UNIT CAPACITY MAXFLOW IN ALMOST $m^{4/3}$ TIME*

TARUN KATHURIA †, YANG P. LIU ‡, AND AARON SIDFORD §

Abstract. We present an algorithm which given any m-edge directed graph with positive integer capacities at most U, vertices a and b, and an approximation parameter $\epsilon \in (0, 1)$ computes an additive ϵmU-approximate a-b maximum flow in time $m^{1+o(1)}/\sqrt{\epsilon}$. By applying the algorithm for $\epsilon = (mU)^{-2/3}$, rounding to an integral flow, and using augmenting paths, we obtain an algorithm which computes an exact a-b maximum flow in time $m^{4/3+o(1)}U^{1/3}$ and an algorithm which given an m-edge bipartite graph computes an exact maximum cardinality matching in time $m^{4/3+o(1)}$.

Key words. maximum flow, optimization, interior point methods

AMS subject classifications. 68Q25, 68R10

1. Introduction.

In this paper, we consider the *maxflow* problem of computing a maximum a-b flow for vertices a and b in an m-edge, n-vertex capacitated directed graph with integer capacities at most U. This problem is among the most fundamental problems in combinatorial optimization, and has been the subject of decades of extensive research. It encompasses prominent problems like s-t minimum cut and maximum bipartite matching. In recent years this problem has served as a proving ground for algorithmic advances in optimization [6, 19, 35, 23, 28, 29, 33, 37, 36].

The main result of this paper is a deterministic $m^{4/3+o(1)}U^{1/3}$ time algorithm for solving maxflow. This runtime improves upon the previous best running times of $m^{11/8+o(1)}U^{1/4}$ [27], $\widetilde{O}(m\sqrt{n}\log U)$ [23],[1] and $O(mn)$ [32] when the graph is not too dense and doesn't have too large capacities.[2] To obtain this result, we provide a new perspective on previous interior point method (IPM) based approaches to $o(m^{3/2})$ maxflow runtimes [28, 29, 27] and extend the method to leverage new subroutines. Further, we show how to implement these subroutines efficiently by extending previous iterative refinement based algorithms [21, 2] to solve a larger class of undirected flow problems to high precision in almost linear time.

1.1. Motivation and Significance.

To motivate our result and explain its significance, throughout Subsection 1.1 we restrict our attention to the simplified problem of computing maxflow on unit capacity sparse graphs, i.e. when $U = 1$ and $m = \widetilde{O}(n)$. This problem is equivalent, up to nearly linear time reductions, to the problems of computing a maximum cardinality set of disjoint a-b paths in sparse graph [22] and computing a maximum cardinality matching in a sparse bipartite graph [25].

Obtaining improved running times for this simple problem is notoriously difficult. $\widetilde{O}(\min(m^{3/2}, mn^{2/3}))$ time algorithms for the problem were established in the 1970s [12] and remained the state-of-the-art until the 2010s. Even for the easier problem of computing an ϵ-approximate maxflow, i.e. a feasible flow of value at least $(1 - \epsilon)$ times the optimum, in an undirected graph, no running time improvement was found until the 2010s.[3]

---

*Merged version of accepted submissions to FOCS 2020.

†UC Berkeley, tarunkathuria@berkeley.edu

‡Stanford University, yangpliu@stanford.edu

§Stanford University, sidford@stanford.edu

[1] Some algorithms discussed in the introduction, including the one in [23], are randomized. We do not distinguish randomized versus deterministic algorithms in the introduction.

[2] Here and throughout the paper we use $\widetilde{O}(\cdot)$ to hide poly $\log(n, m, U)$ factors.

[3] Improved runtimes were obtained for dense graphs in the 1990s [15, 16, 18, 17].

Energy Maximization Based Breakthroughs. This longstanding algorithmic efficiency barrier was broken by Christiano, Kelner, Mądry, Spielman and Teng in 2011 [6]. This breakthrough provided an algorithm which computed an ϵ-approximate maxflow in unit-capacity sparse graphs in time $\widetilde{O}(mn^{1/3}\text{poly}(1/\epsilon))$, which is $\widetilde{O}(m^{4/3})$ on sparse graphs with logarithmic ϵ.

To achieve this result, [6] leveraged a seminal result of Spielman and Teng in 2004 [38] that Laplacian systems, class of linear systems associated with undirected graphs, could be solved in nearly linear time. Solving Laplacian systems is equivalent to computing (to high precision) *a-b electric flows*, the flow $f \in \mathbb{R}^E$ that sends one unit of flow and minimizes *energy*, $\mathcal{E}(f) \stackrel{\text{def}}{=} \sum_{e \in E} r_e f_e^2$ for input edge resistances $r \in \mathbb{R}^E_{>0}$. [6] demonstrated that this fact combined with an iterative method, known as the multiplicative weights update (MWU), yields an $\widetilde{O}(m^{3/2}\text{poly}(1/\epsilon))$-time maxflow algorithm. They then introduced a new technique of *energy maximization* to obtain their improved runtime. More precisely, they noted that when MWU converged slowly, a small number of edges could be removed to increase the energy of the electric flows considered and developed methods which could trade-off the loss in approximation quality from edge removal with possible energy increase.

Interestingly, earlier, in 2008, Daitch and Spielman [11] showed that another powerful class of continuous optimization methods, IPMs, reduces the problem of solving maxflow exactly on directed graphs to solving $\widetilde{O}(m^{1/2})$ Laplacian systems. Consequently, [6] created an exciting possibility of combining energy maximization techniques of [6] with IPMs to achieve faster running times for solving maximum flow

In 2013, Mądry [28] provided a breakthrough result which showed that this was possible; this paper provided an $\widetilde{O}(m^{10/7}) = \widetilde{O}(m^{3/2-1/14})$ time algorithm for directed maxflow on sparse unit capacity graphs. This result was an impressive tour de force that involved multiple careful modifications to standard IPM analysis. Leveraging energy maximization techniques is more difficult in IPMs than in MWU, where there is a type of monotonicity that does not occur naturally in IPMs. Additionally, several aspects of IPMs are somewhat brittle and tailored to ℓ_2 and ℓ_4 norms, rather than ℓ_∞ as in maxflow. Consequently, [28] performed multiple modifications to carefully track how both energy and IPM invariants changed during the IPM. This yielded the first $\widetilde{O}(m^{3/2-c})$ time maxflow algorithm for unit capacity graphs for some constant $c > 0$, though one slower than $\widetilde{O}(m^{4/3})$.

Efficient Energy Maximization. Since [6], energy maximization has been applied to a host of other problems [20, 5, 1]. Further [28] has been simplified and improved [29], and its techniques have been extended and applied to related graph problems such as minimum cost flows [10]. Further, the runtime was recently improved by Liu-Sidford to $m^{11/8+o(1)} = m^{3/2-1/8+o(1)}$ [27] and this in turn led to faster algorithms for mincost flow and shortest paths with negative edge lengths [3].

To obtain this runtime improvement, [27] showed that instead of carefully tuning the weights based on the electrical energy, one can consider the separate problem of finding a new set of weights under a budget constraint to maximize the energy. They showed that a version of this problem reduces to solving ℓ_2-ℓ_p norm flow problems, and leveraged recent results of [21, 2] to solve such problems in almost-linear time to achieve their runtime. Although [27, 3] address IPM energy monotonicity issues in novel ways, they do not run in time $m^{4/3}$ due to issues of maintaining IPM invariants and working with ℓ_4, rather than ℓ_∞ (see Section 3).[4]

[4]Technically, in [27] and [3], weight changes are computed to reduce the ℓ_∞ norm of congestion

Interestingly, there are IPMs for linear programming which only measure centrality in ℓ_∞ norm as opposed to the ℓ_2 or ℓ_4 norm. In particular [9, 24, 41, 40] show how to take a step with respect to a softmax function of the duality gap and trace the central path only maintaining ℓ_∞ norm bounds. However, it is unclear how to leverage these for faster maxflow algorithms and this paper takes a different approach.

Our Contributions: Beyond Energy Maximization. Building on the past decade of maxflow research we obtain an $m^{4/3+o(1)}$ maxflow algorithm on sparse, unit-capacity graphs. This closes the gap between the runtime achieved for approximately solving maxflow using electrical flows and the best runtime known for solving maxflow on directed graphs.[5]

We also shed light on the energy maximization frameworks that underlie previous $m^{3/2-\Omega(1)}$ exact maxflow algorithms and depart from it to achieve our bounds.[6] We show that electric energy arises naturally when locally optimizing the second order Taylor approximation of a Bregman divergence of the standard logarithmic barrier arising in IPMs. We then show that by optimizing the entire function, instead of its second order Taylor approximation, we can obtain improved convergence.

Further, we show that this *divergence maximization* can be performed efficiently for graphs. Whereas [27] showed that energy maximization could be performed efficiently by solving smoothed ℓ_2-ℓ_p flows of [21, 2], here we need to solve problems not immediately representable this way. We show previous solvers can be applied to solve a quadratic extension of the divergence maximization problem in Lemma 6.1, which suffices for our algorithms. More generally, we show in Theorem 6.3 that a range of undirected flow optimization problems, including divergence maximization, can be solved efficiently using iterative refinement and smoothed ℓ_2-ℓ_p flows [21].

1.2. Our Results.

The main result of this paper is the following theorem.

THEOREM 1.1 (Maximum Flow). *There is an algorithm which for any $\epsilon > 0$ computes additive ϵmU-approximate solutions to the maxflow problem in m-edge, integer capacitated graphs with maximum capacity at most U, in time $m^{1+o(1)}/\sqrt{\epsilon}$.*

By choosing $\epsilon = (mU)^{-2/3}$, rounding to an integral flow [8], and running augmenting paths on the residual graph we obtain the following result.

THEOREM 1.2. *There is an algorithm which solves maxflow in m-edge, integer capacitated graphs with maximum capacity at most U in $m^{4/3+o(1)}U^{1/3}$ time.*

Theorem 1.2 yields an exact maxflow runtime matching the $\widetilde{O}(mn^{1/3}\epsilon^{-11/3})$ runtime of [6] for $(1-\epsilon)$-approximate undirected maxflow on sparse graphs. Further, this improves on the recent $m^{11/8+o(1)}U^{1/4}$ time algorithm of Liu-Sidford [27] as long as $U \le m^{1/2-\delta}$ for some $\delta > 0$. When $U \ge m^{1/2}$, the result of Theorem 1.1 and all the algorithms of [28, 29, 27] have runtime $\widetilde{O}(m^{3/2})$, which is already known through [14]. Hence, we assume $U \le \sqrt{m}$ throughout the paper.

An immediate corollary of Theorem 1.2 is the following result on efficiently computing bipartite matchings. It improves over the previous bounds of $m^{11/8+o(1)}$ [27] and $\widetilde{O}(m + n^{3/2})$ [40] on sparse graphs.

of an electric flow vector. However, centrality depends on the ℓ_4 norm yielding slower than a $m^{4/3}$ runtimes. Since the initial version of this paper [26] was released, [3] was updated to leverage the techniques of this paper and achieved a $m^{4/3+o(1)}$ runtime for a broader range of problems.

[5] Since [6], faster runtimes for approximate maxflow on undirected graphs have been achieved [22, 35, 19, 33, 36, 37] and the problem is now solvable in nearly linear time.

[6] Independently, [3] gives a perspective on energy maximization as regularized Newton steps.

Corollary 1.3 (Bipartite Matching). *There is an algorithm which given an m-edge bipartite graph computes a maximum cardinality matching in time $m^{4/3+o(1)}$.*

Note that Theorem 1.1, 1.2, and Corollary 1.3 are deterministic. This follows from [7], which showed that algorithms for many flow algorithms, including Laplacian system solvers [38], smoothed ℓ_2-ℓ_p flow algorithms [21, 2], and some maxflow IPMs [28, 29], may be derandomized with a $m^{o(1)}$ multiplicative runtime increase.

1.3. Previous Work. Here, for brevity, we cover several lines of research closely related to our work. See Section 1.3 of [27] for further references.

