{-2}\log n \right) = \widetilde{O}\left(\epsilon^{-2} \sum_{i\in I} c_i \right)$$

which according to Lemma C.10 is an amortized cost of

$$\widetilde{O}(T\frac{\|c\|_1}{n\epsilon^4} + \sum_{v_i^{(t)} \neq v_i^{(t-1)}} (\frac{c_i}{\epsilon^2} + \frac{\|c\|_1}{n\epsilon^4}\sigma(\mathbf{V}^{(t-1)}\mathbf{A})_i))$$

for the t-th call to UPDATEINDICES. □

Next, we want to analyze the amortized cost of a call to FINDINDICES. Note that the complexity will depend on the size of the sets C_j, because for each $i \in C_j$ the matrix $\mathbf{S}$ in Line 17 will be more dense. An index i is added to C_j in Line 34, if we changed $\overline{\sigma}_i$. Thus before bounding the complexity of FINDINDICES, we will first bound how often $\overline{\sigma}_i$ is changed. We show that we can bound the number of times we change any entry of $\overline{\sigma}$ with respect to how often the function SCALE is called.

Lemma C.12. *Let $\overline{\sigma}^{(t)}$ be the output after the t-th call to* Query *and $v^{(t)}$ be the vector v during the t-th call to* Query*. Then*

$$\sum_{i=1}^{m}\sum_{t=1}^{T}(\sigma(\mathbf{V}^{(t)}\mathbf{A})_i + z_i)\mathbf{1}_{\overline{\sigma}_i^{(t)}\neq\overline{\sigma}_i^{(t-1)}} \leq O\left(\sum_{i=1}^{m}\sum_{t=1}^{T}\sigma(\mathbf{V}^{(t)}\mathbf{A})_i\mathbf{1}_{v_i^{(t)}\neq v_i^{(t-1)}}\epsilon^{-1}\right).$$

Proof. Note that when we update $\overline{\sigma}_i$ in Line 33, then we have $\overline{\sigma}_i \approx_{\epsilon/8} \sigma(\mathbf{VA}) + z_i$, because $\|\vec{e}_i^\top\mathbf{VAH}\|_2^2 \approx_{\epsilon/8} \sigma(\mathbf{VA})$ (see proof of Lemma C.7). Further note that the ith entry of the output vector $\overline{\sigma}$ can only change, whenever $\overline{\sigma}_i \not\approx_{3\epsilon/8} \|\vec{e}_i^\top\mathbf{VAH}\|_2^2 + z_i$ because of Line 32. Thus in order for $\overline{\sigma}_i$ to change, $\sigma(\mathbf{VA})_i + z_i$ must have changed by at least a $\exp(\pm\epsilon/8)$-factor, so we can bound

$$\begin{aligned}
&\sum_{i=1}^{m}\sum_{t=1}^{T}(\sigma(\mathbf{V}^{(t)}\mathbf{A})_i + z_i)\mathbf{1}_{\overline{\sigma}_i^{(t)}\neq\overline{\sigma}_i^{(t-1)}} \\
&= O\left(\sum_{i=1}^{m}\sum_{t=1}^{T}(\sigma(\mathbf{V}^{(t)}\mathbf{A})_i + z_i)\cdot\frac{|(\sigma(\mathbf{V}^{(t)}\mathbf{A})_i + z_i) - (\sigma(\mathbf{V}^{(t-1)}\mathbf{A})_i + z_i)|}{(\sigma(\mathbf{V}^{(t)}\mathbf{A})_i + z_i)\epsilon}\right) \\
&= O\left(\sum_{i=1}^{m}\sum_{t=1}^{T}|\sigma(\mathbf{V}^{(t)}\mathbf{A})_i - \sigma(\mathbf{V}^{(t-1)}\mathbf{A})_i|\epsilon^{-1}\right).
\end{aligned}$$

Here the difference of the two leverage scores can be bounded as follows

$$\begin{aligned}
&|\sigma(\mathbf{V}^{(t)}\mathbf{A})_i - \sigma(\mathbf{V}^{(t-1)}\mathbf{A})_i| \\
&= |(\mathbf{V}_i^{(t)})^2(\mathbf{A}(\mathbf{A}^\top(\mathbf{V}^{(t)})^2\mathbf{A})^{-1}\mathbf{A}^\top)_{i,i} - (\mathbf{V}_i^{(t-1)})^2(\mathbf{A}(\mathbf{A}^\top(\mathbf{V}^{(t-1)})^2\mathbf{A})^{-1}\mathbf{A}^\top)_{i,i}| \\
&\leq |\left((\mathbf{V}_i^{(t)})^2 - (\mathbf{V}_i^{(t-1)})^2\right)(\mathbf{A}(\mathbf{A}^\top(\mathbf{V}^{(t)})^2\mathbf{A})^{-1}\mathbf{A}^\top)_{i,i}| \\
&\quad + |(\mathbf{V}_i^{(t-1)})^2(\mathbf{A}\left((\mathbf{A}^\top(\mathbf{V}^{(t)})^2\mathbf{A})^{-1}\mathbf{A})^{-1} - (\mathbf{A}^\top(\mathbf{V}^{(t-1)})^2\mathbf{A})^{-1}\right)\mathbf{A}^\top)_{i,i}|
\end{aligned}$$

Here the first term can be bounded by

$$\begin{aligned}
&|\left((\mathbf{V}_i^{(t)})^2 - (\mathbf{V}_i^{(t-1)})^2\right)(\mathbf{A}(\mathbf{A}^\top(\mathbf{V}^{(t)})^2\mathbf{A})^{-1}\mathbf{A}^\top)_{i,i}| \\
&= |\left(1 - (\mathbf{V}_i^{(t-1)}/\mathbf{V}_i^{(t)})^2\right)(\mathbf{V}_i^{(t)})^2(\mathbf{A}(\mathbf{A}^\top(\mathbf{V}^{(t)})^2\mathbf{A})^{-1}\mathbf{A}^\top)_{i,i}| \\
&= 3\sigma(\mathbf{V}^{(t)}\mathbf{A})
\end{aligned}$$

by using $v^{(t)} \approx_{1/2} v^{(t-1)}$. The second term can be bounded as follows. For

$$\mathbf{H}_x := \mathbf{A}^\top((1-x)(\mathbf{V}^{(t-1)})^2 + x(\mathbf{V}^{(t)})^2)\mathbf{A}$$

we have

$$(\mathbf{A}^\top(\mathbf{V}^{(t)})^2\mathbf{A})^{-1}\mathbf{A})^{-1} - (\mathbf{A}^\top(\mathbf{V}^{(t-1)})^2\mathbf{A})^{-1} = \int_0^1 \mathbf{H}_x^{-1}\mathbf{A}^\top((\mathbf{V}^{(t)})^2 - (\mathbf{V}^{(t-1)})^2)\mathbf{A}\mathbf{H}_x^{-1}\,dx$$

so

$$\begin{aligned}
&|(\mathbf{V}_i^{(t-1)})^2(\mathbf{A}\left((\mathbf{A}^\top(\mathbf{V}^{(t)})^2\mathbf{A})^{-1}\mathbf{A})^{-1} - (\mathbf{A}^\top(\mathbf{V}^{(t-1)})^2\mathbf{A})^{-1}\right)\mathbf{A}^\top)_{i,i}| \\
&= |\int_0^1 (\mathbf{V}_i^{(t-1)})^2(\mathbf{A}\mathbf{H}_x^{-1}\mathbf{A}^\top((\mathbf{V}^{(t)})^2 - (\mathbf{V}^{(t-1)})^2)\mathbf{A}\mathbf{H}_x^{-1}\mathbf{A}^\top)_{i,i}\,dx| \\
&\leq 2\int_0^1 |(\mathbf{V}^{(t)}\mathbf{A}\mathbf{H}_x^{-1}\mathbf{A}^\top\mathbf{V}^{(t)}(1 - (\mathbf{V}^{(t-1)}\mathbf{V}^{-1})^2)\mathbf{V}^{(t)}\mathbf{A}\mathbf{H}_x^{-1}\mathbf{A}^\top\mathbf{V}^{(t)})_{i,i}|\,dx \\
&= 2\int_0^1 \|(1 - (\mathbf{V}^{(t-1)}\mathbf{V}^{-1})^2)^{1/2}\mathbf{V}^{(t)}\mathbf{A}\mathbf{H}_x^{-1}\mathbf{A}^\top\mathbf{V}^{(t)}\vec{e}_i\|_2^2\,dx
\end{aligned}$$

Note that when taking the sum over all $i \in [m]$ this can be bounded by

$$\begin{aligned}
&\sum_{i\in[m]} \int_0^1 \|(1-(\mathbf{V}^{(t-1)}\mathbf{V}^{-1})^2)^{1/2}\mathbf{V}^{(t)}\mathbf{A}\mathbf{H}_x^{-1}\mathbf{A}^\top\mathbf{V}^{(t)}\vec{e}_i\|_2^2\, dx \\
&= \int_0^1 \|(1-(\mathbf{V}^{(t-1)}\mathbf{V}^{-1})^2)^{1/2}\mathbf{V}^{(t)}\mathbf{A}\mathbf{H}_x^{-1}\mathbf{A}^\top\mathbf{V}^{(t)}\|_F^2\, dx \\
&\leq \|(1-(\mathbf{V}^{(t-1)}\mathbf{V}^{-1})^2)^{1/2}\mathbf{P}\|_F^2\, dx
\end{aligned}$$

where $\mathbf{P} := \mathbf{V}^{(t)}\mathbf{A}(\mathbf{A}^\top(\mathbf{V}^{(t)})^2\mathbf{A})^{-1}\mathbf{A}^\top\mathbf{V}^{(t)}$. This can be written as

$$\begin{aligned}
\|(1-(\mathbf{V}^{(t-1)}/\mathbf{V}^{(t)})^2)^{1/2}\mathbf{P}\|_F^2 &= \sum_{i\in[m]} (1-(v_i^{(t-1)}/v_i^{(t)})^2)^{1/2} \sum_{j\in[m]} \mathbf{P}_{i,j}^2 \\
&= \sum_{i\in[m]} (1-(v_i^{(t-1)}/v_i^{(t)})^2)^{1/2}\sigma(\mathbf{V}^{(t)}\mathbf{A})_i \\
&\leq \sum_{i\in[m]} \mathbf{1}_{v_i^{(t)}\neq v_i^{(t-1)}}\sigma(\mathbf{V}^{(t)}\mathbf{A})_i
\end{aligned}$$

By combining we obtain

$$\sum_{i=1}^m\sum_{t=1}^T(\sigma(\mathbf{V}^{(t)}\mathbf{A})_i + z_i)\mathbf{1}_{\overline{\sigma}_i^{(t)}\neq\overline{\sigma}_i^{(t-1)}} \leq O(\sum_{i=1}^m\sum_{t=1}^T \sigma(\mathbf{V}^{(t)}\mathbf{A})_i\mathbf{1}_{v_i^{(t)}\neq v_i^{(t-1)}}\cdot\epsilon^{-1}).$$

□

Lemma C.13. *The amortized cost of the t-th call to* FindIndices *is*

$$\widetilde{O}\left(\Psi + n\epsilon^{-2}(\max_i \mathrm{nnz}(a_i)) + \frac{\|c\|_1}{n\epsilon^2}\cdot T + Q + \sum_{v_i^{(t)}\neq v_i^{(t-1)}} (\frac{\|c\|_1}{n\epsilon^2}\sigma(\mathbf{V}^{(t-1)}\mathbf{A})_i + c_i)\right).$$

Proof. The cost of FindIndices is dominated by D_j.HeavyQuery(w) for $w = \mathbf{H}\vec{e}_k$, $k = 1, ..., r$, and the computation of $\mathbf{H}$.