Approximate undirected maxflow. An extensive line of work leveraging continuous optimization techniques to obtain faster maxflow algorithms stemmed from [6]. [22] also presented a $O(n^{1/3}\mathsf{poly}(1/\varepsilon))$ iteration algorithm for unit-capacity graphs also using electrical flows. Further, [19] and [35] presented algorithms for maxflow achieving runtimes of $O(m^{1+o(1)}\mathsf{poly}(1/\varepsilon))$, [33] improved this to nearly linear time, and [36, 37] further improved the ϵ dependence, and runtime in dense instances.

IPMs for maxflow. In order to obtain highly accurate solutions and improved runtimes for directed maxflow, recent work has leveraged IPMs for linear programming [31, 34]. As discussed, classic IPMs [11] and nearly linear time Laplacian systems solvers [38] directly yield an $\widetilde{O}(m^{3/2}\log(U))$-time maxflow algorithm. In the case of bounded U, this was improved by the sequence of works [28, 29, 27]. Beyond these results, Lee and Sidford [23] also devised a faster IPM using weighted barriers to achieve a $\widetilde{O}(m\sqrt{n}\log(U))$-runtime for maxflow. Further, [10] achieved a runtime for minimum cost flow matching the runtimes of [28, 29] and this was recently improved by [3], leveraging the techniques of this paper. Recently, [40, 39] also provided IPM-based algorithms which yield $\widetilde{O}(m+n^{3/2})$ runtimes for mincost flows. [13] also provided a $\widetilde{O}(m^{3/2-1/328})$ time algorithm for maxflow on polynomially capacitated graphs.

ℓ_p-flows and beyond. Essential to the results of this paper and [27] is recent work on obtaining high-precision solvers for ℓ_p flow problems on graphs, i.e. sending a specified amount of flow while minimizing a weighted ℓ_p-norm. These problems interpolate between electrical flow problems $p=2$, undirected maxflow problems $p=\infty$, and transshipment problems $p=1$. The first improvement over the $\sqrt{m}$ iteration bound given by IPM theory was by [4]. [1] introduced *iterative refinement* which reduces solving the problem to approximately solving smoothed ℓ_2-ℓ_p flows, i.e. combinations of ℓ_2 and ℓ_p norm pieces. This gave an $O_p(m^{1+\frac{p-2}{3p-2}}\log^2(1/\epsilon))$ time algorithm, where $O_p(\cdot)$ hides constants in p. The p-dependence has been improved signficantly by [2]. Further, [21] showed that smoothed ℓ_2-ℓ_p flows could be solved for unit ℓ_p norm weights and $p\in[\omega(1),o(\log n)]$ in $m^{1+o(1)}$ time, and we apply this result to solve our divergence maximization problem.

1.4. Paper Organization. In Section 2 we cover preliminaries. In Section 3 we give a high level overview of our algorithm, and in Section 4 we describe various pieces in greater detail. We prove that our algorithm runs in $m^{1/3+o(1)}$ iterations in Section 5, and present the final runtime bound in Section 6.

Missing proofs are deferred to Appendix A, and necessary lemmas for iterative refinement of our objectives are given in Appendix B. In Appendix C we give additional convex optimization preliminaries and in Appendix D we prove Theorem 6.3, which shows that a large class of flow problems on graphs may be efficiently solved by reduction to smoothed ℓ_2-ℓ_p flows.

2. Preliminaries.

General notation. We let $\mathbb{R}^m_{\geq\alpha}$ denote the set of m-dimensional real vectors which are entrywise at least α. For $v \in \mathbb{R}^m$ and real $p \geq 1$ we define $\|v\|_p$, the ℓ_p-norm of v, as $\|v\|_p \stackrel{\text{def}}{=} \left(\sum_{i=1}^m |v_i|^p\right)^{1/p}$, and $\|v\|_\infty \stackrel{\text{def}}{=} \max_{i=1}^m |v_i|$. For symmetric positive semidefinite (PSD) matrices $M_1, M_2 \in \mathbb{R}^{n\times n}$ we write $M_1 \approx_r M_2$ for $r \geq 1$ if $r^{-1}x^T M_1 x \leq x^T M_2 x \leq r x^T M_1 x$ for all $x \in \mathbb{R}^n$. For differentiable $f : \mathbb{R}^n \to \mathbb{R}$ we define its induced *Bregman divergence* as $D_f(x\|y) \stackrel{\text{def}}{=} f(x) - f(y) - \nabla f(y)^T(x-y)$ for all $x, y \in \mathbb{R}^n$.

Graphs. Throughout this paper, in the graph problems we consider, we suppose that there are both upper and lower capacity bounds on all edges. We let G be a graph with vertex set V, edge set E, and upper and lower capacities $u_e^+ \geq 0$ and $u_e^- \geq 0$ respectively on edge e. We use U to denote the maximum capacity of any edge, so that $\max\{u_e^+, u_e^-\} \leq U$ for all edges e. We let n denote the number of vertices $|V|$, and let m denote the number of edges $|E|$. Further we view undirected graphs as directed graphs with $u_e^+ = u_e^-$ by arbitrarily orienting its edges.

The Maximum Flow Problem. Given a graph $G = (V, E)$ we call any assignment of real values to the edges of E, i.e. $f \in \mathbb{R}^E$, a *flow*. For a flow $f \in \mathbb{R}^E$, we view f_e as the amount of flow on edge e. If $f_e > 0$ we interpret this as sending f_e units in the direction of the edge orientation and if $f_e < 0$ we interpret this as sending $|f_e|$ units in the direction opposite the edge orientation.

In this paper we consider ab-flows, where $a \in V$ is called the source, and $b \in V$ is called the sink. An ab-flow is a flow which routes t units of flow from a to b for some real number $t \geq 0$. Define the unit demand vector $\chi_{ab} \stackrel{\text{def}}{=} 1_b - 1_a$, a vector with a 1 in position b and -1 in position a. When a and b are implicit, we write $\chi = \chi_{ab}$. In this way, we also refer to an ab-flow which routes t units from a to b as a $t\chi$-flow. The *incidence matrix* for a graph G is the matrix $B \in \mathbb{R}^{E\times V}$, where the row corresponding to edge $e = (u, v)$ has a 1 (respectively -1) in the column corresponding to v (respectively u), i.e. is χ_{uv}. Note that $f \in \mathbb{R}^E$ is a $t\chi$-flow if and only if $B^T f = t\chi$. More broadly, we call any $d \in \mathbb{R}^V$ a demand vector if $1^T d = 0$ and we say $f \in \mathbb{R}^E$ routes d if $B^T f = d$.

We say that a $t\chi$-flow f is *feasible* in G if $-u_e^- \leq f_e \leq u_e^+$ for all $e \in E$, so that f satisfies the capacity constraints. We define the *maximum flow problem (maxflow)* as the problem of given a graph G with upper and lower capacities u^+ and u^-, and source and sink vertices a and b, to compute a maximum feasible ab-flow. We denote the maximum value as t^*. For a real number $t \leq t^*$, we let $F_t \stackrel{\text{def}}{=} t^* - t$ denote the remaining amount of flow to be routed.

3. Algorithm Derivation and Motivation.

In this section we present a principled approach for deriving our new method and the previous energy-based methods [28, 29, 27] for maxflow from an IPM setup.

3.1. Interior Point Method Setup.

The starting point for our method is the broad IPM framework for maxflow of [27], which in turn was broadly inspired by [29]. We consider the setup described in Section 2 and design algorithms that maintain a flow $f \in \mathbb{R}^E$, a parameter $t \geq 0$, and weights $w^+, w^- \in \mathbb{R}^m_{\geq 1}$ such that $B^T f = t\chi$ and f is a high accuracy approximation to $f^*_{t,w}$ defined as

$$f^*_{t,w} \stackrel{\text{def}}{=} \operatorname*{argmin}_{B^T f = t\chi} V(f) \text{ for } V(f) \stackrel{\text{def}}{=} -\sum_{e\in E}\left(w_e^+ \log(u_e^+ - f_e) + w_e^- \log(u_e^- + f_e)\right). \tag{3.1}$$

$V(f)$ is known as the *weighted logarithmic barrier* and penalizes how close f is to breaking the capacity constraints and t is the amount of flow $f^*_{t,w}$ sends from a to b.

Broadly, IPMs proceed towards optimal solutions by increasing the parameter t to iteratively improving the quality, and decreasing the proximity to the constraints by decreasing $V(f)$. Previous maxflow IPMs [23, 28, 29, 27] all follow this template. Specifically, [29, 27] alternate between Newton steps to improve the optimality of f for (3.1) (called *centering steps*) and computing a new flow and weights to approximately solve (3.1) for a larger value of t (called *progress steps*). Applying such an approach, while using Laplacian system solvers to implement the steps in nearly linear time, coupled with a preliminary graph preconditioning step (Subsection 4.1) directly yields an $\widetilde{O}(m^{3/2})$ time algorithm. Recent advances [23, 28, 29, 27] were achieved by additional modifications to the weights and flows used.

3.2. Progress steps via divergence minimization. To understand (and improve upon) previous maxflow IPMs, here we explain how to view progress steps in this framework as computing a divergence minimizing $\delta\chi$-flow for some $\delta > 0$. Note that, without weight changes, the cumulative result of a progress and centering step is essentially moving from $f^*_{t,w}$ to $f^*_{t+\delta,w}$ for a step size δ. The optimality conditions of (3.1) give that the gradient of V at the optimum $f^*_{t,w}$ of (3.1) is perpendicular to the kernel of B^T, so there is a vector y with $By = \nabla V(f^*_{t,w})$. Define

$$\widehat{f} \stackrel{\text{def}}{=} \operatorname*{argmin}_{B^T f = \delta\chi} D_V(f^*_{t,w} + f \| f^*_{t,w}) = \operatorname*{argmin}_{B^T f = \delta\chi} V(f^*_{t,w} + f) - V(f^*_{t,w}) - \nabla V(f^*_{t,w})^T f, \tag{3.2}$$

i.e. the $\delta\chi$-flow with smallest divergence from $f^*_{t,w}$ against the barrier V. Again, optimality conditions yield that there is a vector z with

$$Bz = \nabla D_V(f^*_{t,w} + \widehat{f} \| f^*_{t,w}) = \nabla V(f^*_{t,w} + \widehat{f}) - \nabla V(f^*_{t,w}) \ .$$

Therefore, $B(y+z) = \nabla V(f^*_{t,w} + \widehat{f})$. Since $f^*_{t,w} + \widehat{f}$ is a $(t+\delta)\chi$-flow, we must have $f^*_{t,w} + \widehat{f} = f^*_{t+\delta,w}$ by optimality conditions, so that adding $\widehat{f}$ to an optimal point $f^*_{t,w}$ lands us at the next point $f^*_{t+\delta,w}$.

Now, a standard progress step in this framework may be computed by taking a Newton step, i.e. minimizing the second order Taylor approximation of the divergence. The second order Taylor expansion of $D_V(f^*_{t,w} + f \| f^*_{t,w})$ is $\frac{1}{2} f^T \nabla^2 V(f^*_{t,w}) f$, and the resulting step is

$$\operatorname*{argmin}_{B^T f = \delta\chi} \frac{1}{2} f^T \nabla^2 V(f^*_{t,w}) f = \delta \nabla^2 V(f^*_{t,w})^{-1} B (B^T \nabla^2 V(f^*_{t,w})^{-1} B)^\dagger \chi \ . \tag{3.3}$$

This can be computed in $O(m)$ time plus the time to solve a Laplacian system, i.e. $\widetilde{O}(m)$ [38]. Choosing δ that routes $\Omega(m^{-1/2})$ fraction of the remaining flow, adding the flow in (3.3) to our current point, and taking further Newton steps to re-center yields an $\widetilde{O}(m^{3/2})$ time maxflow algorithm.

3.3. Energy-based improvements. [28, 29, 27] improve over the $\widetilde{O}(m^{3/2})$ time algorithm by more carefully analyzing the largest possible step size δ of the Newton step such that recentering may still be performed in $\widetilde{O}(m)$ time, and by leveraging that the flow in (3.3) is an electric flow. Precisely, the size of the step we may take is governed by the *congestion* of the flow we wish to add, which is defined edge-wise as the ratio of flow on an edge to its residual capacity (see c_e^+, c_e^- in Subsection 4.1). In this way, the ℓ_∞ norm of congestion of the χ-electric flow governs the amount of flow we may add before violating capacity constraints. On the other hand, because the

χ-electric flow was a minimizer to a second order approximation of the divergence, the ℓ_4 norm of congestion of the χ-electric flow instead governs the amount of flow we may add so that centering can still be performed in $\widetilde{O}(m)$ time, whereas a bound on the ℓ_2 norm of congestion suffices to achieve the $\widetilde{O}(m^{3/2})$ time algorithm.