Cost from QueryHeavy The cost of D_j.QueryHeavy is bounded by

$$\begin{aligned}
&\widetilde{O}(\sum_{k\in[r]}\left(\|\mathbf{F}\mathbf{Z}^{-1/2}\mathbf{V}^{(t-2^j)}\mathbf{A}\mathbf{H}\vec{e}_k\|_c^2\epsilon^{-2} + Q\right)) \\
&= \widetilde{O}(\frac{\|c\|_1}{n\epsilon^2}\cdot\|\mathbf{F}\mathbf{V}^{(t-2^j)}\mathbf{A}\mathbf{H}\|_F^2 + Q) \\
&= \widetilde{O}(\frac{\|c\|_1}{n\epsilon^2}\cdot\left(2^{2j} + \sum_{j=t-2^j}^{t}\sum_{v_i^{(j)}\neq v_i^{(j+1)}}\sigma(\mathbf{V}^{(j)}\mathbf{A})_i\right) + Q)
\end{aligned}$$

where the last step bounds the frobenius-norm via (93). Note that this cost is paid every $2^j \leq T$ iterations, so we can interpret this as amortized cost for the t-th call

$$\widetilde{O}\left(\frac{\|c\|_1}{n\epsilon^2}\cdot\left(T + \sum_{v_i^{(t)}\neq v_i^{(t-1)}}\sigma(\mathbf{V}^{(t-1)}\mathbf{A})_i\right) + Q\right).$$

Cost from matrix H Fix some j for which we have to compute $\mathbf{H}$ in Line 20. Here the matrix $\mathbf{R}$ has $O(\log n)$ columns, so we must compute $O(\log n)$ products with $\overline{\mathbf{A}} := \mathbf{SVA}$ and $\overline{\mathbf{A}}' := \mathbf{S}(\mathbf{V} - \Delta^{(j)})\mathbf{A}$, and further solve $O(\log n)$ linear systems in $\overline{\mathbf{A}}^\top \overline{\mathbf{A}}$ and $\overline{\mathbf{A}}'^\top \overline{\mathbf{A}}$.

Note that both $\overline{\mathbf{A}}$ and $\overline{\mathbf{A}}'$ have the same number of non-zero entries, so the time for computing $\mathbf{H}$ is bounded by $\widetilde{O}(\Psi + \operatorname{nnz}(\overline{\mathbf{A}}))$. By definition of $\mathbf{S}$, the number of non-zero entries in both matrices is bounded by

$$\begin{aligned}\operatorname{nnz}(\overline{\mathbf{A}}) &= \widetilde{O}\left(n\epsilon^{-2}(\max_i \operatorname{nnz}(a_i)) + \sum_{i\in S_j} \operatorname{nnz}(a_i) + \sum_{i\in C_j} \operatorname{nnz}(a_i)\right) \\ &= \widetilde{O}\left(n\epsilon^{-2}(\max_i \operatorname{nnz}(a_i)) + \sum_{k=t-2^j}^{t} \sum_{v_i^{(t)}\neq v_i^{(t-1)}} \operatorname{nnz}(a_i) + \sum_{k=t-2^j}^{t} \sum_{\overline{\sigma}_i^{(t-1)}\neq \overline{\sigma}_i^{(t-2)}} \operatorname{nnz}(a_i)\right).\end{aligned}$$

As the cost for computing $\mathbf{H}$ is paid once every 2^j calls to FindIndices, the total cost over all T calls to FindIndices is bounded by

$$\begin{aligned}&\widetilde{O}\left(T\cdot\Psi + Tn\epsilon^{-2}(\max_i \operatorname{nnz}(a_i)) + \left(\sum_{t=1}^{T}\sum_{v_i^{(t)}\neq v_i^{(t-1)}} \operatorname{nnz}(a_i)\right) + \left(\sum_{t=1}^{T}\sum_{\overline{\sigma}_i^{(t-1)}\neq \overline{\sigma}_i^{(t-2)}} \operatorname{nnz}(a_i)\right)\right) \\ &= \widetilde{O}\left(T\cdot\Psi + Tn\epsilon^{-2}(\max_i \operatorname{nnz}(a_i)) + \left(\sum_{t=1}^{T}\sum_{v_i^{(t)}\neq v_i^{(t-1)}} c_i\right) + \left(\frac{\|c\|_1}{n}\sum_{t=1}^{T}\sum_{\overline{\sigma}_i^{(t-1)}\neq \overline{\sigma}_i^{(t-2)}} (\sigma(\mathbf{V}^{(t-2)}\mathbf{A})_i + z_i)\right)\right) \\ &= \widetilde{O}\left(T\cdot\Psi + Tn\epsilon^{-2}(\max_i \operatorname{nnz}(a_i)) + \sum_{t=1}^{T}\sum_{v_i^{(t)}\neq v_i^{(t-1)}} \left(\frac{\|c\|_1}{n\epsilon}\sigma(\mathbf{V}^{(t-1)}\mathbf{A})_i + c_i\right)\right)\end{aligned}$$

by $\|c\|_1 z_i/n \geq c_i \geq \operatorname{nnz}(a_i)$ and Lemma C.12.

Amortized Cost The amortized cost for the t-th call to FindIndices is thus

$$\widetilde{O}\left(\Psi + n\epsilon^{-2}(\max_i \operatorname{nnz}(a_i)) + \frac{\|c\|_1}{n\epsilon^2}\cdot T + Q + \sum_{v_i^{(t)}\neq v_i^{(t-1)}} \frac{\|c\|_1}{n\epsilon^2}(\sigma(\mathbf{V}^{(t-1)}\mathbf{A})_i + c_i)\right).$$

□

We now charge the terms in the complexities of FindIndices and UpdateIndices as amortized cost to Scale and Query to obtain the complexities as stated in Theorem C.2.

Proof of Theorem C.2. We now charge each term of the amortized cost of FindIndices and UpdateIndices to Scale and Query.

We charge the amortized cost of UpdateIndices (Lemma C.11) as follows

$$\widetilde{O}\left(\underbrace{\Psi\epsilon^{-2} + \epsilon^{-4}n(\max_i \operatorname{nnz}(a_i)) + T\frac{\|c\|_1}{n\epsilon^4}}_{\textsc{Query}} + \underbrace{\sum_{v_i^{(t)}\neq v_i^{(t-1)}} \frac{c_i}{\epsilon^2} + \frac{\|c\|_1}{n\epsilon^4}\sigma(\mathbf{V}^{(t-1)}\mathbf{A})_i}_{\textsc{Scale}}\right).$$

And for FindIndices (Lemma C.13) we charge

$$\widetilde{O}\left(\underbrace{\Psi + n\epsilon^{-2}(\max_i \operatorname{nnz}(a_i)) + \frac{\|c\|_1}{n\epsilon^2}\cdot T + Q}_{\textsc{Query}} + \underbrace{\sum_{v_i^{(t)}\neq v_i^{(t-1)}} \frac{\|c\|_1}{n\epsilon^2}\sigma(\mathbf{V}^{(t-1)}\mathbf{A})_i + c_i}_{\textsc{Scale}}\right).$$

Note that we reinitialize the data structure after T iterations. Thus the amortized complexity of QUERY also depends on the cost of initializing the data structure, which we analyze next.

Initialization During initialization we have to compute $\exp(\pm\epsilon)$-approximate leverage scores. It is known that this can be done in time $\widetilde{O}(\mathrm{nnz}(\mathbf{A}) + S)$, where S is the time required to solve a linear system in a matrix of the form $\mathbf{A}^\top\mathbf{W}\mathbf{A}$ by using the Johnson-Lindenstrauss lemma [JL84, SS08]. By Condition 1 of Theorem C.2 computing the leverage score thus takes $\widetilde{O}(P + \Psi + \mathrm{nnz}(\mathbf{A}))$ time. We further initialize $\widetilde{O}(1)$ instances of the assumed HEAVYHITTER data structure, which takes $\widetilde{O}(P)$ time.

Note that re-initializing the data structure every T iterations adds $\widetilde{O}((\Psi + \mathrm{nnz}(\mathbf{A}) + P)/T)$ amortized cost to QUERY.

Query By collecting all the terms that we charge to QUERY, we obtain the following amortized cost per call to QUERY:

$$\begin{aligned}&\widetilde{O}\left(\Psi\epsilon^{-2} + \epsilon^{-4}n(\max_i \mathrm{nnz}(a_i)) + T\frac{\|c\|_1}{n\epsilon^4} + Q + P/T\right)\\ =\ &\widetilde{O}\left(\Psi\epsilon^{-2} + \epsilon^{-4}n(\max_i \mathrm{nnz}(a_i)) + \epsilon^{-2}\sqrt{\frac{P\|c\|_1}{n}} + Q\right)\end{aligned}$$

by choice of $T = \epsilon^2\sqrt{Pn/\|c\|_1}$.

Scale When calling SCALE$(i, \cdot)$, the data structure calls D_j.SCALE$(i, \cdot)$ for $j = 1, ..., \log n$ which takes $\widetilde{O}(c_i)$ time. Together with the amortized complexity terms that we charged to SCALE we obtain an amortized complexity per call to SCALE of

$$\widetilde{O}\left(\frac{\|c\|_1}{n\epsilon^4}\sigma(\mathbf{V}^{(t)}\mathbf{A})_i + \frac{c_i}{\epsilon^2}\right).$$

Here we used that the sum $\sum_{v_i^{(t)} \neq v_i^{(t-1)}} \sigma(\mathbf{V}^{(t-1)}\mathbf{A})_i + c_i$ has one term for each call to SCALE. □

C.3 Stabilizer

In this section we want to prove Lemma C.15. To prove the main lemma, we will first prove a weaker variant that involves only one step on $\mathbf{P}$.

Lemma C.14. *Let $\mathbf{A} \in \mathbb{R}^{m\times n}$. Given $v, v' \in \mathbb{R}^m_+$ with $v \approx_{1/2} v'$. Let $\mathbf{F}$ and $\mathbf{S}$ be diagonal matrices on $\mathbb{R}^{m\times m}$ such that $\mathbf{F}\mathbf{V} = \mathbf{F}\mathbf{V}'$, $(\mathbf{V} - \mathbf{V}')\mathbf{S} = \mathbf{V} - \mathbf{V}'$, $\|\mathbf{F}\|_2 \leq 1$ and $\mathbf{A}^\top\mathbf{S}^2\mathbf{V}^2\mathbf{A} \approx_{1/2} \mathbf{A}^\top\mathbf{V}^2\mathbf{A}$ and $\mathbf{A}^\top\mathbf{S}^2\mathbf{V}'^2\mathbf{A} \approx_{1/2} \mathbf{A}^\top\mathbf{V}'^2\mathbf{A}$. Let $\mathbf{P} = \mathbf{V}\mathbf{A}(\mathbf{A}^\top\mathbf{V}^2\mathbf{S}^2\mathbf{A})^{-1}\mathbf{A}\mathbf{V}$ and $\mathbf{P}' = \mathbf{V}'\mathbf{A}(\mathbf{A}^\top\mathbf{V}'^2\mathbf{S}^2\mathbf{A})^{-1}\mathbf{A}\mathbf{V}'$, we have*

$$\|\mathbf{F}(\mathbf{P}' - \mathbf{P})\mathbf{S}\|_F \lesssim \|\mathbf{V}^{-1}(v' - v)\|_{\sigma(\mathbf{V}\mathbf{A})}.$$

Proof. Let $\Delta = \mathbf{diag}(\frac{v'-v}{v})$. Then, we have

$$\begin{aligned}\mathbf{P}' =&\Delta\mathbf{V}\mathbf{A}(\mathbf{A}^\top\mathbf{V}'^2\mathbf{S}^2\mathbf{A})^{-1}\mathbf{A}\mathbf{V}' + \mathbf{V}\mathbf{A}(\mathbf{A}^\top\mathbf{V}'^2\mathbf{S}^2\mathbf{A})^{-1}\mathbf{A}\mathbf{V}\Delta + \mathbf{V}\mathbf{A}(\mathbf{A}^\top\mathbf{V}'^2\mathbf{S}^2\mathbf{A})^{-1}\mathbf{A}\mathbf{V} \qquad (94)\end{aligned}$$