In this way, it is natural to attempt to compute weight changes that reduce the ℓ_4 norm of congestion induced by the χ-electric flow. Mądry [28, 29] achieves this by increasing weights of edges with high congestion in the χ-electric flow and trading off against a potential function that is the energy of the χ-electric flow with resistances induced by the Hessian of the weighted logarithmic barrier at the current point.

To improve on the algorithm of [29], [27] instead views increasing energy via budgeted weight change as its own optimization problem. Precisely, the optimization problem was to maximize the energy of an electric flow in a graph G that originally had resistances r under a resistance increase budget. Written algebraically, for a weight budget W, this is

$$\max_{\|r'\|_1 \le W} \min_{B^T f = \delta \chi} \sum_{e \in E} (r_e + r'_e) f_e^2. \tag{3.4}$$

[27] showed that a smoothed version of this objective was solvable in $m^{1+o(1)}$ time using smoothed ℓ_2-ℓ_p flows [21], and that the combinatorial edge boosting framework of [28, 29] can essentially be viewed as greedily taking gradient steps against the objective in (3.4).

3.4. Our new method: beyond electric energy. As discussed, while the ℓ_∞ norm of congestion governs the amount of flow we may add and still have a feasible flow, the algorithms in [28, 29, 27] all instead control the ℓ_4 norm of congestion. This is done to allow for efficient centering; although ℓ_∞-based steps can be taken without breaking capacity constraints, there is sufficient loss in local optimality that $\widetilde{O}(1)$ centering Newton steps cannot recenter it. This leads to the heart of our improvement – we resolve this discrepancy between the ℓ_∞ and ℓ_4 norm of congestion by directly augmenting via the divergence minimizing flow of (3.2). As a result, it suffices to compute weight changes to minimize the ℓ_∞ norm of congestion of the divergence minimizing flow.

A key challenge in this approach is to compute this divergence minimizing flow, and compute weight changes to reduce the ℓ_∞ norm of its congestion. To approach this, we consider the problem of moving from $f^*_{t,w}$ to $f^*_{t+\delta,w}$ for a step size δ, assuming that the residual capacities induced by $f^*_{t,w}$ and $f^*_{t+\delta,w}$ are within 1.05 multiplicatively. This implies that $\nabla^2 V(f^*_{t,w}) \approx_{1.2} \nabla^2 V(f^*_{t+\delta,w})$. To solve this problem, for each piece of the $V(f)$ objective, i.e. $(w_e^+ \log(u_e^+ - f_e) + w_e^- \log(u_e^- + f_e))$, we replace it with a *quadratic extension*, a function that agrees with it on some interval, and extends quadratically outside. Our new objective will have a stable Hessian everywhere, hence can be minimized by Newton's method. By construction, the optimum of the quadratically extended problem and original are the same using convexity (see Observation 1). Further details are provided in Subsection 4.2.

Finally, we wish to compute weights that reduce the ℓ_∞ norm of congestion of the divergence minimizing flow. As the approach of [27] computes weight changes to maximize the electric energy, we instead compute weight changes to maximize the divergence of the divergence minimizing flow. Doing this requires extending the analysis of [29] and energy maximization framework of [27] to nonlinear objectives, such as the quadratic extensions described above, and then generalizing the iterative refinement framework introduced by [1, 21] to a large family of new objectives. We hope that both this unified view of energy and divergence maximization as well as the methods we give for performing this optimization efficiently may have further utility.

4. Technical Ingredients. In this section, we elaborate on several technical aspects discussed in Section 3. We give details for setting up the IPM in Subsection 4.1, discuss preconditioning in Subsection 4.1, elaborate on quadratic extensions in Subsection 4.2, and discuss iterative refinement in Subsection 4.3.

4.1. IPM Details and Preconditioning. In this section, we give a detailed description of our IPM setup. One can reduce directed maxflow to undirected maxflow with linear time overhead and only $O(1)$ capacity increase (see [25, 28] or [27] Section B.4) and consequently, we assume our graph is undirected, so that $u_e^+ = u_e^-$.

Assuming that there is a feasible $t\chi$-flow, optimality conditions of (3.1) give that the gradient of V at the optimum $f_{t,w}^*$ of (3.1) is perpendicular to the kernel of B^T, i.e. there is a dual vector $y \in \mathbb{R}^V$ such that $By = \nabla V(f_{t,w}^*)$. Consequently, for parameter t and weight vectors w^+, w^- we say that a flow f is on the *weighted central path* if and only if there exists a dual vector $y \in \mathbb{R}^V$ such that

$$B^T f = t\chi \quad \text{and} \quad [By]_e = [\nabla V(f)]_e = \frac{w_e^+}{u_e^+ - f_e} - \frac{w_e^-}{u_e^- + f_e} \text{ for all } e \in E \tag{4.1}$$

For simplicity, we write $w = (w^+, w^-) \in \mathbb{R}^{2E}_{\geq 1}$, where we define $\mathbb{R}^{2E}_{\geq \alpha} \stackrel{\text{def}}{=} \mathbb{R}^{E}_{\geq \alpha} \times \mathbb{R}^{E}_{\geq \alpha}$. We define *residual capacities* $c_e^+ \stackrel{\text{def}}{=} u_e^+ - f_e, c_e^- \stackrel{\text{def}}{=} u_e^- + f_e$ and $c_e = \min(c_e^+, c_e^-)$. Note $c_e \geq 0$ for all $e \in E$ if and only if f is feasible.

We initialize $w_e^+ = w_e^- = 1$, $t = 0$, and $f = 0$, which is central. Previous IPM based algorithms for maxflow [28, 29, 27] alternated between progress and centering steps. Progress steps increase the path parameter t at the cost of centrality, which is the distance of f from satisfying (4.1) in the inverse norm of the Hessian of $V(f)$ – see [27] Definition 4.1. Centering steps improve the centrality of the current point without increasing the path parameter t. Our algorithm more directly advances along the central path – given parameter t, weights w, and central path point $f_{t,w}^*$ we compute new weights w^{new}, advance the path parameter to $t + \delta$, and compute $f_{t+\delta, w^{\text{new}}}^*$.

The IPM will reduce the value of the residual flow $F_t = t^* - t$ below a threshold, at which point we may round and use augmenting paths [8]. We assume that the algorithm knows F_t throughout, as our algorithm succeeds with any underestimate of the optimal flow value t^*, and we can binary search for the optimal flow value.

Preconditioning. To precondition our undirected graph G, we add m undirected edges of capacity $2U$ between source a and sink b. This increases the maximum flow value by $2mU$. Throughout the remainder of the paper, we say that the graph G is preconditioned if it is undirected and we have added these edges. Intuitively, preconditioning guarantees that a constant fraction of the remaining flow in the residual graph may be routed in its undirectification, i.e. G with capacities c_e. We use the following lemma from [27]

Lemma 4.1. *[[27], Section B.5.] Consider a preconditioned graph G. For parameter t and weights w let c_e be the residual capacities induced by the flow $f_{t,w}^*$. Then for every preconditioning edge e we have that $c_e \geq \frac{F_t}{7\|w\|_1}$. If $\|w\|_1 \leq 3m$ then $c_e \geq \frac{F_t}{21m}$.*

At the start of the algorithm, as we initialized $w^+ = w^- = 1$, we have $\|w\|_1 = 2m$. To apply Lemma 4.1 we maintain the stronger invariant that $\|w\|_1 \leq 5m/2$ before each step, but may temporarily increase to $\|w\|_1 \leq 3m$ during the step.

4.2. Advancing along the central path via quadratic smoothing. Let t be a path parameter, and let δ be a step size. Let c_e^+, c_e^- be the residual capacities induced by $f^*_{t,w}$, and let $(c_e^+)', (c_e^-)'$ be those induced by $f^*_{t+\delta,w}$. We sketch an algorithm that computes $f^*_{t+\delta,w}$ to high accuracy from $f^*_{t,w}$ in $\widetilde{O}(m)$ time under the assumption $c_e^+ \approx_{1.05} (c_e^+)'$ and $c_e^- \approx_{1.05} (c_e^-)'$ for all $e \in E$. Let $\widehat{f} = f^*_{t+\delta,w} - f^*_{t,w}$ and define the change in the value of the barrier V when we add f as

$$\begin{aligned}\mathcal{B}(f) &\stackrel{\text{def}}{=} V(f + f^*_{t,w}) - V(f^*_{t,w}) \\ &= -\sum_{e \in E} \left(w_e^+ \log \left(1 - \frac{f_e}{u_e^+ - [f^*_{t,w}]_e} \right) + w_e^- \log \left(1 + \frac{f_e}{u_e^- + [f^*_{t,w}]_e} \right) \right), \end{aligned} \tag{4.2}$$

so that $\widehat{f} = \operatorname{argmin}_{B^T f = \delta\chi} \mathcal{B}(f)$. To compute $\widehat{f}$, we smooth (4.2) by replacing each instance of $\log(\cdot)$ with a function $\widetilde{\log}(\cdot)$ defined as

$$\widetilde{\log}_\eta(1+x) \stackrel{\text{def}}{=} \begin{cases} \log(1+x) & \text{for } |x| \le \eta \\ \log(1+\eta) + \log'(1+\eta)(x-\eta) + \frac{\log''(1+\eta)}{2}(x-\eta)^2 & \text{for } x \ge \eta \\ \log(1-\eta) + \log'(1-\eta)(x+\eta) + \frac{\log''(1-\eta)}{2}(x+\eta)^2 & \text{for } x \le -\eta. \end{cases}$$

Here, we fix $\eta = 1/10$ and write $\widetilde{\log}(1+x) \stackrel{\text{def}}{=} \widetilde{\log}_{1/10}(1+x)$. Note that $\widetilde{\log}_\eta(1+x)$ is the natural quadratic extension of $\log(1+x)$ outside the interval $|x| \le \eta$. Specifically, the functions agree for $|x| \le \eta$, and we $\widetilde{\log}_\eta(1+x)$ is the second order Taylor expansion of $\log(1+x)$ at $\eta, -\eta$ for $x > \eta$, $x < -\eta$ respectively. In this way, $\widetilde{\log}(1+x)$ is twice differentiable everywhere. Define

$$\widetilde{\mathcal{B}}(f) \stackrel{\text{def}}{=} -\sum_{e \in E} \left(w_e^+ \widetilde{\log} \left(1 - \frac{f_e}{u_e^+ - [f^*_{t,w}]_e} \right) + w_e^- \widetilde{\log} \left(1 + \frac{f_e}{u_e^- + [f^*_{t,w}]_e} \right) \right).$$

We now claim that $\widehat{f} = \operatorname{argmin}_{B^T f = \delta\chi} \widetilde{\mathcal{B}}(f)$, and that it can be computed in $\widetilde{O}(m)$ time. To argue the latter, note that by construction, all Hessians $\nabla^2 \widetilde{\mathcal{B}}(f)$ are within a multiplicative factor of 2 of each other, hence we can compute $\operatorname{argmin}_{B^T f = \delta\chi} \widetilde{\mathcal{B}}(f)$ in $\widetilde{O}(m)$ time using Newton's method and electric flow computations. Because $c_e^+ \approx_{1.05} (c_e^+)'$ and $c_e^- \approx_{1.05} (c_e^-)'$, we know that $\mathcal{B}$ and $\widetilde{\mathcal{B}}$ agree in a neighborhood of $\widehat{f}$, so $\widehat{f} = \operatorname{argmin}_{B^T f = \delta\chi} \widetilde{\mathcal{B}}(f)$ by the following simple observation, which is similar to aspects of [4]. For completeness, we provide a proof in Appendix A.1.