By Woodbury matrix identity, we have

$$\begin{aligned}(\mathbf{A}^\top\mathbf{V}'^2\mathbf{S}^2\mathbf{A})^{-1} &= (\mathbf{A}^\top\mathbf{V}\mathbf{S}(\mathbf{I} + \Delta)^2\mathbf{S}\mathbf{V}\mathbf{A})^{-1}\\ &= (\mathbf{A}^\top\mathbf{V}^2\mathbf{S}^2\mathbf{A})^{-1} - (\mathbf{A}^\top\mathbf{V}^2\mathbf{S}^2\mathbf{A})^{-1}\mathbf{A}^\top\mathbf{V}\mathbf{S}((2\Delta + \Delta^2)^{-1} + \mathbf{P})^{-1}\mathbf{S}\mathbf{V}\mathbf{A}(\mathbf{A}^\top\mathbf{V}^2\mathbf{S}^2\mathbf{A})^{-1}.\end{aligned}$$

Applying this into (94), we have

$$\mathbf{P}' - \mathbf{P} =\Delta\mathbf{V}\mathbf{A}(\mathbf{A}^\top\mathbf{V}'^2\mathbf{S}^2\mathbf{A})^{-1}\mathbf{A}\mathbf{V}' + \mathbf{V}\mathbf{A}(\mathbf{A}^\top\mathbf{V}'^2\mathbf{S}^2\mathbf{A})^{-1}\mathbf{A}\mathbf{V}\Delta + \mathbf{P}\mathbf{S}((2\Delta + \Delta^2)^{-1} + \mathbf{P})^{-1}\mathbf{S}\mathbf{P}$$

Since $\mathbf{FV} = \mathbf{FV}'$ and $(\mathbf{V} - \mathbf{V}')\mathbf{S} = \mathbf{V} - \mathbf{V}'$, we have $\mathbf{F}\Delta = 0$ and $\Delta\mathbf{S} = \Delta$. Together with $\|\mathbf{F}\|_2 \le 1$, we have

$$\begin{aligned}\|\mathbf{F}(\mathbf{P}' - \mathbf{P})\mathbf{S}\|_F \le& \|\mathbf{FVA}(\mathbf{A}^\top \mathbf{V}'^2 \mathbf{S}^2 \mathbf{A})^{-1}\mathbf{AV}\Delta\|_F + \|\mathbf{FPS}((2\Delta + \Delta^2)^{-1} + \mathbf{P})^{-1}\mathbf{SPS}\|_F \\ \le& \|\mathbf{VA}(\mathbf{A}^\top \mathbf{V}'^2 \mathbf{S}^2 \mathbf{A})^{-1}\mathbf{AV}\Delta\|_F + \|\mathbf{PS}((2\Delta + \Delta^2)^{-1} + \mathbf{P})^{-1}\mathbf{SPS}\|_F.\end{aligned}$$

For the first term, we have

$$\begin{aligned}\|\mathbf{VA}(\mathbf{A}^\top \mathbf{V}'^2 \mathbf{S}^2 \mathbf{A})^{-1}\mathbf{AV}\Delta\|_F^2 &= \operatorname{Tr} \Delta\mathbf{VA}(\mathbf{A}^\top \mathbf{V}'^2 \mathbf{S}^2 \mathbf{A})^{-1}\mathbf{AV}^2\mathbf{A}(\mathbf{A}^\top \mathbf{V}'^2 \mathbf{S}^2 \mathbf{A})^{-1}\mathbf{AV}\Delta \\ &\lesssim \operatorname{Tr} \Delta\mathbf{VA}(\mathbf{A}^\top \mathbf{V}'^2 \mathbf{S}^2 \mathbf{A})^{-1}\mathbf{AV}\Delta \\ &\lesssim \operatorname{Tr} \Delta\mathbf{VA}(\mathbf{A}^\top \mathbf{V}^2 \mathbf{S}^2 \mathbf{A})^{-1}\mathbf{AV}\Delta \\ &= \|\mathbf{V}^{-1}(v' - v)\|_{\sigma(\mathbf{VA})}^2\end{aligned}$$

where we used $\mathbf{A}^\top \mathbf{V}'^2 \mathbf{S}^2 \mathbf{A} \approx \mathbf{A}^\top \mathbf{V}'^2 \mathbf{A} \approx \mathbf{A}^\top \mathbf{V}^2 \mathbf{A}$ at the first and second inequality.

For the second term, we have

$$\mathbf{PS}^2\mathbf{P} = \mathbf{VA}(\mathbf{A}^\top \mathbf{V}^2 \mathbf{S}^2 \mathbf{A})^{-1}\mathbf{A}^\top \mathbf{S}^2 \mathbf{V}^2 \mathbf{A}(\mathbf{A}^\top \mathbf{V}^2 \mathbf{S}^2 \mathbf{A})^{-1}\mathbf{A}^\top \mathbf{V} = \mathbf{P}.$$

Hence, we have

$$\begin{aligned}\|\mathbf{PS}((2\Delta + \Delta^2)^{-1} + \mathbf{P})^{-1}\mathbf{SPS}\|_F^2 &\lesssim \|\mathbf{PS}((2\Delta + \Delta^2)^{-1} + \mathbf{P})^{-1}\mathbf{SP}\|_F^2 \\ &\le \operatorname{Tr} \mathbf{PS}((2\Delta + \Delta^2)^{-1} + \mathbf{P})^{-2}\mathbf{SP} \\ &\le \operatorname{Tr} \mathbf{PS}(2\Delta + \Delta^2)^2\mathbf{SP} \\ &\le 9 \operatorname{Tr} \mathbf{PS}\Delta^2\mathbf{SP}\end{aligned}$$

where we used $\Delta \preceq I$ at the last inequality. Finally, using $\mathbf{S}\Delta^2\mathbf{S} = \Delta^2$, we have that the last term bounded by $O(\|\mathbf{V}^{-1}(v' - v)\|_{\sigma(\mathbf{VA})}^2)$. Combining both terms give the result. □

Now, we can prove the main statement.

Lemma C.15. *Let $\mathbf{A} \in \mathbb{R}^{m\times n}$. Let $v^{(0)}, v^{(1)}, \cdots, v^{(T)} \in \mathbb{R}_+^m$ be a sequence of vectors with $v^{(j)} \approx_{1/4} v^{(j-1)}$ for $j = 1, 2, \cdots, T$. Let $\mathbf{F}$ and $\mathbf{S}$ be diagonal matrices on $\mathbb{R}^{m\times m}$ such that $\|\mathbf{F}\|_2 \le 1$,*

- *$\mathbf{FV}^{(j)} = \mathbf{FV}^{(j-1)}$ for $j \in \{1, 2, \cdots, T\}$,*
- *$(\mathbf{V}^{(j)} - \mathbf{V}^{(j-1)})\mathbf{S} = \mathbf{V}^{(j)} - \mathbf{V}^{(j-1)}$ for $j \in \{1, 2, \cdots, T\}$,*
- *$\mathbf{A}^\top \mathbf{S}^2 \mathbf{V}^{(j)2}\mathbf{A} \approx_{1/2} \mathbf{A}^\top \mathbf{V}^{(j)2}\mathbf{A}$ for $j \in \{0, 1, 2, \cdots, T\}$.*

Suppose that there is a sequence $\overline{v}^{(0)}, \overline{v}^{(1)}, \cdots, \overline{v}^{(T)} \in \mathbb{R}_+^m$ such that

$$\overline{v}^{(j)} \approx_{1/6} v^{(j)} \tag{95}$$

$$\|(\widetilde{\mathbf{V}}^{(j-1)})^{-1}(\overline{v}^{(j)} - \overline{v}^{(j-1)})\|_{\sigma(\widetilde{\mathbf{V}}^{(j-1)}\mathbf{A})+\infty} \le \frac{1}{6}. \tag{96}$$

for $j = 1, 2, \cdots, T$. Then, for $\mathbf{P}^{(j)} \stackrel{\text{def}}{=} \mathbf{V}^{(j)}\mathbf{A}(\mathbf{A}^\top \mathbf{V}^{(j)2}\mathbf{S}^2\mathbf{A})^{-1}\mathbf{AV}^{(j)}$, we have

$$\|\mathbf{F}(\mathbf{P}^{(T)} - \mathbf{P}^{(0)})\mathbf{S}\|_F^2 \lesssim T^2 + \sum_{j=1}^{T} \sum_{v_i^{(j)} \neq v_i^{(j-1)}} \sigma(\mathbf{V}^{(j)}\mathbf{A})_i.$$

Proof. Let I be the set of indices i such that $v_i^{(j)}$ changed during the $j \in \{0, 1, \cdots, T\}$. The proof involves defining a new weight sequence

$$w_i^{(j)} = \begin{cases} \overline{v}_i^{(t)} v_i^{(0)} / \overline{v}_i^{(0)} & \text{if } i \in I \\ v_i^{(0)} & \text{elses} \end{cases}$$

and $w_i^{(T+1)} = v_i^{(t)}$. We define

$$\widetilde{\mathbf{P}}^{(j)} = \mathbf{W}^{(j)}\mathbf{A}(\mathbf{A}^\top \mathbf{W}^{(j)2}\mathbf{S}^2\mathbf{A})^{-1}\mathbf{A}\mathbf{W}^{(j)}.$$

Since $w_i^{(0)} = v_i^{(0)}$ and $w_i^{(T+1)} = v_i^{(T)}$, we have

$$\|\mathbf{F}(\mathbf{P}^{(T)} - \mathbf{P}^{(0)})\mathbf{S}\|_F \le \|\mathbf{F}(\widetilde{\mathbf{P}}^{(T+1)} - \widetilde{\mathbf{P}}^{(0)})\mathbf{S}\|_F \le \sum_{j=1}^{T+1} \|\mathbf{F}(\widetilde{\mathbf{P}}^{(j)} - \widetilde{\mathbf{P}}^{(j-1)})\mathbf{S}\|_F.$$

Now we check the conditions of Lemma C.14. Since $\overline{v}^{(j)} \approx_{1/6} v^{(j)}$ and $\|(\widetilde{\mathbf{V}}^{(j-1)})^{-1}(\overline{v}^{(j)} - \overline{v}^{(j-1)})\|_\infty \le \frac{1}{6}$, we have $w^{(j)} \approx_{1/2} w^{(j-1)}$ for all j. For any $i \in I$, we have $\mathbf{F}_{ii} = 0$ and $\mathbf{S}_{ii} = 1$ (due to the condition $\mathbf{F}\mathbf{V}^{(j)} = \mathbf{F}\mathbf{V}^{(j-1)}$ and $(\mathbf{V}^{(j)} - \mathbf{V}^{(j-1)})\mathbf{S} = \mathbf{V}^{(j)} - \mathbf{V}^{(j-1)}$). For $i \notin I$, $w_i^{(j)}$ is a constant. Hence, this verifies the conditions of Lemma C.14: $\mathbf{F}\mathbf{W}^{(j)} = \mathbf{F}\mathbf{W}^{(j-1)}$, $(\mathbf{W}^{(j)} - \mathbf{W}^{(j-1)})\mathbf{S} = \mathbf{W}^{(j)} - \mathbf{W}^{(j-1)}$ and $\mathbf{A}^\top(\mathbf{S}^{(j-1)})^2(\mathbf{V}^{(j-1)})^2\mathbf{A} \approx_{1/2} \mathbf{A}^\top(\mathbf{V}^{(j-1)})^2\mathbf{A}$.

For $j \le T$, Lemma C.14 shows that

$$\begin{aligned}\|\mathbf{F}(\widetilde{\mathbf{P}}^{(j)} - \widetilde{\mathbf{P}}^{(j-1)})\mathbf{S}\|_F &\lesssim \|(\mathbf{W}^{(j-1)})^{-1}(w^{(j)} - w^{(j-1)})\|_{\sigma(\mathbf{W}^{(j-1)}\mathbf{A})} \\ &= \|(\widetilde{\mathbf{V}}^{(j-1)})^{-1}(\overline{v}^{(j)} - \overline{v}^{(j-1)})\|_{\sigma(\mathbf{W}^{(j-1)}\mathbf{A})} \\ &\lesssim \|(\widetilde{\mathbf{V}}^{(j-1)})^{-1}(\overline{v}^{(j)} - \overline{v}^{(j-1)})\|_{\sigma(\widetilde{\mathbf{V}}^{(j-1)}\mathbf{A})} \lesssim 1\end{aligned}$$

where we used the formula of $w^{(j)}$ and $w^{(j-1)}$ in the first equality, $w^{(j-1)} \approx_{1/6} \overline{v}^{(j-1)}$ and the assumption on $\overline{v}$ at the last line.

For $j = T+1$, Lemma C.14 shows that

$$\begin{aligned}\|\mathbf{F}(\widetilde{\mathbf{P}}^{(T+1)} - \widetilde{\mathbf{P}}^{(T)})\mathbf{S}\|_F^2 &\lesssim \|(\mathbf{W}^{(T)})^{-1}(w^{(T+1)} - w^{(T)})\|_{\sigma(\mathbf{W}^{(T)}\mathbf{A})}^2 \\ &\lesssim \sum_{w_i^{(j)} \ne w_i^{(j-1)}} \sigma_i(\mathbf{W}^{(T)}\mathbf{A}) \lesssim \sum_{i\in I} \sigma_i(\widetilde{\mathbf{V}}^{(T)}\mathbf{A})\end{aligned}$$

where we used $\|(\mathbf{W}^{(j-1)})^{-1}(w^{(j)} - w^{(j-1)})\|_\infty \lesssim 1$ in the second inequality, we used $\overline{v}^{(T)} \approx_{1/6} w^{(T)}$ and I contains all changing indices at the end.

Combining both cases, we have

$$\|\mathbf{F}(\mathbf{P}^{(T)} - \mathbf{P}^{(0)})\mathbf{S}\|_F^2 \lesssim T^2 + \sum_{i\in I} \sigma_i(\widetilde{\mathbf{V}}^{(T)}\mathbf{A}).$$

Hence, it suffices to prove that

$$\sum_{i\in I} \sigma_i(\widetilde{\mathbf{V}}^{(T)}\mathbf{A}) \lesssim \sum_{i\in I} \sigma_i(\widetilde{\mathbf{V}}^{(t_i)}\mathbf{A}) + T^2.$$

where t_i be the time such that $v_i^{(t_i)} \ne v_i^{(t_i-1)}$. To prove this, we define $\sigma_i^{(t_i)} = \sigma_i(\widetilde{\mathbf{V}}^{(t_i)}\mathbf{A})$ and for all $j \ge t_i$, we define

$$\sigma_i^{(j+1)} = \begin{cases} \frac{\sigma_i(\widetilde{\mathbf{V}}^{(j+1)}\mathbf{A})}{\sigma_i(\widetilde{\mathbf{V}}^{(j)}\mathbf{A})} \cdot \sigma_i^{(j)} & \left|\frac{\sigma_i(\widetilde{\mathbf{V}}^{(j+1)}\mathbf{A}) - \sigma_i(\widetilde{\mathbf{V}}^{(j)}\mathbf{A})}{\sigma_i(\widetilde{\mathbf{V}}^{(j)}\mathbf{A})}\right| \ge \frac{1}{T} \\ \sigma_i^{(j)} & \text{elses} \end{cases}.$$

Intuitively, $\sigma_i^{(j)}$ is essentially the same as $\sigma_i(\widetilde{\mathbf{V}}^{(j)}\mathbf{A})$ except we ignore all smaller than $\frac{1}{T}$ relative movement. Since there are only T steps, we have $\sigma_i^{(j)} \approx_2 \sigma_i(\widetilde{\mathbf{V}}^{(j)}\mathbf{A})$. By the assumption $\|(\widetilde{\mathbf{V}}^{(j-1)})^{-1}(\overline{v}^{(j)} - \overline{v}^{(j-1)})\|_{\sigma(\widetilde{\mathbf{V}}^{(j-1)}\mathbf{A})+\infty} \le \frac{1}{6}$ and [LS15, Lemma 14], we have that

$$\|\sigma(\widetilde{\mathbf{V}}^{(j-1)}\mathbf{A})^{-1}(\sigma(\widetilde{\mathbf{V}}^{(j)}\mathbf{A}) - \sigma(\widetilde{\mathbf{V}}^{(j-1)}\mathbf{A}))\|_{\sigma(\widetilde{\mathbf{V}}^{(j-1)}\mathbf{A})} \lesssim 1.$$

Using the definition of $\sigma_i^{(j)}$, we have a similar bound for $\sigma_i^{(j)}$:

$$\|(\sigma^{(j-1)})^{-1}(\sigma^{(j)} - \sigma^{(j-1)})\|_{\sigma^{(j-1)}} \lesssim 1.$$

Since $\sigma^{(j)}$ only makes relative movement at least $\frac{1}{T}$, this implies $\sum_i |\sigma_i^{(j)} - \sigma_i^{(j-1)}| \lesssim T$. Hence, we have

$$\sum_{i\in I} \sigma_i(\widetilde{\mathbf{V}}^{(T)}\mathbf{A}) \lesssim \sum_{i\in I} \sigma_i^{(T)} \lesssim \sum_{i\in I} \sigma_i^{(t_i)} + T^2 \lesssim \sum_{i\in I} \sigma_i(\widetilde{\mathbf{V}}^{(t_i)}\mathbf{A}) + T^2.$$

□

D Primal and Gradient Maintenance

Theorem D.1 (Primal/Gradient Maintenance)**.** *There exists a deterministic data-structure that supports the following operations*

- *Initialize* ($\mathbf{A} \in \mathbb{R}^{m\times n}, x^{(\mathrm{init})} \in \mathbb{R}^m, g \in \mathbb{R}^m, \widetilde{\tau} \in \mathbb{R}^m, z \in \mathbb{R}^m, w \in [0,1]^m, \epsilon > 0$)*: The data-structure preprocesses the given matrix* $\mathbf{A} \in \mathbb{R}^{m\times n}$*, vectors* $x^{(\mathrm{init})}, g, \widetilde{\tau}, z \in \mathbb{R}^m$*, and the accuracy parameters* $w \in [0,1]^m$ *and* $\epsilon > 0$ *in* $\widetilde{O}(\mathrm{nnz}(\mathbf{A}))$ *time. We denote* $\mathbf{G}$ *the diagonal matrix* $\mathbf{Diag}(g)$*. The data-structure assumes* $0.5 \le z \le 2$ *and* $n/m \le \widetilde{\tau} \le 2$*.*
- *Update*($i \in [m], a \in \mathbb{R}, b \in \mathbb{R}, c \in \mathbb{R}$)*: Sets* $g_i \leftarrow a$*,* $\widetilde{\tau}_i \leftarrow b$ *and* $z_i \leftarrow c$ *in* $O(\mathrm{nnz}(a_i) + \log n)$ *time. The data-structure assumes* $0.5 \le b \le 2$ *and* $n/m \le c \le 2$*.*
- *SetAccuracy*(i, δ) *Sets* $w_i \leftarrow \delta$ *in* $O(\log n)$ *time.*
- *QueryProduct*()*: Returns* $\mathbf{A}^\top \mathbf{G} \nabla\Psi(\overline{z})^{\flat(\overline{\tau})} \in \mathbb{R}^n$ *for some* $\overline{\tau} \in \mathbb{R}^m$*,* $\overline{z} \in \mathbb{R}^m$ *with* $\overline{\tau} \approx_\epsilon \widetilde{\tau}$ *and* $\|\overline{z} - z\|_\infty \le \epsilon$*, where*

  $$x^{\flat(\overline{\tau})} := \mathrm{argmax}_{\|w\|_{\overline{\tau}+\infty}\le 1} \langle x, w\rangle.$$

  Every call to QueryProduct must be followed by a call to QuerySum, and we bound their complexity together (see QuerySum).
- *QuerySum*($h \in \mathbb{R}^m$)*: Let* $v^{(\ell)}$ *be the vector* $\mathbf{G}\nabla\Psi(\overline{z})^{\flat(\overline{\tau})}$ *used for the result of the* ℓ*-th call to QueryProduct. Let* $h^{(\ell)}$ *be the input vector* h *given to the* ℓ*-th call to QuerySum. We define*

  $$x^{(t)} := x^{(\mathrm{init})} + \sum_{\ell=1}^{t} \left(v^{(\ell)} + h^{(\ell)}\right).$$

  Then the t*-th call to QuerySum returns a vector* $\overline{x} \in \mathbb{R}^m$ *with*

  $$\|w^{-1}(\overline{x} - x^{(t)})\|_\infty \le \epsilon.$$

  Assuming the input vector h *is given in a sparse representation (e.g. a list of non-zero entries), then after* T *calls to QuerySum and QueryProduct the total time for all calls together is bounded by*

  $$O\left(Tn\epsilon^{-2}\log n + \log n \cdot \sum_{\ell=0}^{T} \|h^{(\ell)}\|_0 + T\log n \cdot \sum_{\ell=1}^{T} \|v^{(\ell)}/w^{(\ell-1)}\|_2^2/\epsilon^2\right)$$

  The output $\overline{x} \in \mathbb{R}^m$ *is returned in a compact representation to reduce the size. In particular, the data-structure returns a pointer to* $\overline{x}$ *and a set* $J \subset [m]$ *of indices which specifies which entries of* $\overline{x}$ *have changed between the current and previous call to QuerySum.*
- *ComputeExactSum*()*: Returns the exact* $x^{(t)}$ *in* $O(m\log n)$ *time.*
- *Potential*()*: Returns* $\Psi(\overline{z}) = \sum_i \cosh(\lambda\overline{z}_i)$ *in* $O(1)$ *time for some* $\overline{z}$ *with* $\|\overline{z} - z\|_\infty \le \epsilon$*.*

This is almost exactly Theorem 7.1 in [BLN+20] except here we include an additional per-coordinate accuracy parameter w, and in QuerySum the guarantee becomes $\|w^{-1}(\overline{x}-x^{(t)})\|_\infty \leq \epsilon$ instead of $\|\overline{x}^{-1}(\overline{x}-x^{(t)})\|_\infty \leq \epsilon$. The implementation and analysis of the data strucutre largely follows from [BLN+20], and we include it for completeness. The main idea is to maintain a $O(\epsilon^{-2}\log n)$-dimensional approximation $\nabla\Psi(\overline{z})^{\flat(\overline{\tau})}$ of the m-dimensional exact gradient $\nabla\Psi(z)^{\flat(\widetilde{\tau})} \in \mathbb{R}^m$ by slightly perturbing $\widetilde{\tau}$ and z. The approximation $\nabla\Psi(\overline{z})^{\flat(\overline{\tau})}$ is formally still m-dimensional, but we say $O(\epsilon^{-2}\log n)$ dimension in the sense that its m entries can be put into $O(\epsilon^{-2}\log n)$ buckets, and entries in the same bucket share a common value. The proof of the theorem follows from two sub data structures which we specify below. The first lemma addresses the construction and maintenance of the low dimensional approximation of the exact gradient, and we use the corresponding result from [BLN+20] without modification.