Observation 1. *Let $S \subseteq \mathbb{R}^n$ be a convex set, and let $f, g : \mathbb{R}^n \to \mathbb{R}$ be convex functions. Let $x^* = \operatorname{argmin}_{x \in S} f(x)$, and assume that f, g agree on a neighborhood of x^* in $\mathbb{R}^n$. Then $g(x^*) = \min_{x \in S} g(x)$.*

We mention that we do not directly perform the procedure described here, and instead apply quadratic smoothing in a different form in Section 5 for convenience. There we smooth the function $D \stackrel{\text{def}}{=} D_{-\log(1-x)}(x \| 0)$, the Bregman divergence of x to 0 with respect to the function $-\log(1-x)$, instead of directly smoothing $\log(1+x)$. The smoothed function $\widetilde{D}$ is shown in (5.1), (5.2).

4.3. Iterative refinement. The idea of *iterative refinement* was introduced in [1, 21] to solve p-norm regression problems, such as $\min_{B^T f = d} \|f\|_p^p$. Iterative refinement solves such an objective by reducing it to approximately minimizing objectives which are combinations of quadratic and ℓ_p norm pieces. Specifically, Lemma B.2 shows that for a fixed flow f there is a gradient vector g such that for any circulation Δ we have

$$\|f+\Delta\|_p^p - \|f\|_p^p = g^T\Delta + \Theta_p\left(\sum_{e\in E}|f_e|^{p-2}\Delta_e^2 + \|\Delta\|_p^p\right), \tag{4.3}$$

so that approximately minimizing (4.3) over circulations Δ suffices to make multiplicative progress towards the optimum of $\min_{B^T f=d}\|f\|_p^p$. We show in Appendix B that a general class of objectives may all be reduced to approximately minimizing objectives which are combinations of quadratic and ℓ_p norm pieces. Specifically, any convex function h with stable second derivative admits an expansion for $h(x+\Delta)^p - h(x)^p$ similar to (4.3), which we show in Lemma B.3. We then combine this expansion with the almost linear time smoothed ℓ_2-ℓ_p solver of [21] to show Theorem 6.3.

5. Iteration Complexity Analysis. In this section we present our divergence maximization algorithm and prove the iteration bound of $\widetilde{O}(1/\sqrt{\epsilon})$ required to establish Theorem 1.1. In the following Section 6, we show how to implement each iteration efficiently in almost linear time to achieve the desired runtime for Theorem 1.1.

The section is organized as follows. After introducing the algorithm's invariants and notational conventions, we define the divergence and smoothed divergence $D, \widetilde{D}$, and the regularized objectives $\mathrm{val}(f), \widetilde{\mathrm{val}}(f)$ induced by the divergence maximization problem (see (5.7)). Our algorithm for taking a single step is given in Algorithm 1. Given this, we bound the optimal value of the objective $\widetilde{\mathrm{val}}(f)$ in Lemma 5.3. The following Lemmas 5.4 to 5.6 bound various quantities relating to the amount of flow on an edge, its congestion, and the total initial weight increase. Finally, Lemma 5.7 combines the previous pieces to show to formally show that Algorithm 1 goes from one central path path to the next exact central path point, and bounds the total weight increase from a reduced set of weights. The main pieces we require in Section 6 are the pseudocode of Algorithm 1 and the guarantees of Lemma 5.7.

Algorithm invariants. We assume G is preconditioned (Subsection 4.1) and we maintain the invariant $\|w\|_1 \le 5m/2$ before each step, and $\|w\|_1 \le 3m$ at all times. We assume that our algorithm knows F_t, as discussed in Subsection 4.1. For $p = 2\lceil\sqrt{\log m}\rceil$, we assume that that $F_t \ge 10\epsilon m^{1+2/p}U$, since we wish to compute an ϵmU additive approximate flow. Further, we assume, without loss of generality, that $\epsilon \le m^{-2/p}$.

Notational conventions. We largely adopt the same notation as used in Subsection 4.2. We use D to refer to functions which are a Bregman divergence, and for a function h, we use $\widetilde{h}$ to denote a quadratic extension of h. For flows we use f and $\widehat{f}$, the latter which refers to flows we wish to augment by. We use Δ and $\widehat{\Delta}$ for circulations. The letters w, μ, ν refer to weights and weight changes, and W refers to a weight budget.

For $\eta < 1$ define the functions

$$D(x) \stackrel{\text{def}}{=} D_{-\log(1-x)}(x\|0) = -\log(1-x) - x \quad \text{and} \tag{5.1}$$

$$\widetilde{D}_\eta(x) \stackrel{\text{def}}{=} \begin{cases} D(x) & \text{for } |x| \le \eta \\ D(\eta) + D'(\eta)(x-\eta) + \frac{D''(\eta)}{2}(x-\eta)^2 & \text{for } x \ge \eta \\ D(-\eta) + D'(-\eta)(x+\eta) + \frac{D''(-\eta)}{2}(x+\eta)^2 & \text{for } x \le -\eta. \end{cases} \tag{5.2}$$

Throughout, we will omit η and write $\widetilde{D}(x) \stackrel{\text{def}}{=} \widetilde{D}_{1/10}(x)$. As discussed in Subsection 4.2 $\widetilde{D}(x)$ is such that it behaves as a quadratic around $x = 0$, and has second derivatives in $[1/2, 2]$ (Lemma 5.1). We have defined D as the Bregman divergence of $-\log(1-x)$ from 0, and $\widetilde{D}$ is the quadratic extension of D as described in Subsection 4.2. Several useful properties of the derivatives and stability of D and $\widetilde{D}$ are collected in Lemma 5.1, which we prove in Appendix B.

Lemma 5.1 (Properties of $\widetilde{D}$). *We have that $1/2 \le \widetilde{D}''(x) \le 2$ for all $x \in \mathbb{R}$. Also, for $x \ge 0$ we have that $x/2 \le \widetilde{D}'(x) \le 2x$ and $-x/2 \ge \widetilde{D}'(-x) \ge -2x$. We have that $x^2/4 \le \widetilde{D}(x) \le x^2$ for all x.*

Now we define the analog of electric energy which we maximize under a weight budget. Below, we assume without loss of generality that $c_e^+ \le c_e^-$ for all edges e, as the orientation of each edge is arbitrary in the algorithm. In this way, $c_e = c_e^+$ for all e.

$$D_w^V(f) \stackrel{\text{def}}{=} \sum_{e \in E} \left(w_e^+ D\left(\frac{f_e}{c_e^+}\right) + w_e^- D\left(-\frac{f_e}{c_e^-}\right) \right) \text{ and} \tag{5.3}$$

$$\widetilde{D}_w^V(f) \stackrel{\text{def}}{=} \sum_{e \in E} \left(w_e^+ \widetilde{D}\left(\frac{f_e}{c_e^+}\right) + w_e^- \widetilde{D}\left(-\frac{f_e}{c_e^-}\right) \right). \tag{5.4}$$

While minimizing $\widetilde{D}_w^V(f)$ is sufficient (by its optimality conditions) to compute a new flow which is central, provided the old flow is central and t is increased by at most a certain amount, if the weights w do not change. However, we change the weights as well to prove better bounds on the congestion of the flow. This creates another complication as our old flow may no longer be central with respect to the new weights. We fix it by enforcing a constraint in our search for the new weights $w + \nu$ such that the old flow is central with respect to these new weights and we show that this can still efficiently be minimized.

Next, in (5.5) and (5.6) we define val and $\widetilde{\text{val}}$, where $p = 2\lceil\sqrt{\log m}\rceil$, and $W = \epsilon^2 m^3 / F_t^2$ is a constant. For clarity, we express the vector inside the $\|\cdot\|_p$ piece of (5.5), (5.6) coordinate-wise, where the coordinate corresponding to edge e is written.

$$\text{val}(f) \stackrel{\text{def}}{=} D_w^V(f) + W \left\| \left(c_e^+\right)^2 \left(D\left(\frac{f_e}{c_e^+}\right) + \left(\frac{c_e^-}{c_e^+}\right) D\left(-\frac{f_e}{c_e^-}\right) \right) \right\|_p \text{ and} \tag{5.5}$$

$$\widetilde{\text{val}}(f) \stackrel{\text{def}}{=} \widetilde{D}_w^V(f) + W \left\| \left(c_e^+\right)^2 \left(\widetilde{D}\left(\frac{f_e}{c_e^+}\right) + \left(\frac{c_e^-}{c_e^+}\right) \widetilde{D}\left(-\frac{f_e}{c_e^-}\right) \right) \right\|_p. \tag{5.6}$$

These are defined so that for q as the dual norm of p, i.e. $1/q + 1/p = 1$, we have that val and $\widetilde{\text{val}}$ correspond to maximizing the minimum values of $D_w^V(f)$ and $\widetilde{D}_w^V(f)$ under a weighted ℓ_q weight budget. Specifically, we can compute using Sion's minimax theorem that

$$\max_{\substack{\|(c_e^+)^{-2}\nu_e^+\|_q \le W \\ \nu\in\mathbb{R}^{2E}_{\ge 0} \\ \frac{\nu_e^+}{c_e^+} = \frac{\nu_e^-}{c_e^-} \text{ for all } e\in E}} \min_{B^T f = d} D^V_{w+\nu}(f) = \min_{B^T f = d} \max_{\substack{\|(c_e^+)^{-2}\nu_e^+\|_q \le W \\ \nu\in\mathbb{R}^{2E}_{\ge 0} \\ \frac{\nu_e^+}{c_e^+} = \frac{\nu_e^-}{c_e^-} \text{ for all } e\in E}} D^V_{w+\nu}(f) \tag{5.7}$$

$$= \min_{B^T f = d} \max_{\substack{\|(c_e^+)^{-2}\nu_e^+\|_q \le W \\ \nu\in\mathbb{R}^{2E}_{\ge 0} \\ \frac{\nu_e^+}{c_e^+} = \frac{\nu_e^-}{c_e^-} \text{ for all } e\in E}} D^V_w(f) + D^V_\nu(f) \tag{5.8}$$

$$= \min_{B^T f = d} \max_{\substack{\|(c_e^+)^{-2}\nu_e^+\|_q \le W \\ \nu^+\in\mathbb{R}^{2E}_{\ge 0}}} D^V_w(f) + \sum_{e\in E}\left(\nu_e^+ D\left(\frac{f_e}{c_e^+}\right) + \frac{c_e^-}{c_e^+}\nu_e^+ D\left(-\frac{f_e}{c_e^-}\right)\right) \tag{5.9}$$

$$= \min_{B^T f = d} \mathrm{val}(f) \tag{5.10}$$

and similarly for $\widetilde{D}^V_w(f)$ and $\widetilde{\mathrm{val}}(f)$. As mentioned above, the objective requires the constraint $\nu_e^+/c_e^+ = \nu_e^-/c_e^-$ for all $e \in E$ to ensure that the weight increase ν maintains centrality of the old flow (and hence guarantees centrality of the new flow), and the coefficient of $(c_e^+)^{-2}$ in the weight budget $\|(c_e^+)^{-2}\nu_e^+\|_q \le W$ is chosen to ensure that the ℓ_p piece in $\widetilde{\mathrm{val}}(f)$ is ensured to have approximately unit weights on the f_e. Precisely, by Lemma 5.1 and $c_e^+ \le c_e^-$ we have

$$(c_e^+)^2\left(\widetilde{D}\left(\frac{f_e}{c_e^+}\right) + \left(\frac{c_e^-}{c_e^+}\right)\widetilde{D}\left(-\frac{f_e}{c_e^-}\right)\right) = \Theta(f_e^2).$$

We require this property to apply the smoothed ℓ_2-ℓ_p flow solvers in Theorem 6.2.

As $\min_{B^T f = d}\mathrm{val}(f)$ is the result of applying Sion's minimax theorem to a saddle point problem, there will be an optimal solution pair (f^*, μ^*). Ultimately, f^* will be the flow which we add to our current flow to arrive at the next central path point, and the weight change will be derived from applying a weight reduction to μ^*.

The remaining arguments in this section heavily use local optimality of convex functions. For this reason, we show that $\mathrm{val}(f)$ and $\widetilde{\mathrm{val}}(f)$ are convex in Appendix A.2.

LEMMA 5.2. $\mathrm{val}(f)$ *and* $\widetilde{\mathrm{val}}(f)$ *are convex.*

From now on, we fix a step size $\delta = \frac{\sqrt{\epsilon}F_t}{10^5}$. This simplifies our analysis, as our objectives are no longer linear in δ, as is the case with electric flows. We now bound the minimum value of $\widetilde{D}^V_w(f)$ over all $\delta\chi$-flows, and show a congestion bound for the minimizer, where we recall that congestion of a flow is the ratio of flow on an edge to its residual capacity c_e. Lemmas 5.3 and 5.4 generalize corresponding bounds for electric flows shown in [29] and [27] Lemma 4.5 and 5.2.