Lemma D.2 ([BLN+20, Lemma 7.2])**.** *There exists a deterministic data-structure that supports the following operations*

- *Initialize* ($\mathbf{A} \in \mathbb{R}^{m\times n}, g \in \mathbb{R}^m, \widetilde{\tau} \in \mathbb{R}^m, z \in \mathbb{R}^m, \epsilon > 0$)*: The data-structure preprocesses the given matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, vectors $g, \widetilde{\tau}, z \in \mathbb{R}^m$, and accuracy parameter $\epsilon > 0$ in $O(\mathrm{nnz}(\mathbf{A}))$ time. The data-structure assumes $0.5 \leq z \leq 2$ and $n/m \leq \widetilde{\tau} \leq 2$. The output is a partition $\bigcup_{k=1}^K I_k = [m]$ with $K = O(\epsilon^{-2}\log n)$.*
- *Update*($i \in [m], a \in \mathbb{R}, b \in \mathbb{R}, c \in \mathbb{R}$)*: Sets $g_i = a$, $\widetilde{\tau}_i = b$ and $z_i = c$ in $O(\mathrm{nnz}(a_i))$ time. The data-structure assumes $0.5 \leq z \leq 2$ and $n/m \leq \widetilde{\tau} \leq 2$. The index i might be moved to a different set, so the data-structure returns k such that $i \in I_k$.*
- *Query*()*: Returns $\mathbf{A}^\top\mathbf{G}\nabla\Psi(\overline{z})^{\flat(\overline{\tau})} \in \mathbb{R}^n$ for some $\overline{\tau} \in \mathbb{R}^m$, $\overline{z} \in \mathbb{R}^m$ with $\overline{\tau} \approx_\epsilon \widetilde{\tau}$ and $\|\overline{z}-z\|_\infty \leq \epsilon$, where $x^{\flat(\overline{\tau})} := \mathrm{argmax}_{\|w\|_{\overline{\tau}^+\infty}\leq 1}\langle x, w\rangle$. The data-structure further returns the low dimensional representation $s \in \mathbb{R}^K$ such that*

$$\sum_{k=1}^K s_k \mathbf{1}_{i\in I_k} = \left(\nabla\Psi(\overline{z})^{\flat(\overline{\tau})}\right)_i$$

  for all $i \in [m]$, in $O(n\epsilon^{-2}\log n)$ time.
- *Potential*() *Returns $\Psi(z)$ in $O(1)$ time.*

The next lemma maintains the desired sum for QuerySum, which we need to slightly modify the corresponding result (Lemma 7.6 in [BLN+20]) to accomodate our per-coordinate accuracy requirement. The implementation of the data structure for the lemma is given in Algorithm 8. The input to the data structure is the $O(\epsilon^{-2}\log n)$ dimensional representations of the gradients of each iteration, which is computed by the data structure of the previous lemma. Here for a set $I \subset [m]$ we also use I as 0/1-vector with $I_i = 1$ when $i \in I$ and $I_i = 0$ otherwise.

Lemma D.3. *There exists a deterministic data-structure that supports the following operations*

- *Initialize* ($x^{(\mathrm{init})} \in \mathbb{R}^m, g \in \mathbb{R}^m, (I_k)_{1\leq k\leq K}, \epsilon \in (0,1]^m$)*: The data-structure initialized on the given vectors $x^{(\mathrm{init})}, g \in \mathbb{R}^m$, the partition $\bigcup_{k=1}^K I_k = [m]$ where $K = O(\epsilon^{-2}\log n)$, and the per-coordinate accuracy parameter $\epsilon \in (0,1]^m$ in $O(m)$ time.*
- *Scale*($i \in [m], a \in \mathbb{R}$)*: Sets $g_i \leftarrow a$ in $O(\log n)$ amortized time.*
- *Move*($i \in [m], k \in [1, K]$)*: Moves index i to set I_k in $O(\log n)$ amortized time.*
- *SetAccuracy*($i \in [m], \delta \in (0,1]$)*: Sets $\epsilon_i \leftarrow \delta$ in $O(\log n)$ amortized time.*
- *Query*($s \in \mathbb{R}^K, h \in \mathbb{R}^m$)*: Let $g^{(\ell)}$ and $\epsilon^{(\ell)}$ be the state of vector g and ϵ respectively during the ℓ-th call to Query and let $s^{(\ell)}$ and $h^{(\ell)}$ be the input arguments of the respective call. The vector h will always be provided as a sparse vector so that we know where are the non-zeros in the vector. Define $y^{(\ell)} = \mathbf{G}^{(\ell)}\sum_{k=1}^K I_k^{(\ell)} s_k^{(\ell)}$ and $x^{(t)} = x^{(\mathrm{init})} + \sum_{\ell=1}^t h^{(\ell)} + y^{(\ell)}$,*

then the t-th call to Query *returns a vector $\overline{x}$ satisfying $|\overline{x}_i - x_i^{(t)}| \le \epsilon_i^{(t)}$ for all $i \in [m]$. After T calls to* Query*, the total time of all T calls is bounded by*

$$O\left(TK + \log n \cdot \sum_{\ell=0}^{T} \|h^{(\ell)}\|_0 + T \log n \cdot \sum_{\ell=1}^{T} \|y^{(\ell)}/\epsilon^{(\ell-1)}\|_2^2\right).$$

The vector $\overline{x} \in \mathbb{R}^m$ is returned as a pointer and additionally a set $J \subset [m]$ is returned that contains the indices where $\overline{x}$ changed compared to the result of the previous Query *call.*

- *ComputeExactSum(): Returns the current exact vector $x^{(t)}$ in $O(m \log n)$ time.*

The proof follows a trivial adaptation of the proof for Lemma 7.6 in [BLN+20] and we reproduce it below for completeness.

Proof of Lemma D.3. We start by analyzing the correctness.

Invariant Let $s^{(t)}, h^{(t)}, g^{(t)}, I_k^{(t)}, \epsilon^{(t)}$ be the state of s, h, g, I_k, ϵ during the t-th call to Query and by definition of $x^{(t)}$ we have for any index i that

$$x_i^{(t)} = x_i^{(\text{init})} + \sum_{\ell=1}^{t} g_i^{(\ell)} \left(\sum_{k=1}^{K} s_k^{(\ell)} \mathbf{1}_{i \in I_k^{(\ell)}}\right) + h_i^{(\ell)}.$$

It is easy to check that $\widehat{\ell}_i$ always store the most recent iteration when $\overline{x}_i$ is updated by ComputeX(i, h_i). We first prove by induction that this update is always calculated correct, that is, the data-structure maintains the invariant $\overline{x}_i = x_i^{(\widehat{\ell}_i)}$.

We see from Line 32 that the data-structure maintains $f^{(t)} = \sum_{k=1}^{t} s^{(k)}$. Further note that ComputeX(i, h_i) is called whenever h_i is non-zero (Line 34), g_i or ϵ_i is changed, or i is moved to a different I_k. Thus if $\widehat{\ell}_i < t$, we know none of these events happened during iteration $\ell \in (\widehat{\ell}_i, t]$ and the only moving part is the $s^{(\ell)}$'s over these iterations, which is exactly $f^{(t)} - f^{(\widehat{\ell}_i)}$. Thus, if k is the set I_k where i belongs to over iterations $(\widehat{\ell}_i, t]$, the execution of Line 11 gives

$$g_i \cdot (f_k^{(t)} - f_k^{(\widehat{\ell}_i)}) + h_i^{(t)} = g_i \sum_{\ell=\widehat{\ell}_i+1}^{t} s_k^{(\ell)} + h_i^{(t)} = h_i^{(t)} + \sum_{\ell=\widehat{\ell}_i+1}^{t} g_i^{(\ell)} s_k^{(\ell)}$$

$$= h_i^{(t)} + \sum_{\ell=\widehat{\ell}_i+1}^{t} g_i^{(\ell)} \left(\sum_{k=1}^{K} s_k^{(\ell)} \mathbf{1}_{i \in I_k^{(\ell)}}\right) = \sum_{\ell=\widehat{\ell}_i+1}^{t} \left(g_i^{(\ell)} \left(\sum_{k=1}^{K} s_k^{(\ell)} \mathbf{1}_{i \in I_k^{(\ell)}}\right) + h_i^{(\ell)}\right)$$

where the first equality uses $f^{(t)} = \sum_{\ell=1}^{t} s^{(\ell)}$ and the second equality uses $g_i^{(\ell)} = g_i^{(t)}$ for all $\widehat{\ell}_i < \ell \le t$, because ComputeX$(i, h_i)$ is called whenever g_i is changed. The third equality is because ComputeX(i, h_i) is called whenever i is moved to a different set, so $i \in I_k^{(\ell)}$ for the same k for all $\widehat{\ell}_i < \ell \le t$. The last equality uses $h_i^{(\ell)} = 0$ for $\widehat{\ell}_i < \ell < t$, because ComputeX$(i, h_i)$ is called whenever h_i is non-zero. Thus by induction over the number of calls to ComputeX(i, h_i), when $\widehat{\ell}_i$ is increased to t we have

$$\overline{x}_i = x_i^{(t)} = x_i^{(\text{init})} + \sum_{\ell=1}^{t} \left(g_i^{(\ell)} \left(\sum_{k=1}^{K} s_k^{(\ell)} \mathbf{1}_{i \in I_k^{(\ell)}}\right) + h_i^{(\ell)}\right) = x_i^{(t)},$$

so the invariant is always maintained.

Algorithm 8: Algorithm for accumulating $\mathbf{G}\nabla\Psi(\overline{v})^\flat$ (Lemma D.3)

members
  $I_1, ..., I_K$; `// Partition` $\bigcup_k I_k = [m]$
  $t \in \mathbb{N}, \overline{x} \in \mathbb{R}^m$; `// QUERY counter and approximation of` $x^{(t)}$
  $\widehat{\ell} \in \mathbb{N}^m$; `//` $\widehat{\ell}_i$ `is value of` t `when we last update` $\overline{x}_i \leftarrow x_i$
  $f^{(t)} \in \mathbb{R}^K$; `// Maintain` $f^{(t)} = \sum_{k=1}^t s^{(k)}$
  $\Delta^{(high)}, \Delta^{(low)} \in \mathbb{R}^m$; `// Maintain` $\Delta_i = f_k^{(\widehat{\ell}_i)} \pm |\epsilon_i/(10g_i)|$ `if` $i \in I_k$
procedure *Initialize*$(x^{(\text{init})} \in \mathbb{R}^m, g \in \mathbb{R}^m, (I_k)_{1\le k\le K}, \epsilon \in (0,1]^m)$
  $\overline{x} \leftarrow x^{(\text{init})}$, $(I_k)_{1\le k\le K} \leftarrow (I_k)_{1\le k\le K}$, $t \leftarrow 0$, $f^{(0)} \leftarrow \vec{0}_K$, $g \leftarrow g$, $\epsilon \leftarrow \epsilon$
private procedure *ComputeX*(i, h_i)
  Let k be such that $i \in I_k$
  $\overline{x}_i \leftarrow \overline{x}_i + g_i \cdot (f_k^{(t)} - f_k^{(\widehat{\ell}_i)}) + h_i$
  $\widehat{\ell}_i \leftarrow t$
  $J \leftarrow J \cup \{i\}$
private procedure *UpdateDelta*(i)
  Let k be such that $i \in I_k$.
  $\Delta_i^{(high)} \leftarrow f_k^{(\widehat{\ell}_i)} + |\epsilon_i/(10g_i)|$
  $\Delta_i^{(low)} \leftarrow f_k^{(\widehat{\ell}_i)} - |\epsilon_i/(10g_i)|$
procedure *Move*$(i \in [m], k)$
  ComputeX$(i, 0)$
  Move index i to set I_k
  UpdateDelta(i)
private procedure *Scale*(i, a)
  ComputeX$(i, 0)$
  $g_i \leftarrow a$
  UpdateDelta(i)
private procedure *SetAccuracy*(i, δ)
  ComputeX$(i, 0)$
  $\epsilon_i \leftarrow \delta$
  UpdateDelta(i)
procedure *Query*$(s \in \mathbb{R}^K, h \in \mathbb{R}^m)$
  $t \leftarrow t + 1$, $J \leftarrow \emptyset$; `// Collect all entries that have changed since the last call to QUERY`
  $f^{(t)} \leftarrow f^{(t-1)} + s$
  for *i such that $h_i \neq 0$* **do**
    ComputeX(i, h_i), UpdateDelta(i)
  for $k = 1, \ldots, K$ **do**
    for *$i \in I_k$ with $f_k^{(t)} > \Delta_i^{(high)}$ or $f_k^{(t)} < \Delta_i^{(low)}$* **do**
      ComputeX$(i, 0)$, UpdateDelta(i)
  return $\overline{x}$, J
procedure *ComputeExactSum*$()$
  for *$i \in [m]$ and $\widehat{\ell}_i < t$* **do**
    ComputeX$(i, 0)$, UpdateDelta(i)
  return $\overline{x}$

Correctness of Query. We claim that the function QUERY returns a vector $\overline{x}$ such that for all i

$$\overline{x}_i \in [x_i^{(t)} \pm \epsilon_i^{(t)}] := \left[x_i^{(\text{init})} + \sum_{\ell=1}^{t} \left(g_i^{(\ell)} \left(\sum_{k=1}^{K} s_k^{(\ell)} \mathbf{1}_{i \in I_k^{(\ell)}} \right) + h_i^{(\ell)} \right) \pm \epsilon_i^{(t)} \right].$$

Given the invariant discussed above, we only need to guarantee COMPUTEEXACT(i, h_i) is called whenever the approximation guarantee is violated for some i. Moreover, same as when we proved the invariant above, we only need to guarantee this in the case that since iteration $\widehat{\ell}_i$, h_i is always 0, g_i, ϵ_i remain constant and i remains in the same I_k for some k. Thus, the task is equivalent to detect whenever

$$\left| \sum_{\ell=\widehat{\ell}_i+1}^{t} g_i^{(\ell)} s_k^{(\ell)} \right| = |g_i \cdot (f_k^{(t)} - f_k^{(\widehat{\ell}_i)})| > \frac{\epsilon_i^{(t)}}{10}.$$

which is the same as

$$f_k^{(t)} \notin [f_k^{(\widehat{\ell}_i)} - |\epsilon_i/(10g_i)|, f_k^{(\widehat{\ell}_i)} + |\epsilon_i/(10g_i)|]$$

Note that the lower and upper limits in the above range are exactly $\Delta_i^{(high)}$ and $\Delta_i^{(low)}$ as maintained by UPDATEDELTA(i), which will be called whenever any of the terms involved in the calculation of these limits changes. Thus Line 37 makes sure that we indeed maintain

$$|\overline{x}_i - x_i^{(t)}| \le \epsilon_i \qquad \forall i.$$

Also it is easy to check the returned set J contains all i's such that $\overline{x}_i$ changed since last QUERY.

Complexity The call to INITIALIZE takes $O(m + K)$ as we initialize a constant number of K and m dimensional vectors, and this reduces to $O(m)$ since there can be at most m non-empty I_k's. A call to COMPUTEX takes $O(1)$ time.

To implement Line 37 efficiently without enumerating all m indices, we maintain for each $k \in [K]$ two sorted lists of the i's in I_k, sorted by $\Delta_i^{(high)}$ and $\Delta_i^{(low)}$ respectively. Maintaining these sorted lists results in $O(\log n)$ time per call for UPDATEDELTA. Hence MOVE, SCALE and SETACCURACY also run in $O(\log n)$ time. To implement the loop for Line 37 we can go through the two sorted lists in order, but stop as soon as the check condition no longer holds. This bounds the cost of the loop by $O(K)$ plus $O(\log n)$ times the number of indices i satisfying $f_k^{(t)} > \Delta_i^{(high)}$ or $f^{(t)} < \Delta_i^{(low)}$, i.e. $|f_k^{(t)} - f_k^{(\widehat{\ell}_i)}| > \Theta(\epsilon_i^{(\widehat{\ell}_i)}/g_i^{(\widehat{\ell}_i)})$. Note if a COMPUTEX and UPDATEDELTA is triggered by this condition for any i, h_i must be 0 during $(\widehat{\ell}_i, t]$ iterations. Thus, let $z^{(t)} := x^{(\text{init})} + \sum_{\ell=1}^{t} \mathbf{G}^{(\ell)} \sum_k I_k^{(\ell)} s_k^{(\ell)}$, we can rewrite that condition as $|z_i^{(t)} - z_i^{(\widehat{\ell}_i)}| > \Theta(|\epsilon_i^{(\widehat{\ell}_i)}|)$. Throughout T calls to QUERY, we can bound the total number of times where i satisfies $|z_i^{(t)} - z_i^{(\widehat{\ell}_i)}| > \Theta(|\epsilon_i^{(\widehat{\ell}_i)}|)$ by

$$O\left(T \sum_{\ell=1}^{T} \|\mathbf{G}^{(\ell)} (\sum_k I_k^{(\ell)} s_k^{(\ell)}) / \epsilon^{(\ell-1)}\|_2^2 \right).$$

The number of times COMPUTEX and UPDATEDELTA are triggered due to $h_i^{(t)} \neq 0$ is $\|h^{(t)}\|_0$ each iteration, and updating $f^{(t)}$ takes $O(K)$ time. So the total time for T calls to QUERY can be bounded by

$$O(TK + \log(n) \cdot \sum_{\ell=0}^{T} \|h^{(\ell)}\|_0 + \log(n) \cdot T \sum_{\ell=1}^{T} \|\mathbf{G}^{(\ell)} (\sum_k s_k^{(\ell)} I_k^{(\ell)}) / \epsilon^{(\ell-1)}\|_2^2 / \epsilon^2).$$

The time for COMPUTEEXACTSUM is $O(m \log n)$ since it just calls COMPUTEX and UPDATEDELTA on all m indices. □

Proof of Theorem D.1. The data-structure for Theorem D.1 follows directly by combining Lemma D.2 and Lemma D.3. The result for QueryProduct is obtained from Lemma D.2, and the result for QuerySum is obtained from Lemma D.3 using the vector $s \in \mathbb{R}^K$ returned by Lemma D.2 as input to Lemma D.3 and ϵ_i being $w_i\epsilon$. Note we charge the cost incurred by calling the data structure of Lemma D.3 to the QuerySum complexity. □

E Dual Slack Maintenance

Theorem E.1 (Dual Maintenance)**.** *Assuming a (P, z, Q)-HeavyHitter data structure as in Definition 3.1, there exists a data-structure (Algorithm 9) that supports the following operations. Note in the bounds we use $\widetilde{O}$ to hide polynomials in $\log(nP/\|z\|_1)$ in addition to $\log n$ factors, and in our instantiations of the data structure the former factor will be bounded by $\log n$.*

- *InitializE($\mathbf{A} \in \mathbb{R}^{m\times n}, v^{(\text{init})} \in \mathbb{R}^m, w^{(\text{init})} \in [0,1]^m, \epsilon \in [0,1]$) The data-structure pre-processes the given matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, the vector $v^{(\text{init})} \in \mathbb{R}^m$ and accuracy vector $0 < w^{(\text{init})} \leq 1$ in $\widetilde{O}(P)$ time.*
- *SetAccuracy(i, δ): Sets $w_i \leftarrow \delta$ in $\widetilde{O}(z_i)$ amortized time.*
- *Add($h \in \mathbb{R}^n$): Suppose this is the t-th time the Add operations is called, and let $h^{(k)}$ be the vector h given when the Add operation is called for the k^{th} time. Define $v^{(t)} \in \mathbb{R}^m$ to be the vector*

$$v^{(t)} = v^{(\text{init})} + \mathbf{A}\sum_{k=1}^{t} h^{(k)}.$$

Then the data structure returns a vector $\overline{v}^{(t)} \in \mathbb{R}^m$ such that $\|w^{-1}(\overline{v}^{(t)} - v^{(t)})\|_\infty \leq \epsilon$. The output will be in a compact representation to reduce the size. In particular, the data-structure returns a pointer to $\overline{v}$ and a set $I \subset [m]$ of indices i where $\overline{v}_i^{(t)}$ is changed compared to $\overline{v}_i^{(t-1)}$, i.e., the result of the previous call to Add. The amortized time for the t-th call to Add is

$$\widetilde{O}\left(Q + \sqrt{nP/\|z\|_1} \cdot \|(v^{(t)} - v^{(t-1)})/w^{(t)}\|_z^2 \epsilon^{-2} + \sqrt{\|z\|_1 P/n}\right).$$

- *ComputeExact(): Returns $v^{(t)} \in \mathbb{R}^m$ in $O(\text{nnz}(\mathbf{A}))$ time, where t is the number of times Add is called so far (i.e., $v^{(t)}$ is the state of the exact vector v after the most recent call to Add).*

The data structure for the theorem above is given in Algorithm 9. Both the implementation and analysis follow straightforward adaptations of Algorithm 4 and Theorem 6.1 in [BLN+20], and we reproduce the analysis below for completeness. Our new version makes the formal reduction to HeavyHitters (Definition 3.1) more clear and adds the SetAccuracy method. Throughout this section we denote $h^{(t)}$ the input vector h of the t-th call to Add (or equivalently referred to as the t-th iteration), and let $v^{(t)} = v^{(\text{init})} + \mathbf{A}\sum_{k=1}^{t} h^{(k)}$ be the state of the exact solution v (as defined in Theorem E.1) for the t-th call to Add.

In our algorithm (see Algorithm 9) we maintain a vector $\widehat{f}$ which is the sum of all past input vectors h, so we can retrieve the exact value of $v_i^{(t)} = v_i^{(\text{init})} + (\mathbf{A}\widehat{f})_i$ for any i efficiently. This value is computed and assigned to $\overline{v}_i$ whenever the approximation $\overline{v}$ that we maintain no longer satisfies the error guarantee for some coordinate i. As to how we detect when this may happen, we know the difference between $v^{(t)}$ and the state of v at an earlier t'-th Add call is

$$v^{(t)} - v^{(t')} = \mathbf{A}\left(\sum_{k=t'+1}^{t} h^{(t)}\right),$$

and thus we can detect all coordinates i that changes above certain threshold from t' to t-th Add call using the (P, z, Q)-HeavyHitter data structure of Definition 3.1 (by querying it with $\sum_{k=t'+1}^{t} h^{(t)}$ as the parameter h). Note since the error guarantee we want is multiplicative in w (i.e., $\overline{v}_i^{(t)} \in v_i^{(t)} \pm \epsilon w_i$ for all i), while the threshold ϵ in Definition 3.1 is absolute and uniform, we give w^{-1} as the scaling vector to the HeavyHitter data structure to accommodate this.