LEMMA 5.3. *Let* $\delta = \frac{\sqrt{\epsilon}F_t}{10^5}$. *Then* $\min_{B^T f = \delta\chi}\widetilde{D}^V_w(f) \le 5\cdot 10^{-7}\epsilon m$.

Proof. Let f' be the flow which routes δ/m units of flow on each of the m preconditioning edges. For a preconditioning edge e_p, Lemma 4.1 and the invariant $\|w\|_1 \le 3m$ tells us that

$$\frac{f'_{e_p}}{c_{e_p}} \le \frac{\delta/m}{F_t/7\|w\|_1} \le \frac{21\sqrt{\epsilon}}{10^5}.$$

Therefore, applying Lemma 5.1 to P (the set of m preconditioning edges added in Lemma 4.1) and again applying that $\|w\|_1 \le 3m$ gives us the desired bound, as

$$\widetilde{D}_w^V(f') = \sum_{e\in P}\left(w_e^+ \widetilde{D}\left(\frac{f'_e}{c_e^+}\right) + w_e^- \widetilde{D}\left(-\frac{f'_e}{c_e^-}\right)\right)$$
$$\le \left(\frac{f'_{e_p}}{c_{e_p}}\right)^2 \sum_{e\in P}\left(w_e^+ + w_e^-\right) \le \left(\frac{21\sqrt{\epsilon}}{10^5}\right)^2 \|w\|_1 \le 5\cdot 10^{-7}\epsilon m \qquad \square$$

The next lemma relates, using optimality conditions of the divergence maximization subproblem, the Bregman divergence to the congestion of our flow and almost gets us to our desired upper bound on the congestion.

LEMMA 5.4. *$|\widehat{f}_e| c_e^{-2} \le \sqrt{\epsilon} m/F_t$ for $\delta = \frac{\sqrt{\epsilon}F_t}{10^5}$ and $\widehat{f} = \operatorname{argmin}_{B^T f = \delta\chi} \widetilde{D}_w^V(f)$.*

Proof. Local optimality implies that exist $z \in \mathbb{R}^V$ satisfying $Bz = \nabla \widetilde{D}_w^V(\widehat{f})$. This, Lemma 5.3, and Lemma 5.1, specifically that $x\widetilde{D}'(x) \le 2x^2 \le 8\widetilde{D}(x)$, yield

$$\delta\chi^T z = \widehat{f}^T Bz = \widehat{f}^T \nabla \widetilde{D}_w^V(\widehat{f}) \tag{5.11}$$
$$= \sum_{e\in E} \widehat{f}_e \left(\frac{w_e^+}{c_e^+}\widetilde{D}'\left(\frac{\widehat{f}_e}{c_e^+}\right) - \frac{w_e^-}{c_e^-}\widetilde{D}'\left(-\frac{\widehat{f}_e}{c_e^-}\right)\right) \le 8\widetilde{D}_w^V(\widehat{f}). \tag{5.12}$$

Note that the flow $\widehat{f}$ is acyclic, i.e. there is no cycle where the flow is in the positive direction for every cycle edge. This follows because decreasing the flow along a cycle reduces the objective value, which is monotone and minimized at 0 for each edge. Also, for all edges $e = (u, v)$, we have $z_v - z_u = [Bz]_e = [\nabla \widetilde{D}_w^V(\widehat{f})]_e$, which has the same sign as $\widehat{f}_e$. As $\widehat{f}$ is acyclic, it can be decomposed into a-b paths. Since, some path contains the edge e, we get that $|[Bz]_e| = |z_v - z_u| \le z_b - z_a = \chi^T z$. Using that $x\widetilde{D}'(x) \ge x^2/2$ from Lemma 5.1 we get that

$$|[Bz]_e| = |[\nabla \widetilde{D}_w^V(\widehat{f})]_e| \ge \frac{w_e^+ |\widehat{f}_e|}{2(c_e^+)^2} + \frac{w_e^- |\widehat{f}_e|}{2(c_e^-)^2} \ge \frac{1}{2}|\widehat{f}_e| c_e^{-2}.$$

Combining these observations with (5.11), (5.12) gives us

$$|\widehat{f}_e| c_e^{-2} \le 2|[Bz]_e| \le 2\chi^T z \le 16\delta^{-1}\widetilde{D}_w^V(\widehat{f}) \le \sqrt{\epsilon} m/F_t$$

after using Lemma 5.3 and $\delta = \frac{\sqrt{\epsilon}F_t}{10^5}$. $\square$

Now, we show that computing $\widehat{f} = \operatorname{argmin}_{B^T f = \delta\chi} \widetilde{\mathrm{val}}(f)$ gives us weight changes to control the congestion of the divergence maximizing flow. For clarity, the process is shown in Algorithm 1.

Now, we analyze Algorithm 1. We first show that $\widehat{f} = \operatorname{argmin}_{B^T f = \delta\chi} \widetilde{D}_{w+\mu}^V(f)$ for the weight change μ in Line 6 of Algorithm 1, and that μ has bounded ℓ_1 norm. This essentially follows from duality in our setup in (5.7).

LEMMA 5.5. *Let parameters W, c_e^+, c_e^-, δ, flow $\widehat{f}$, and weight change μ be defined as in Algorithm 1. Assume that $F_t \ge 100\epsilon m^{1+2/p} U$. Then we have that $\|\mu\|_1 \le m/2$, $f_{t,w}^* = f_{t,w+\mu}^*$, and $\widehat{f} = \operatorname{argmin}_{B^T f = \delta\chi} \widetilde{D}_{w+\mu}^V(f)$.*

Algorithm 1: AUGMENT$(G, w, F_t, f^*_{t,w}, \epsilon)$. Takes a preconditioned undirected graph G with maximum capacity U, weights $w \in \mathbb{R}^{2E}_{\geq 1}$ with $\|w\|_1 \leq 5m/2$, residual flow $F_t = t^* - t$, central path point $f^*_{t,w}$. Returns step size δ, weights ν, and $\delta\chi$-flow $\widehat{f}$ with $f^*_{t,w+\nu} = f^*_{t,w} + \widehat{f}$.

1 $\delta \leftarrow \frac{\sqrt{\epsilon}F_t}{10^5}$ ▷ Step size.
2 $c_e^+ \leftarrow u_e^+ - [f^*_{t,w}]_e, c_e^- \leftarrow u_e^- + [f^*_{t,w}]_e$. ▷ Residual capacities.
3 $W \leftarrow \epsilon^2 m^3/F_t^2$ ▷ Weight budget.
4 $\widehat{f} \leftarrow \text{argmin}_{B^T f = \delta\chi} \widetilde{\text{val}}(f)$ ▷ $\widetilde{\text{val}}(f)$ implicitly depends on W, c_e^+, c_e^-.
5 $v \in \mathbb{R}^E$ defined as $v_e \leftarrow (c_e^+)^2 \left(\widetilde{D}\left(\frac{\widehat{f}_e}{c_e^+}\right) + \left(\frac{c_e^-}{c_e^+}\right)\widetilde{D}\left(-\frac{\widehat{f}_e}{c_e^-}\right)\right)$ for all $e \in E$.
6 $\mu \in \mathbb{R}^{2E}_{\geq 0}$ defined as $\mu_e^+ \leftarrow W(c_e^+)^2 \cdot \frac{v_e^{p-1}}{\|v\|_p^{p-1}}$ and $\mu_e^- \leftarrow \frac{c_e^-}{c_e^+}\mu_e^+$. ▷ Preliminary weight change.
7 Initialize $\nu \in \mathbb{R}^{2E}_{\geq 0}$. ▷ Reduced weight change, $\frac{\nu_e^+}{c_e^+ - \widehat{f}_e} - \frac{\nu_e^-}{c_e^- + \widehat{f}_e} = \frac{\mu_e^+}{c_e^+ - \widehat{f}_e} - \frac{\mu_e^-}{c_e^- + \widehat{f}_e}$
8 **for** $e \in E$ **do**
9   **if** $\frac{\mu_e^+}{c_e^+ - \widehat{f}_e} - \frac{\mu_e^-}{c_e^- + \widehat{f}_e} \geq 0$ **then**
10         $\nu_e^+ \leftarrow (c_e^+ - \widehat{f}_e)\left(\frac{\mu_e^+}{c_e^+ - \widehat{f}_e} - \frac{\mu_e^-}{c_e^- + \widehat{f}_e}\right), \nu_e^- \leftarrow 0$
11   **else**
12         $\nu_e^+ \leftarrow 0, \nu_e^- \leftarrow -(c_e^- + \widehat{f}_e)\left(\frac{\mu_e^+}{c_e^+ - \widehat{f}_e} - \frac{\mu_e^-}{c_e^- + \widehat{f}_e}\right)$
13 Return $(\delta, \widehat{f}, \nu)$.

Proof. Let $v \in \mathbb{R}^E$ be the vector as defined in Line 5 in Algorithm 1. By local optimality of $\widehat{f}$, we have that there is a vector z satisfying for all $e \in E$ that

$$[Bz]_e = \left[\nabla\widetilde{\text{val}}(\widehat{f})\right]_e$$

(5.13)
$$= \left(\frac{w_e^+}{c_e^+} + \frac{v_e^{p-1}}{\|v\|_p^{p-1}} \cdot \frac{W(c_e^+)^2}{c_e^+}\right)\widetilde{D}'\left(\frac{\widehat{f}_e}{c_e^+}\right) - \left(\frac{w_e^-}{c_e^-} + \frac{v_e^{p-1}}{\|v\|_p^{p-1}} \cdot \frac{Wc_e^+c_e^-}{c_e^-}\right)\widetilde{D}'\left(-\frac{\widehat{f}_e}{c_e^-}\right).$$

For clarity, we rewrite Line 6 of Algorithm 1 here as

$$\mu_e^+ = W(c_e^+)^2 \cdot \frac{v_e^{p-1}}{\|v\|_p^{p-1}} \text{ and } \mu_e^- = \frac{c_e^-}{c_e^+}\mu_e^+ = Wc_e^+c_e^- \cdot \frac{v_e^{p-1}}{\|v\|_p^{p-1}}. \tag{5.14}$$

Note that $\mu_e^+/c_e^+ = \mu_e^-/c_e^-$, hence $f^*_{t,w} = f^*_{t,w+\mu}$ by (4.1). Combining (5.13), (5.14) and optimality conditions of the objective $\min_{B^T f = \delta\chi} \widetilde{D}^V_{w+\mu}(f)$ shows that $\widehat{f} = \text{argmin}_{B^T f = \delta\chi} \widetilde{D}^V_{w+\mu}(f)$. Also, if q is the dual of p, i.e. $1/q + 1/p = 1$, then

$$\|\mu\|_1 \leq m^{1/p}\|\mu\|_q \leq 2m^{1/p}WU^2 \left\|\frac{v_e^{p-1}}{\|v\|_p^{p-1}}\right\|_q = 2m^{1/p}WU^2 = 2m^{1/p}\frac{m(\epsilon mU)^2}{F_t^2} \leq m/2$$

as $F_t \geq 10\epsilon m^{1+2/p}U$ by assumption. □

We now show congestion bounds on $\widehat{f}$ by imitating the proof of Lemma 5.3 and applying Lemma 5.4. Recall that $c_e = \min(c_e^+, c_e^-)$.

Lemma 5.6. *Let parameters W, c_e^+, c_e^-, δ, flow $\widehat{f}$, and weight change μ be defined as in Algorithm 1. Then we have $|\widehat{f}_e| \le \frac{1}{500}\frac{F_t}{\sqrt{\epsilon m}}$ and $|\widehat{f}_e| \le \frac{1}{20}c_e$ for all edges e. It follows that $\widehat{f} = \operatorname{argmin}_{B^T f = \delta\chi} \operatorname{val}(f)$.*

Proof. We first show $\widetilde{\operatorname{val}}(\widehat{f}) \le 10^{-6}\epsilon m$. Let f' be the flow which routes $\frac{\delta}{m}$ units of flow on each of the m preconditioning edges. As in Lemma 5.3 we have that $\widetilde{D}_w^V(f') \le 5 \cdot 10^{-7}\epsilon m$. For a preconditioning edge e, using Lemma 5.1 and Lemma 4.1 gives that

$$(c_e^+)^2 \left(\widetilde{D}\left(\frac{f_e'}{c_e^+}\right) + \left(\frac{c_e^-}{c_e^+}\right)\widetilde{D}\left(-\frac{f_e'}{c_e^-}\right)\right) \le (c_e^+)^2 \left(\left(\frac{f_e'}{c_e^+}\right)^2 + \left(\frac{c_e^-}{c_e^+}\right)\left(\frac{f_e'}{c_e^-}\right)^2\right)$$

$$\le 2(f_e')^2 \le 2(\delta/m)^2 \le 2m^{-2}\left(\frac{\sqrt{\epsilon}F_t}{10^5}\right)^2 \le 10^{-9}\epsilon F_t^2/m^2.$$

For the choice $W = \epsilon^2 m^3/F_t^2$ we get that

$$\widetilde{\operatorname{val}}(f') \le \widetilde{D}_w^V(f') + W\left\|(c_e^+)^2\left(\widetilde{D}\left(\frac{f_e'}{c_e^+}\right) + \left(\frac{c_e^-}{c_e^+}\right)\widetilde{D}\left(-\frac{f_e'}{c_e^-}\right)\right)\right\|_p$$

$$\le 5 \cdot 10^{-7}\epsilon m + 10^{-9} m^{1/p}\frac{\epsilon F_t^2}{m^2} \cdot \frac{\epsilon^2 m^3}{F_t^2}$$

$$\le 5 \cdot 10^{-7}\epsilon m + 10^{-9} m^{1/p}\epsilon^3 m \le 10^{-6}\epsilon m$$

where we have used $\|x\|_p \le m^{1/p}\|x\|_\infty$ for the choice $p = 2\left\lceil\sqrt{\log m}\right\rceil$, and $\epsilon \le m^{-2/p}$ to get $m^{1/p}\epsilon^2 \le 1$.

We now show $|\widehat{f}_e| \le \frac{1}{500}\frac{F_t}{\sqrt{\epsilon m}}$ for all e. Indeed, applying $\widetilde{D}(x) \ge x^2/4$ from Lemma 5.1 yields

$$\frac{1}{4}W\widehat{f}_e^2 \le \widetilde{\operatorname{val}}(f) \le 10^{-6}\epsilon m.$$

Using the choice $W = \epsilon^2 m^3/F_t^2$ and rearranging gives us $|\widehat{f}_e| \le \frac{1}{500}\frac{F_t}{\sqrt{\epsilon m}}$.

Let μ be the weight increases given by Line 6 of Algorithm 1, and (5.14). As $\widehat{f} = \min_{B^T f = \delta\chi} \widetilde{D}_{w+\mu}^V(f)$ and $\|w+\mu\|_1 \le 5m/2 + m/2 \le 3m$ by our invariant and Lemma 5.5, using Lemma 5.4 gives us

$$|\widehat{f}_e|c_e^{-1} = \left(|\widehat{f}_e| \cdot |\widehat{f}_e|c_e^{-2}\right)^{1/2} \le \left(\frac{1}{500}\frac{F_t}{\sqrt{\epsilon m}} \cdot \frac{\sqrt{\epsilon}m}{F_t}\right)^{1/2} \le \frac{1}{20}.$$

Using that the functions $D(x)$ and $\widetilde{D}(x)$ agree for $|x| \le \frac{1}{10}$, and $|\widehat{f}_e| \le \frac{1}{20}c_e$ for all e, Observation 1 gives us that $\widehat{f}$ is also a minimizer of $\min_{B^T f = \delta\chi} \operatorname{val}(f)$ as desired. ∎

We now show that applying weight change μ and adding $\widehat{f}$ to our current central path point $f_{t,w}^*$ stays on the central path, for path parameter $t + \delta$. This follows from optimality conditions on the objective, which we designed to satisfy exactly the desired property.

As the weight change μ may be too large, we reduce the weight change μ to a weight change ν after advancing the path parameter, and bound $\|\nu\|_1$. Intuitively, this weight reduction procedure can never hurt the algorithm. It happens to help because we carefully designed our objective to induce smoothed ℓ_2-ℓ_p flow instances with unit weights on the ℓ_p part, which are the only instances known to admit almost linear runtimes [21]. Later in Section 6, we use this fact to implement each iteration of our algorithm in almost linear $m^{1+o(1)}$ time.[7] The following lemma is crucial to our analysis as it allows us to keep the weight increase bounded which in turn allows us to make significant progress on the central path, while keeping the congestion of the flow small by Lemma 5.6 and Lemma 5.5.

LEMMA 5.7. *Let parameters W, c_e^+, c_e^-, δ, flow $\widehat{f}$, and weight changes μ, ν be defined as in Algorithm 1, and assume that $F_t \geq 10\epsilon m^{1+2/p} U$. Then we have that $\|\nu\|_1 \leq \sqrt{\epsilon} m^{1-1/p}$ and $f^*_{t+\delta, w+\nu} = f^*_{t,w} + \widehat{f}$.*

Proof. We first show $f^*_{t+\delta,w+\mu} = f^*_{t,w+\mu} + \widehat{f} = f^*_{t,w} + \widehat{f}$. By Lemma 5.6, we have $\widehat{f} = \mathrm{argmin}_{B^T f = \delta\chi} \mathrm{val}(f)$. Let the vector v be defined as in Line 5 of Algorithm 1. There exist vectors $y, z \in \mathbb{R}^E$ such that

$$
\begin{aligned}
[Bz]_e &= \left[\nabla \mathrm{val}(\widehat{f})\right]_e \\
&= \left(\frac{w_e^+}{c_e^+} + \frac{v_e^{p-1}}{\|v\|_p^{p-1}} \cdot \frac{W(c_e^+)^2}{c_e^+}\right) D'\left(\frac{\widehat{f}_e}{c_e^+}\right) - \left(\frac{w_e^-}{c_e^-} + \frac{v_e^{p-1}}{\|v\|_p^{p-1}} \cdot \frac{W c_e^+ c_e^-}{c_e^-}\right) D'\left(-\frac{\widehat{f}_e}{c_e^-}\right) \\
&= \left[\frac{w_e^+ + \mu_e^+}{c_e^+ - \widehat{f}_e} - \frac{w_e^+ + \mu_e^+}{c_e^+}\right] - \left[\frac{w_e^- + \mu_e^-}{c_e^- + \widehat{f}_e} - \frac{w_e^- + \mu_e^-}{c_e^-}\right] \\
&= \left[\frac{w_e^+ + \mu_e^+}{u_e^+ - \left[f^*_{t,w+\mu}\right]_e - \widehat{f}_e} - \frac{w_e^- + \mu_e^-}{u_e^- + \left[f^*_{t,w+\mu}\right]_e + \widehat{f}_e}\right] - [By]_e.
\end{aligned}
$$

Here, the first line follows from local optimality of $\widehat{f} = \mathrm{argmin}_{B^T f = \delta\chi} \mathrm{val}(f)$, the third is explicit computation of D', and the fourth follows from centrality of $f^*_{t,w+\mu}$. Therefore, the $(t+\delta)\chi$-flow which is $f^*_{t,w+\mu} + \widehat{f}$ satisfies

$$
\left[\frac{w_e^+ + \mu_e^+}{u_e^+ - \left[f^*_{t,w+\mu}\right]_e - \widehat{f}_e} - \frac{w_e^- + \mu_e^-}{u_e^- + \left[f^*_{t,w+\mu}\right]_e + \widehat{f}_e}\right] = [B(y+z)]_e
$$

hence is central for weights $w + \mu$. So $f^*_{t+\delta,w+\mu} = f^*_{t,w+\mu} + \widehat{f} = f^*_{t,w} + \widehat{f}$.

Now, note that ν as defined in lines 7 to 12 of Algorithm 1 satisfies

$$
\frac{\nu_e^+}{c_e^+ - \widehat{f}_e} - \frac{\nu_e^-}{c_e^- + \widehat{f}_e} = \frac{\mu_e^+}{c_e^+ - \widehat{f}_e} - \frac{\mu_e^-}{c_e^- + \widehat{f}_e}
$$

and centrality conditions (4.1) tell us that $f^*_{t+\delta,w+\nu} = f^*_{t+\delta,w+\mu}$.

We now bound $\|\nu\|_1$. Line 12 of Algorithm 1 and $\mu_e^+/c_e^+ = \mu_e^-/c_e^-$ gives us that

$$
\begin{aligned}
\nu_e^+ + \nu_e^- &= -(c_e^- + \widehat{f}_e)\left(\frac{\mu_e^+}{c_e^+ - \widehat{f}_e} - \frac{\mu_e^-}{c_e^- + \widehat{f}_e}\right) \\
&= -\mu_e^-\left(\left(\frac{c_e^- + \widehat{f}_e}{c_e^+ - \widehat{f}_e}\right)\frac{c_e^+}{c_e^-} - 1\right) \leq 3c_e^{-1}|\widehat{f}_e|\mu_e^-,
\end{aligned}
$$

where we have used that $c_e^{-1}|\widehat{f}_e| \leq 1/20$. A similar analysis of Line 10 gives that

$$
\nu_e^+ + \nu_e^- = (c_e^+ - \widehat{f}_e)\left(\frac{\mu_e^+}{c_e^+ - \widehat{f}_e} - \frac{\mu_e^-}{c_e^- + \widehat{f}_e}\right) = \mu_e^+\left(1 - \left(\frac{c_e^+ - \widehat{f}_e}{c_e^- + \widehat{f}_e}\right)\frac{c_e^-}{c_e^+}\right) \leq 3c_e^{-1}|\widehat{f}_e|\mu_e^+.
$$

[7] Even with an oracle for smoothed ℓ_2-ℓ_p flow that handles arbitrary weights, we do not know how to achieve maxflow runtimes faster than those achieved by this paper.

In both cases, we have that $\nu_e^+ + \nu_e^- \le 3c_e^{-1}|\widehat{f}_e|(\mu_e^+ + \mu_e^-)$. Using the choice $W = \epsilon^2 m^3/F_t^2$, $F_t \ge 10\epsilon m^{1+2/p}U$, and (5.14), Lemma 5.6 yield

$$\begin{aligned}
\|\nu\|_1 &\le \sum_{e\in E} 3c_e^{-1}|\widehat{f}_e|(\mu_e^- + \mu_e^+) \le 6W \sum_{e\in E} |\widehat{f}_e| c_e^- \cdot \frac{v_e^{p-1}}{\|v\|_p^{p-1}} \\
&\le 12\frac{\epsilon^2 m^3}{F_t^2}\cdot\frac{1}{500}\frac{F_t}{\sqrt{\epsilon m}}U\left\|\frac{v_e^{p-1}}{\|v\|_p^{p-1}}\right\|_1 \le \frac{\epsilon^{3/2}m^2U}{F_t}m^{1/p}\left\|\frac{v_e^{p-1}}{\|v\|_p^{p-1}}\right\|_q \le \sqrt{\epsilon}m^{1-1/p}. \square
\end{aligned}$$

Overall, Lemma 5.7 shows that one step of Algorithm 1 decreases the residual flow F_t by a multiplicative $(1-\delta) = (1-\sqrt{\epsilon}/10^5)$, and moves from one central path point $f^*_{t,w+}$ to $f^*_{t+\delta,w+\nu}$. Thus, the number of iterations is bounded by $\widetilde{O}(\epsilon^{-1/2})$. The cost of each iteration is one call to a solver for $\min_{B^Tf=d}\widetilde{\mathrm{val}}(f)$, which we show can be computed in $m^{1+o(1)}$ time in Lemma 6.1. Additionally, the total weight increase is at most $\sqrt{\epsilon}m^{1-1/p}$ so long as $F_t \ge 10\epsilon m^{1+2/p}U$. For the choice $\epsilon = (mU)^{-2/3}$, the number of iterations is $\widetilde{O}(m^{1/3}U^{1/3})$, and the amount of residual flow is at most $10\epsilon m^{1+2/p}U \le m^{1/3+o(1)}U^{1/3}$. Since that amount of residual flow can be routed in that many iterations of augmenting paths, after running our IPM method, we can get an exact maxflow in $m^{1/3+o(1)}U^{1/3}$ rounds of augmenting paths.