Since the most recent updates on $\overline{v}_i$ for different indices i's happen at different iterations, we need to track accumulated changes to v_i's over different intervals to detect the next time

Algorithm 9: Algorithm for Theorem E.1

members
  $\widehat{f}, \overline{v} \in \mathbb{R}^m, w \in \mathbb{R}^m, t \in \mathbb{N}$
  $D_j, T = \sqrt{nP/\|z\|_1}$ // D_j are (P, z, Q)-HeavyHitter (Definition 3.1)
  $f^{(j)} \in \mathbb{R}^n$ and $F_j \subset [m]$ for $0 \le j \le \log T$
procedure *Initialize*$(\mathbf{A}, v^{(\text{init})}, w^{(\text{init})}, \epsilon)$
  $\overline{v} \leftarrow v^{(\text{init})}$, $\widehat{f} \leftarrow \vec{0}_n$, $w \leftarrow w^{(\text{init})}$, $t \leftarrow 0$
  for $j = 0, \ldots, \log T$ **do**
    D_j.Initialize$(\mathbf{A}, w^{-1})$ (Definition 3.1)
    $f^{(j)} \leftarrow \vec{0}_n$, $F_j \leftarrow \emptyset$
  return $\mathbf{A}v^{(\text{init})}$
private procedure *FindIndices*$(h \in \mathbb{R}^n)$
  $I \leftarrow \emptyset$
  for $j = \log T, ..., 0$ **do**
    $f^{(j)} \leftarrow f^{(j)} + h$ // When $2^j | t$, then $f^{(j)} = \sum_{k=t-2^j+1}^{t} h^{(k)}$
    if $2^j | t$ **then**
      $I \leftarrow I \cup D_j$.HeavyQuery$(f^{(j)}, 0.2\epsilon/\log n)$
      $f^{(j)} \leftarrow \vec{0}_n$
  return I
procedure *SetAccuracy*(i, δ)
  $w_i \leftarrow \delta$
  for $j = 0, ..., \log T$ **do** $F_j \leftarrow F_j \cup \{i\}$, D_j.Scale$(i, 0)$;
private procedure *VerifyIndex*(i)
  if $|\overline{v}_i - (v^{(\text{init})} + \mathbf{A}\widehat{f})_i| \ge 0.2 w_i \epsilon / \log n$ **then**
    $\overline{v}_i \leftarrow (v^{(\text{init})} + \mathbf{A}\widehat{f})_i$
    for $j = 0, ..., \log T$ **do**
      $F_j \leftarrow F_j \cup \{i\}$ // Notify other D_j's to stop tracking i.
      D_j.Scale$(i, 0)$
    return True
  return False
procedure *Add*$(h \in \mathbb{R}^n)$
  if $t = T$ **then return** Initialize$(\mathbf{A}, \mathbf{A}(\widehat{f} + h), w, \epsilon)$;
  $t \leftarrow t + 1$, $\widehat{f} \leftarrow \widehat{f} + h$, $I \leftarrow$ FindIndices(h)
  $I \leftarrow \{i | i \in I$ and VerifyIndex$(i) = True\}$
  for $j : 2^j | t$ **do** $I \leftarrow I \cup \{i | i \in F_j$ and VerifyIndex$(i) = True\}$;
  for $j : 2^j | t$ **do**
    for $i \in I \cup F_j$ **do**
      D_j.Scale$(i, 1/w_i)$
    $F_j \leftarrow \emptyset$
  return $I, \overline{v}$
procedure *ComputeExact*$()$
  return $v^{(\text{init})} + \mathbf{A}\widehat{f}$

an update is necessary for each i. Thus, it is not sufficient to just have one copy of the HeavyHitter. On the other hand, keeping one individual copy of the HeavyHitter for each $0 \leq t' < t$ will be too costly in terms of running time. We handle this by instantiating $\log T$ copies of the HeavyHitter data structure D_j for $j = 0, ..., \log T$ where $T = \sqrt{nP/\|z\|_1}$, and each copy takes charge of batches with increasing number of iterations. In particular, the purpose of D_j is to detect all coordinates i in v with large accumulated change over batches of 2^j iterations (see how we update and reset $f^{(j)}$ in FindIndices in Algorithm 9). Each D_j has its local copy of a scaling vector, which is initialized to be w^{-1}, we refer to it as $\widehat{g}^{(j)}$, and the cost to query D_j is proportional to $\|\widehat{\mathbf{G}}^{(j)}\mathbf{A}f^{(j)}\|_z^2 + Q$. Note $f^{(j)}$ accumulates updates over 2^j iterations, and $\|\sum_{k=1}^{2^j} h^{(k)}\|_z^2$ can be as large as $2^j \sum_{k=1}^{2^j} \|h^{(k)}\|_z^2$. Since we want to bound the cost of our data structure by the sum of the squares of updates (which can in turn be bounded by our IPM method) instead of the square of the sum of updates, querying D_j incurs an additional 2^j factor overhead. Thus for efficiency purposes, if v_i would take much less than 2^j iterations to accumulate a large enough change, we can safely let D_j stop tracking i during its current batch since v_i's change would have been detected by a $D_{j'}$ of appropriate (and much smaller) j' so that $\overline{v}_i$ would have been updated to be the exact value (see implementation of VerifyIndex). Technically, we keep a set F_j to store all indices i that D_j stops tracking for its current batch of iterations and set $\widehat{g}_i^{(j)}$ to 0 so we don't pay for coordinate i when we query D_j. Also note that whenever the accuracy requirement w_i for coordinate i changes, we add i to all F_j's (see implementation of SetAccuracy), and this will make sure we explicitly check whether $\overline{v}_i$ is within the approximation requirement when Add is called since we will call VerifyIndex on i (see Line 33). At the start of a new batch of 2^j iterations for D_j, we add back all indices in F_j to D_j (Line 37) and reset F_j. As a result, only those i's that indeed would take (close to) 2^j iterations to accumulate a large enough change are necessary to be tracked by D_j, so we can query D_j less often for large j to offset its large cost. In particular, we query each D_j every 2^j iterations (see Line 16).

We start our formal analysis with the following lemma, which adapts Lemma 6.2 in [BLN+20].

Lemma E.2. *Suppose we perform the t-th call to Add. Then the call to FindIndices in Line 32 returns a set $I \subset [m]$ containing all $i \in [m]$ such that there exists some j with $2^j | t$ satisfying both $i \notin F_j$ and $|v_i^{(t-2^j)} - v_i^{(t)}| \geq 0.2 w_i^{(t)} \epsilon / \log n$.*

Proof. Pick any j with $2^j | t$, if $|v_i^{(t-2^j)} - v_i^{(t)}| \geq 0.2 w_i^{(t)} \epsilon / \log n$, then

$$\left| e_i^\top \mathbf{A} \sum_{k=t-2^j+1}^{t} h^{(k)} \right| \geq 0.2 w_i^{(t)} \epsilon / \log n.$$

We will argue why FindIndices detect all i's satisfying this condition.

Note that we have $f^{(j)} = \sum_{k=t-2^j+1}^{t} h^{(k)}$, and thus by guarantee of Definition 3.1 when we call D_j.HeavyQuery$(f^{(j)}, 0.2\epsilon/\log n)$ (in Line 16), we obtain for every j with $2^j | t$ and all $i \in [m]$ with

$$\left| \widehat{g}_i^{(j)} e_i^\top \mathbf{A} \sum_{k=t-2^j+1}^{t} h^{(k)} \right| \geq 0.2\epsilon / \log n.$$

Here $\widehat{g}_i^{(j)} = 0$ if $i \in F_j$ due to change of w_i in SetAccuracy or in Line 27, which happens whenever $\overline{v}_i$ is changed in Line 24. Thus by Line 37 we have $\widehat{g}_i^{(j)} = 1/w_i^{(t)}$ for all $i \notin F_j$. Equivalently, we obtain all indices $i \notin F_j$ satisfying the following condition, which proves the lemma.

$$\left| e_i^\top \mathbf{A} \left(\sum_{k=t-2^j+1}^{t} h^{(k)} \right) \right| \geq 0.2 w_i^{(t)} \epsilon / \log n$$

□

To guarantee that the approximation $\overline{v}$ we maintain is within the required ϵ error bound of the exact vector v, we need to argue that the D_j's altogether are sufficient to detect all potential events that would cause $\overline{v}_i$ to become outside of $v \pm \epsilon w$. It is easy to see that if an index i is included in the returned set I of FindIndices (Line 32), then our algorithm will follow up with a call to VerifyIndex(i), which will guarantee that $\overline{v}_i$ is close to the exact value v_i (or $\overline{v}_i$ will be updated to be v_i). Thus, if we are in iteration t, and $\overline{t}$ is the most recent time VerifyIndex(i) is called, we know $\overline{v}_i$ satisfies the approximation guarantee for $v_i^{(\overline{t})}$, the value of $\overline{v}_i$ remains the same since iteration $\overline{t}$, and the index i is not in the result of FindIndices for any of the iterations after $\overline{t}$. We will demonstrate the last condition is sufficient to show $v_i^{(\overline{t})} \approx v_i^{(t)}$, which in turn will prove $\overline{v}_i \approx v_i^{(t)}$. To start, we first need to argue that for any two iterations $\overline{t} < t$, the interval can be partitioned into a small number of batches such that each batch is exactly one of the batches tracked by some D_j. We cite without proof the following lemma from [BLN+20].

Lemma E.3 ([BLN+20, Lemma 6.3]). *Given any $\overline{t} < t$, there exists a sequence*

$$t = t_0 > t_1 > \ldots > t_k = \overline{t}$$

such that $k \le 2 \log t$ and $t_{x+1} = t_x - 2^{\ell_x}$ where ℓ_x satisfies $2^{\ell_x} | t_x$ for all $x = 0, \ldots, k-1$.

Now we can argue $\overline{v}$ stays in the desired approximation range around v with the same argument (with slight adaptation) of Lemma 6.4 in [BLN+20].

Lemma E.4 (Correctness of Theorem E.1). *Assume we perform the t-th call to Add, the returned vector $\overline{v}$ satisfies $|v_i^{(t)} - \overline{v}_i| \le \epsilon w_i^{(t)}$ for all $i \in [m]$, and I contains all indices that have changed since the $(t-1)$-th Add call.*

Proof. By $\widehat{f} = \sum_{j=1}^{t} h^{(j)}$ we have $v^{(\text{init})} + \mathbf{A}\widehat{f} = v^{(t)}$ in Line 23. So after a call to VerifyIndex(i) we know that $|\overline{v}_i - v_i^{(t)}| < 0.2\epsilon w_i^{(t)} / \log n$, either because the comparison $|\overline{v}_i - (v^{(\text{init})} + \mathbf{A}\widehat{f})_i| \ge 0.2\epsilon w_i / \log n$ in Line 23 returned false, or because we set $\overline{v}_i \leftarrow (v^{(\text{init})} + \mathbf{A}\widehat{f})_i$ in Line 24. Note this is also the only place we may change $\overline{v}_i$. So consider some time $\overline{t} \le t$ when VerifyIndex(i) was called for the last time (alternatively $\overline{t} = 0$). Then $\overline{v}_i$ has not changed during the past $t - \overline{t}$ calls to Add, and we know $|\overline{v}_i - v_i^{(\overline{t})}| \le 0.2\epsilon w_i / \log n$. We now want to argue that $|v^{(\overline{t})} - v^{(t)}| \le 0.2\epsilon w_i$, which via triangle inequality would then imply $|\overline{v}_i - v_i^{(t)}| \le \epsilon w_i$. Note here we can omit the superscript of w_i since it has not changed during the past $t - \overline{t}$ calls to Add, since otherwise i would have been added to all D_j's (particularly D_0), which would have triggered a call to VerifyIndex in Line 34.

For $t = \overline{t}$ this is obvious, so consider $\overline{t} < t$. We know from Lemma E.3 the existence of a sequence

$$t = t_0 > t_1 > \ldots > t_k = \overline{t}$$

with $2^{\ell_x} | t_x$ and $t_{x+1} = t_x - 2^{\ell_x}$. In particular, this means that the interval between iteration t_{x+1} and t_x correspond to exactly a batch tracked by D_{ℓ_x}. Thus, at iteration t_x when FindIndices is called, D_{ℓ_x}.HeavyQuery is executed in Line 16. This gives us $|v_i^{(t_x)} - v_i^{(t_{x+1})}| < 0.2\epsilon w_i / \log n$ for all x, because by Lemma E.2 the set $I \cup (\bigcup_j F_j)$ contains all indices i which might have changed by $0.2 w_i \epsilon / \log n$ over the past 2^ℓ iterations for any $2^\ell | t$, and because VerifyIndex(i) is called for all $i \in I \cup (\bigcup_j F_j)$ in Line 33 and Line 34.