6. Runtime Analysis. In Subsection 6.1 we show how to implement a single round of Algorithm 1 in almost linear $m^{1+o(1)}$ time by making calls to a smoothed ℓ_2-ℓ_p flow oracle. In particular, we show in Theorem 6.3, using the iterative refinement framework of [1], that a more general class of flow problems can all be solved using $m^{o(1)}$ calls to a smoothed ℓ_2-ℓ_p flow oracle. Then in Subsection 6.2 we apply Lemma 5.7 to give our final maxflow algorithm which uses $\widetilde{O}(1/\sqrt{\epsilon})$ iterations of Algorithm 1, which costs $m^{1+o(1)}$ time per iteration, plus $O(\epsilon m^{1+2/p}U)$ rounds of augmenting paths. For $\epsilon = (mU)^{-2/3}$, the total runtime is $m^{4/3+o(1)}U^{1/3}$. Also, in the proof of Theorem 1.1 in Subsection 6.2, we verify that the total weight increase over the iterations of our algorithm is bounded by $m/2$.

6.1. Efficient Divergence Maximization. Lemma 5.7 shows that our algorithm just needs to compute $\mathrm{argmin}_{B^Tf=\delta\chi}\widetilde{\mathrm{val}}(f)$ in Line 4 of Algorithm 1, as all other lines clearly take $O(m)$ time. Here, we show how to do this in time $m^{1+o(1)}$.

Lemma 6.1. *There is an algorithm that in $m^{1+o(1)}$ time computes a flow f' with $B^Tf' = \delta\chi$ and $\widetilde{\mathrm{val}}(f') \le \min_{B^Tf=\delta\chi}\widetilde{\mathrm{val}}(f) + \frac{1}{2^{\mathrm{poly}(\log m)}}$.*

To prove Lemma 6.1, we extend the following result of [21] and show the general Theorem 6.3, which shows that flow problems that are combinations a quadratic and ℓ_p norm of functions with stable Hessians may be solved to high accuracy in almost linear time. This encompasses problems such as computing electric and ℓ_p norm minimizing flows, smoothed ℓ_2-ℓ_p flows, and the divergence maximizing flows our algorithm must compute.

Theorem 6.2 (Theorem 1.1 in [21], arXiv version). *Consider $p \in (\omega(1), (\log n)^{2/3-o(1)})$, $g \in \mathbb{R}^E$, $r \in \mathbb{R}^E_{\ge 0}$, demand vector $d \in \mathbb{R}^V$, real number $s \ge 0$, and initial solution $f_0 \in \mathbb{R}^E$ such that all parameters are bounded by $2^{\mathrm{poly}(\log m)}$ and $B^Tf_0 = d$. For a flow f, define*

$$\mathrm{val}_{g,r,s}(f) \stackrel{\mathrm{def}}{=} \sum_{e\in E} g_e f_e + \left(\sum_{e\in E} r_e f_e^2\right) + s\|f\|_p^p \quad \textit{and} \quad OPT_{g,r,s} \stackrel{\mathrm{def}}{=} \min_{B^Tf=d} \mathrm{val}_{g,r,s}(f).$$

There is an algorithm that in $m^{1+o(1)}$ time computes a flow f such that $B^T f = d$ and

$$\mathrm{val}_{g,r,s}(f) - OPT_{g,r,s} \le \frac{1}{2^{\mathrm{poly}(\log m)}}(\mathrm{val}_{g,r,s}(f_0) - OPT_{g,r,s}) + \frac{1}{2^{\mathrm{poly}(\log m)}}.$$

Theorem 6.3. *For graph $G = (V, E)$ and all $e \in E$, let $a_e \in [0, 2^{\mathrm{poly}(\log m)}]$ be constants, $q_e : \mathbb{R} \to \mathbb{R}$ be functions with $|q_e(0)|, |q_e'(0)| \le 2^{\mathrm{poly}(\log m)}$ and $q_e''(x) \in [a_e/4, 4a_e]$ for all $x \in \mathbb{R}$, and $h_e : \mathbb{R} \to \mathbb{R}$ be functions with $h_e(0) = h_e'(0) = 0$ and $h_e''(x) \in [1/4, 4]$ for all $x \in \mathbb{R}$. For demand $d \in \mathbb{R}^V$ with entries bounded by $2^{\mathrm{poly}(\log m)}$, even integer $p \in (\omega(1), (\log n)^{2/3-o(1)})$, and all flows $f \in \mathbb{R}^E$ define*

$$\mathrm{val}(f) \stackrel{\mathrm{def}}{=} \sum_{e \in E} q_e(f_e) + \left(\sum_{e \in E} h_e(f_e)^p\right)^{1/p} \quad \text{and} \quad OPT \stackrel{\mathrm{def}}{=} \min_{B^T f = d} \mathrm{val}(f).$$

We can compute in time $m^{1+o(1)}$ a flow f' with $B^T f' = d$ and $\mathrm{val}(f') \le OPT + \frac{1}{2^{\mathrm{poly}(\log m)}}$.

The reason that we enforce that $a_e, |q_e(0)|, |q_e'(0)| \le 2^{\mathrm{poly}(\log m)}$ is that [21, Theorem 1.1] requires quasipolynomially bounded inputs to achieve its $m^{1+o(1)}$ runtime. Also, while [21, Theorem 1.1] is stated with $\frac{1}{\mathrm{poly}(m)}$ errors, it can be made to $\frac{1}{2^{\mathrm{poly}(\log m)}}$ as explained in [27] Appendix D.3.

Though the full proof of Theorem 6.3 is deferred to Appendix D, we give a brief proof sketch here. Roughly, we first use Lemma B.3 of [27] to reduce optimizing the objective $\mathrm{val}(f)$ in Theorem 6.3 to solving $\widetilde{O}(1)$ problems of the following kind:

$$\mathrm{val}_{p,W}(f) \stackrel{\mathrm{def}}{=} \sum_{e \in E} q_e(f_e) + W \sum_{e \in E} h_e(f_e)^p \quad \text{and} \quad OPT_{p,W} \stackrel{\mathrm{def}}{=} \min_{B^T f = d} \mathrm{val}_{p,W}(f)$$

find a flow f' with $B^T f' = d$ and $\mathrm{val}_{p,W}(f') \le OPT_{p,W} + \frac{1}{2^{\mathrm{poly}(\log m)}}$. This achieved through careful binary search which is elaborated on Appendix D and proven formally in Lemma D.1.

We then apply the iterative refinement framework to reduce the high accuracy minimization of the objective $\mathrm{val}_{p,W}(f)$ to solving $\widetilde{O}(2^{O(p)})$ smoothed quadratic and ℓ_p norm flow problems, which may be solved using Theorem 6.2. The main difference from the analysis of [21] is that we show that any convex function h with stable second derivatives admits an expansion for $h(x+\Delta)^p - h(x)^p$, while [21] only considers the function $h(x) = x^2$. This is done formally in Lemma D.2.

Proof of Lemma 6.1. By scaling down by W, it suffices to check that $\widetilde{\mathrm{val}}(f)$ satisfies the constraints of Theorem 6.3 for

$$q_e(x) \stackrel{\mathrm{def}}{=} W^{-1}\left(w_e^+ \widetilde{D}\left(\frac{x}{c_e^+}\right) + w_e^- \widetilde{D}\left(-\frac{x}{c_e^-}\right)\right) \quad \text{and} \quad a_e \stackrel{\mathrm{def}}{=} W^{-1}\left(\frac{w_e^+}{(c_e^+)^2} + \frac{w_e^-}{(c_e^-)^2}\right),$$

$$h_e(x) \stackrel{\mathrm{def}}{=} (c_e^+)^2\left(\widetilde{D}\left(\frac{x}{c_e^+}\right) + \left(\frac{c_e^-}{c_e^+}\right)\widetilde{D}\left(-\frac{x}{c_e^-}\right)\right).$$

To analyze $q_e(x)$, we compute that

$$\begin{aligned} q_e''(x) &= W^{-1}\left(\frac{w_e^+}{(c_e^+)^2}\widetilde{D}''\left(\frac{x}{c_e^+}\right) + \frac{w_e^-}{(c_e^-)^2}\widetilde{D}''\left(\frac{x}{c_e^-}\right)\right) \\ &\le 2W^{-1}\left(\frac{w_e^+}{(c_e^+)^2} + \frac{w_e^-}{(c_e^-)^2}\right) = 2a_e \end{aligned}$$

by Lemma 5.1. A lower bound $q_e''(x) \geq \frac{1}{2}W^{-1}\left(\frac{w_e^+}{(c_e^+)^2} + \frac{w_e^-}{(c_e^-)^2}\right) = a_e/2$ follows equivalently. $a_e \leq 2^{\mathrm{poly}(\log m)}$ follows from the fact that resistances and residual capacities are polynomially bounded on the central path – see [27] Lemma D.1.

To analyze $h_e(x)$, we can compute using (5.2) that $h_e(0) = h_e'(0) = 0$ and

$$h_e''(x) = \widetilde{D}''\left(\frac{x}{c_e^+}\right) + \frac{c_e^+}{c_e^-}\widetilde{D}''\left(-\frac{x}{c_e^-}\right) \leq 4$$

by Lemma 5.1. We get $h_e''(x) \geq 1/2$ similarly. ∎

6.2. Main Algorithm. We now combine Lemma 6.1 with Lemma 5.7 in Algorithm 2 to prove Theorem 1.1. Our algorithm repeatedly takes steps computed with Algorithm 1 until the remaining flow F_t is at most $\epsilon m^{1+o(1)}U$.

Algorithm 2: MAXFLOW(ϵ, G). Takes a preconditioned undirected graph G with maximum capacity U. Returns the maximum ab flow in G.

---

1 $f \leftarrow 0, t \leftarrow 0, w \leftarrow 1$.
2 **while** $F_t \geq 10\epsilon m^{1+2/p}U$ **do**
3 $\quad (\delta, \widehat{f}, \nu) \leftarrow$ AUGMENT$(G, w, t^* - t, f)$.
4 $\quad f \leftarrow f + \widehat{f}, w \leftarrow w + \nu$, and $t \leftarrow t + \delta$.
5 Round to an integral flow [22, 28] and use augmenting paths until done.

Proof of Theorem 1.1. We show that MAXFLOW(G) computes a maximum flow on G in time $m^{1+o(1)}/\sqrt{\epsilon}$ time. Correctness follows from Lemmas 5.7 and 6.1. It suffices then to maintain upper bounds control the weights. Our choice of δ guarantees that we route $\Omega(\sqrt{\epsilon})$ fraction of the remaining flow per iteration, hence Line 2 executes $\widetilde{O}(1/\sqrt{\epsilon})$ times. $\|\nu\|_1 \leq \sqrt{\epsilon}m^{1-1/p}$ always by Lemma 5.7, hence at the end of the algorithm we have

$$\|w\|_1 \leq 2m + \widetilde{O}\left(1/\sqrt{\epsilon} \cdot \sqrt{\epsilon}m^{1-1/p}\right) \leq 5m/2\ .$$

To analyze the runtime, first note that by Lemma 6.1 which we will establish in the next section, Line 3 takes $m^{1+o(1)}$ time, so the total runtime of these throughout the algorithm is $m^{1+o(1)}/\sqrt{\epsilon}$ as desired. Note that the total remaining flow at the end is at most $10\epsilon m^{1+2/p}U$. Now, we may take $\epsilon \to \frac{\epsilon}{10m^{2/p}}$ to get the desired bounds in Theorem 1.1. This completes our proof of Theorem 1.1. ∎

7. Conclusion. We conclude by first stating the difficulties in going beyond an $m^{4/3}$ runtime for maxflow, and then discussing potential directions for future research on the topic.

A Possible Barrier at $m^{4/3}$. Here we briefly discuss why we believe that $m^{4/3}$ is a natural runtime barrier for IPM based algorithms for maxflow on sparse unweighted graphs. All currently known weighted IPM advances satisfy the following two properties. First, weights only increase and do not become super linear. Second, the methods step from one central path point to the next one in $\Theta(m)$ time and the congestion of this step is multiplicatively bounded. Our algorithm does both these pieces optimally — we precisely compute weight changes under a budget to ensure that the congestion to the next central path point is reduced significantly. In this sense, to break the $m^{4/3}$ barrier, one would either have to find a new way to backtrack on weight changes on the central path so that they are not additive throughout the algorithm, or show better amortized bounds on weight change than shown here. Alternatively, one would need a novel algorithm to step to faraway points on the central path, outside a ball where the congestions are bounded.