Note we can assume $\log t \le \log T$ since we reset the data-structure every $T = \sqrt{nP/\|z\|_1}$ iterations, and this bounds the length of the sequence $k \le 2 \log T$. This then yields the bound

$$|v_i^{(\overline{t})} - v_i^{(t)}| = |v_i^{(t_k)} - v_i^{(t_0)}| \le \sum_{x=1}^{k} |v_i^{(t_x)} - v_i^{(t_{x-1})}| \le k \cdot 0.2\epsilon w_i / \log T \le 0.4\epsilon w_i$$

Thus we have $|\overline{v}_i - v_i^{(t)}| \leq (0.4\epsilon + 0.2\epsilon/\log n)w_i$, which satisfies our approximation guarantee. It is also straightforward to check that when we return the set I at the end of ADD, I contains all the i's where VERIFYINDEX(i) is called and returned true in this iteration, which are exactly all the i's where $\overline{v}_i$'s are changed in Line 24. □

Now we proceed to the complexity of our data structure. We start with the cost of FINDINDICES, which is mainly on the cost of querying D_j's. As we discussed at the beginning, there can be a large overhead for large j, but this is compensated by querying large j less frequently. The following is a straightforward adaptation of Lemma 6.5 in [BLN+20]. Note that SETACCURACY also triggers VERIFYINDEX (indirectly in Add since we add i to all F_j's whenever we change w_i). We attribute the cost of these VERIFYINDEX calls to the amortized running time of SETACCURACY instead of counting it in the time spent in VERIFYINDEX.

Lemma E.5. *After T calls to ADD, the total time spent in FINDINDICES and VERIFYINDEX is bounded by*

$$\widetilde{O}\left(T\epsilon^{-2}\sum_{t=1}^{T}\|(v^{(t)} - v^{(t-1)})/w^{(t)}\|_z^2 + TQ\right)$$

Proof. We start with the cost of FINDINDICES. Every call to ADD invokes a call to FINDINDICES, so we denote the t-th call to FINDINDICES as the one associated with the t-th ADD. Fix any j and consider the cost for D_j. We update $f^{(j)}$ once in each call, which takes $O(n) = O(Q)$ ($Q \geq n$ because the Heavy Hitter data structure needs to read the input which costs $O(n)$ time). Every 2^j calls would incur the cost to D_j.HEAVYQUERY($f^{(j)}$). Without loss of generality we consider the cost of the first time this happens (at iteration 2^j) since the other batches follow the same calculation. We denote $\widehat{g}^{(j)}$ as the scaling vector in D_j when the query happens. We know $\widehat{g}_i^{(j)} = 0$ if $i \in F_j$, and $\widehat{g}_i^{(j)} = 1/w_i$ otherwise. Note here we can skip the superscript indicating the iteration number, since if $i \notin F_j$ it must be that $\overline{v}_i$ and w_i have not changed over the 2^j iterations. The cost to query D_j in Line 16 can then be bounded by.

$$\begin{aligned}
\widetilde{O}(\|\widehat{\mathbf{G}}^{(j)}\mathbf{A}f^{(j)}\|_z^2\epsilon^{-2} + Q) &\leq \widetilde{O}(\|\sum_{t=1}^{2^j}\mathbf{Diag}(1/w^{(t)})\mathbf{A}h^{(t)}\|_z^2\epsilon^{-2} + Q) \\
&\leq \widetilde{O}\left(\left(\sum_{t=1}^{2^j}\|\mathbf{Diag}(1/w^{(t)})\mathbf{A}h^{(t)}\|_z\right)^2\epsilon^{-2} + Q\right) \\
&\leq \widetilde{O}\left(2^j\cdot\sum_{t=1}^{2^j}\|\mathbf{Diag}(1/w^{(t)})\mathbf{A}h^{(t)}\|_z^2\epsilon^{-2} + Q\right)
\end{aligned}$$

The first line is by Definition 3.1, and note we use $0.2\epsilon/\log n$ as the error parameter in the call. The first inequality is by looking at $\widehat{g}_i^{(j)}$ for each index i separately. The value is either 0 so replacing it by $1/w_i^{(t)}$ only increase the norm, or we know $i \notin F_j$, so $\widehat{g}_i^{(j)} = 1/w_i^{(t)}$ for all $t \in [1, 2^j]$. The second inequality uses the triangle inequality, and the third inequality uses Cauchy-Schwarz. The cost of all subsequent queries to D_j follows similar calculation, and as this query is only performed once every 2^j iterations, the total time after T iterations is

$$\widetilde{O}\left(T\epsilon^{-2}\sum_{t=1}^{T}\|\mathbf{Diag}(1/w^{(t)})\mathbf{A}h^{(t)}\|_z^2 + T2^{-j}Q\right) = \widetilde{O}\left(T\epsilon^{-2}\sum_{t=1}^{T}\|(v^{(t)} - v^{(t-1)})/w^{(t)}\|_z^2 + T2^{-j}Q\right)$$

Note that the equality follows the definition of $v^{(t)}$ in Theorem E.1. We can then sum over the total cost for all D_j's as well as updating $f^{(j)}$ for $j = 1, \ldots, \log T$ to get the final running time bound in the lemma statement.

As to the cost of VerifyIndex(i), each call computes $(v^{(\text{init})}+\mathbf{A}\widehat{f})_i$ which takes $O(\text{nnz}(a_i)) \leq z_i$ time as each row of $\mathbf{A}$ has $\text{nnz}(a_i)$ non-zero entries. Further, the updates to F_j's and calls to D_j.Scale for all j's take $O(z_i \log T)$ time. Now we need to bound the total number of times we call VerifyIndex for some i, which can only happen in two cases. The first case (Line 33) is when i is returned by FindIndices in Line 32, and the total time we spent in VerifyIndex is bounded by $O(\sum_{i\in I} z_i \log T)$. By the guarantee of $\sum_{i\in I} z_i$ given in Definition 3.1 for the QueryHeavy calls, we know the total cost over T iterations of such VerifyIndex calls is bounded by the toal running time of FindIndices (up to a $\log T$ factor). The second case (Line 34) is when i is in some F_j because $\overline{v}_i$ was updated due to v_i changing by more than $0.2\epsilon w_i/\log n$ (or w_i is updated, but we count such cost separately in SetAccuracy), and the total cost can be bounded by

$$\widetilde{O}\left(T\epsilon^{-2}\sum_{t=1}^{T}\|(v^{(t)}-v^{(t-1)})/w^{(t)}\|_z^2\right).$$

Adding up the total cost of VerifyIndex and FindIndices proves the lemma. □

We proceed to prove the complexity bounds in Theorem E.1. We will set T to be $\sqrt{nP/\|z\|_1}$ and re-initialize the data structure every T Add calls.

Initialize. The main work is to initialize the data structures D_j for $j = 1, ..., \log T$, which takes $\widetilde{O}(P)$ time in total by Definition 3.1.

SetAccuracy. The main work is to call D_j.Scale for all j's, and this takes $\widetilde{O}(z_i)$ time. Recall we also add i to all F_j's, which triggers VerifyIndex later when we call Add. Technically for i's that are added to all F_j's due to SetAccuracy we can flag them and skip the inner for loop in VerifyIndex since the loop does nothing to these i's. Thus, the total time to just update $\overline{v}_i$ is $O(\text{nnz}(a_i))$ which is bounded by $O(z_i)$. We have another D_j.Scale cost incurred in Add when we restart D_j every 2^j iterations (see line Line 37), and such cost is bounded by $\widetilde{O}(z_i)$.

Add. The cost not associated with any FindIndices and VerifyIndex is $O(n)$. Together with Lemma E.5 gives the bound of total time of T calls to Add

$$\widetilde{O}\left(T\epsilon^{-2}\sum_{t=1}^{T}\|(v^{(t)}-v^{(t-1)})/w^{(t)}\|_2^2 + TQ\right)$$

Moreover, the cost to reinitialize the data structure (once every T iterations) is $\widetilde{O}(P)$. Together with the bound above we get the amortized time specified in the theorem statement.

ComputeExact. This just takes $O(\text{nnz}(A))$ to compute the matrix-vector product.

F Graph Data Structures

In this section we formally state the data structure results we need to efficient implement our interior point method to show Theorem 1.4 in Section 7.

Lemma F.1 ([BLN+20, Lemma 5.1]). *There exists a (P, c, Q)-HeavyHitter data structure (Definition 3.1) for matrices $\mathbf{A}$, where $\mathbf{A}$ is obtained by removing one column from an incidence matrix of a directed graph with $n+1$ vertices and m edges, $P = \widetilde{O}(m)$, $c_i = \widetilde{O}(1)$ for all $i \in [m]$, and $Q = \widetilde{O}(n \log W)$ where W is the ratio of the largest to smallest non-zero entry in g.*

Proof. If $\mathbf{A}$ is an incidence matrix, then [BLN+20, Lemma 5.1] yields a (P, c, Q)-HeavyHitter with P, c, Q as stated in Lemma F.1. If we remove one column from $\mathbf{A}$, then the remaining matrix can be considered an incidence matrix with at most n additional rows that contain

only a single non-zero entry (i.e $+1$ or -1). For the rows that form an incidence matrix we use [BLN+20, Lemma 5.1], while for the remaining n rows we compute $(\mathbf{G}\mathbf{A}h)_i$ explicitly and return the index i if $|(\mathbf{G}\mathbf{A}h)_i| > \epsilon$. This additional explicit computation requires only $O(n)$ time which is subsumed by Q. □

Lemma F.2. *There exists a (P, c, Q)-InverseMaintenance data structure for matrices $\mathbf{A} \in \mathbb{R}^{m\times n}$, where $\mathbf{A}$ is obtained by removing one column from an incidence matrix of a directed graph with $n+1$ vertices and m edges, $P = \widetilde{O}(m)$, $c_i = 1$ for all $i \in [m]$, and $Q = n$.*

Proof. For any diagonal matrix $\mathbf{V} \in \mathbb{R}^{m\times m}_{\geq 0}$ we have that $\mathbf{A}^\top\mathbf{V}\mathbf{A}$ is a symmetric diagonally dominant matrix. For such matrices there exist nearly-linear time solvers, e.g. Lemma 7.1. Thus Initialize consists of reading the matrix. During Update no operation is performed and for Solve we use Lemma 7.1. □

We use the following lemma which follows from directly from Lemma 5.1 and 8.2 in [BLN+20].

Lemma F.3 ([BLN+20, Lemma 5.1 and 8.2])**.** *There exists the following data structure*

- *Initialize($\mathbf{A} \in \mathbb{R}^{m\times n}, g \in \mathbb{R}^m_{>0}, \overline{\tau}$) Initializes the data structure on the given matrix $\mathbf{A}$ in $\widetilde{O}(m)$ time, where $\mathbf{A}$ is obtained by removing one column from an incidence matrix of a directed graph with $n+1$ vertices and m edges.*
- *Scale(i, a, b) Sets $g_i \leftarrow a$ and $\overline{\tau}_i \leftarrow i$ in $\widetilde{O}(1)$ time.*
- *Sample($h \in \mathbb{R}^n, C_1, C_2, C_3$) Returns a random diagonal matrix $\mathbf{R} \in \mathbb{R}^{m\times m}$ where independently for all i we have $\mathbf{R}_{i,i} = 1/p_i$ with probability p_i and $\mathbf{R}_{i,i} = 0$ otherwise for*

$$p_i \geq \min\left\{1, C_1\frac{m}{\sqrt{n}}\cdot\frac{(\mathbf{G}\mathbf{A}h)_i^2}{\|\mathbf{G}\mathbf{A}h\|_2^2} + C_2\frac{1}{\sqrt{n}} + C_3\overline{\tau}_i\right\}.$$

With high probability the time (and thus also the output size of $\mathbf{R}$) is bounded by $\widetilde{O}((C_1 + C_2)m/\sqrt{n} + C_3 n\log W)$ where W is a bound on the ratio of largest to smallest entry in g.

Note that by Corollary 4.43 the data structure of Lemma F.3 yields the following (P, c, Q)-HeavySampler.

Corollary F.4. *There exists a (P, c, Q)-HeavySampler data structure for edge-vertex incience matrices $\mathbf{A} \in \mathbb{R}^{m\times n}$ with $P = \widetilde{O}(m)$, $c_i = \widetilde{O}(1)$ for all $i \in [m]$, and $Q = \widetilde{O}(m/\sqrt{n} + n\log W)$, where W is the ratio of the largest to smallest non-zero entry in g.*