Recent Progress and Future Directions. Since the original version of this paper was published, there have been several exciting improvements to the runtimes of maxflow and mincost in capacitated and dense graphs. The current fastest algorithms for graphs with arbitrary polynomially bounded capacities (as opposed to the unit weight case considered in this paper) have runtime $\widetilde{O}((m + n^{3/2}) \log^2 U)$ for mincost flow [40, 39] and $\widetilde{O}(m^{3/2-1/328} \log U)$ for maxflow [13]. The former algorithms are nearly linear time on sufficiently dense graphs, and the latter is the first $m^{3/2-\Omega(1)} \log U$ time algorithm for capacitated maxflow on sparse graphs. Interestingly, these algorithms do not work by reducing the iteration complexity as is done in this paper, and instead use *dynamic data structures* to speed up the runtime of implementing each iteration of an IPM.

We believe that several exciting directions remain, in the context of this work as well as the recent progress discussed. First, can we combine ideas from this work and [40, 39] to achieve $mn^{1/3+o(1)}$ or even $m + n^{4/3+o(1)}$ runtimes for unit capacity maxflow? Also, can we design an $m^{4/3+o(1)}$ time algorithm for maxflow that only uses Laplacian system solvers, as opposed to stronger ℓ_2-ℓ_p norm flows? Also, is there an IPM for capacitated maxflow that runs in $m^{1/2-\Omega(1)} \log^{O(1)} U$ iterations? Finally, on the other hand, can we use stronger primitives such as smoothed ℓ_2-ℓ_p flows to design even more efficient algorithms for unit capacity maxflow, potentially beyond the use of IPMs?

8. Acknowledgements. Yang P. Liu was supported by the Department of Defense (DoD) through the National Defense Science and Engineering Graduate Fellowship (NDSEG) Program. Aaron Sidford was supported by NSF CAREER Award CCF-1844855. Tarun Kathuria is supported by NSF Grant CCF 1718695. We thank Arun Jambulapati, Michael B. Cohen, Yin Tat Lee, Jonathan Kelner, Aleksander Mądry, and Richard Peng for helpful discussions. The first author would like to thank Jelena Diakonikolas and Daniel Spielman for helpful discussions. The second and third authors are extremely grateful to Yin-Tat Lee for the fruitful suggestion of using quadratic extensions. Finally, we would like to thank anonymous reviewers for helping us improve the presentation of the paper.

REFERENCES

[1] D. Adil, R. Kyng, R. Peng, and S. Sachdeva, *Iterative refinement for ℓ_p-norm regression*, in Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019, 2019, pp. 1405–1424, https://doi.org/10.1137/1.9781611975482.86, https://doi.org/10.1137/1.9781611975482.86.

[2] D. Adil and S. Sachdeva, *Faster p-norm minimizing flows, via smoothed q-norm problems*, in Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, 2020, pp. 892–910.

[3] K. Axiotis, A. Mądry, and A. Vladu, *Circulation control for faster minimum cost flow in unit-capacity graphs*, in 2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS), IEEE, 2020, pp. 93–104.

[4] S. Bubeck, M. B. Cohen, Y. T. Lee, and Y. Li, *An homotopy method for ℓ_p regression provably beyond self-concordance and in input-sparsity time*, in Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018, I. Diakonikolas, D. Kempe, and M. Henzinger, eds., ACM, 2018, pp. 1130–1137, https://doi.org/10.1145/3188745.3188776, https://doi.org/10.1145/

3188745.3188776.

[5] H. H. Chin, A. Mądry, G. L. Miller, and R. Peng, *Runtime guarantees for regression problems*, in Innovations in Theoretical Computer Science, ITCS '13, Berkeley, CA, USA, January 9-12, 2013, R. D. Kleinberg, ed., ACM, 2013, pp. 269–282, https://doi.org/10.1145/2422436.2422469, https://doi.org/10.1145/2422436.2422469.

[6] P. Christiano, J. A. Kelner, A. Mądry, D. A. Spielman, and S. Teng, *Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs*, in Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011, 2011, pp. 273–282, https://doi.org/10.1145/1993636.1993674, https://doi.org/10.1145/1993636.1993674.

[7] J. Chuzhoy, Y. Gao, J. Li, D. Nanongkai, R. Peng, and T. Saranurak, *A deterministic algorithm for balanced cut with applications to dynamic connectivity, flows, and beyond*, CoRR, abs/1910.08025 (2019), http://arxiv.org/abs/1910.08025, https://arxiv.org/abs/1910.08025.

[8] E. Cohen, *Approximate max-flow on small depth networks*, SIAM J. Comput., 24 (1995), pp. 579–597, https://doi.org/10.1137/S0097539792236717, https://doi.org/10.1137/S0097539792236717.

[9] M. B. Cohen, Y. T. Lee, and Z. Song, *Solving linear programs in the current matrix multiplication time*, in Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019., 2019, pp. 938–942.

[10] M. B. Cohen, A. Mądry, P. Sankowski, and A. Vladu, *Negative-weight shortest paths and unit capacity minimum cost flow in $\widetilde{O}(m^{10/7}\log w)$ time (extended abstract)*, in Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19, 2017, pp. 752–771.

[11] S. I. Daitch and D. A. Spielman, *Faster approximate lossy generalized flow via interior point algorithms*, in Proceedings of the fortieth annual ACM symposium on Theory of computing, ACM, 2008, pp. 451–460.

[12] S. Even and R. E. Tarjan, *Network flow and testing graph connectivity*, SIAM J. Comput., 4 (1975), pp. 507–518, https://doi.org/10.1137/0204043, https://doi.org/10.1137/0204043.

[13] Y. Gao, Y. P. Liu, and R. Peng, *Fully dynamic electrical flows: sparse maxflow faster than goldberg-rao*, arXiv preprint arXiv:2101.07233, (2021).

[14] A. V. Goldberg and S. Rao, *Beyond the flow decomposition barrier*, J. ACM, 45 (1998), pp. 783–797, https://doi.org/10.1145/290179.290181, https://doi.org/10.1145/290179.290181.

[15] D. R. Karger, *Using random sampling to find maximum flows in uncapacitated undirected graphs*, in Annual ACM Symposium on Theory of Computing: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing, vol. 4, 1997, pp. 240–249.

[16] D. R. Karger, *Better random sampling algorithms for flows in undirected graphs*, in SODA, vol. 98, Citeseer, 1998, pp. 490–499.

[17] D. R. Karger, *Random sampling in cut, flow, and network design problems*, Mathematics of Operations Research, 24 (1999), pp. 383–413.

[18] D. R. Karger and M. S. Levine, *Finding maximum flows in undirected graphs seems easier than bipartite matching*, in STOC, vol. 98, Citeseer, 1998, pp. 69–78.

[19] J. A. Kelner, Y. T. Lee, L. Orecchia, and A. Sidford, *An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations*, in Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014, 2014, pp. 217–226, https://doi.org/10.1137/1.9781611973402.16, https://doi.org/10.1137/1.9781611973402.16.

[20] J. A. Kelner, G. L. Miller, and R. Peng, *Faster approximate multicommodity flow using quadratically coupled flows*, in Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012, H. J. Karloff and T. Pitassi, eds., ACM, 2012, pp. 1–18, https://doi.org/10.1145/2213977.2213979, https://doi.org/10.1145/2213977.2213979.

[21] R. Kyng, R. Peng, S. Sachdeva, and D. Wang, *Flows in almost linear time via adaptive preconditioning*, in Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019., 2019, pp. 902–913, https://doi.org/10.1145/3313276.3316410, https://doi.org/10.1145/3313276.3316410.

[22] Y. T. Lee, S. Rao, and N. Srivastava, *A new approach to computing maximum flows using electrical flows*, in Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013, 2013, pp. 755–764, https://doi.org/10.1145/2488608.2488704, https://doi.org/10.1145/2488608.2488704.

[23] Y. T. Lee and A. Sidford, *Path finding methods for linear programming: Solving linear programs in $\widetilde{O}(\sqrt{rank})$ iterations and faster algorithms for maximum flow*, in 55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014, 2014, pp. 424–433, https://doi.org/10.1109/FOCS.2014.52, https://doi.org/10.1109/FOCS.2014.52.
[24] Y. T. Lee, Z. Song, and Q. Zhang, *Solving empirical risk minimization in the current matrix multiplication time*, in Conference on Learning Theory, COLT 2019, 25-28 June 2019, Phoenix, AZ, USA, A. Beygelzimer and D. Hsu, eds., vol. 99 of Proceedings of Machine Learning Research, PMLR, 2019, pp. 2140–2157, http://proceedings.mlr.press/v99/lee19a.html.
[25] H. Lin, *Reducing directed max flow to undirected max flow*, Unpublished Manuscript, 4 (2009).
[26] Y. P. Liu and A. Sidford, *Faster divergence maximization for faster maximum flow*, CoRR, abs/2003.08929 (2020), https://arxiv.org/abs/2003.08929, https://arxiv.org/abs/2003.08929.
[27] Y. P. Liu and A. Sidford, *Faster energy maximization for faster maximum flow*, in Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020, K. Makarychev, Y. Makarychev, M. Tulsiani, G. Kamath, and J. Chuzhoy, eds., ACM, 2020, pp. 803–814, https://doi.org/10.1145/3357713.3384247, https://doi.org/10.1145/3357713.3384247.
[28] A. Mądry, *Navigating central path with electrical flows: From flows to matchings, and back*, in 54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA, 2013, pp. 253–262, https://doi.org/10.1109/FOCS.2013.35, https://doi.org/10.1109/FOCS.2013.35.
[29] A. Mądry, *Computing maximum flow with augmenting electrical flows*, in IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA, 2016, pp. 593–602, https://doi.org/10.1109/FOCS.2016.70, https://doi.org/10.1109/FOCS.2016.70.
[30] Y. Nesterov, *Introductory lectures on convex programming volume i: Basic course*, Lecture notes, 3 (1998), p. 5.
[31] Y. E. Nesterov and A. Nemirovskii, *Interior-point polynomial algorithms in convex programming*, vol. 13 of Siam studies in applied mathematics, SIAM, 1994.
[32] J. B. Orlin, *Max flows in $O(nm)$ time, or better*, in Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013, 2013, pp. 765–774, https://doi.org/10.1145/2488608.2488705, https://doi.org/10.1145/2488608.2488705.
[33] R. Peng, *Approximate undirected maximum flows in $O(m\text{polylog}n)$ time*, in Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016, 2016, pp. 1862–1867, https://doi.org/10.1137/1.9781611974331.ch130, https://doi.org/10.1137/1.9781611974331.ch130.
[34] J. Renegar, *A polynomial-time algorithm, based on Newton's method, for linear programming*, Math. Program., 40 (1988), pp. 59–93.
[35] J. Sherman, *Nearly maximum flows in nearly linear time*, in 54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA, 2013, pp. 263–269, https://doi.org/10.1109/FOCS.2013.36, https://doi.org/10.1109/FOCS.2013.36.
[36] J. Sherman, *Area-convexity, ℓ_∞ regularization, and undirected multicommodity flow*, in Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017, 2017, pp. 452–460, https://doi.org/10.1145/3055399.3055501, https://doi.org/10.1145/3055399.3055501.
[37] A. Sidford and K. Tian, *Coordinate methods for accelerating ℓ_∞ regression and faster approximate maximum flow*, in 59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018, 2018, pp. 922–933, https://doi.org/10.1109/FOCS.2018.00091, https://doi.org/10.1109/FOCS.2018.00091.
[38] D. A. Spielman and S. Teng, *Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems*, in Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004, 2004, pp. 81–90, https://doi.org/10.1145/1007352.1007372, https://doi.org/10.1145/1007352.1007372.
[39] J. van den Brand, Y. T. Lee, Y. P. Liu, T. Saranurak, A. Sidford, Z. Song, and D. Wang, *Minimum cost flows, MDPs, and ℓ_1-regression in nearly linear time for dense instances*, in STOC, ACM, 2021, pp. 859–869.
[40] J. van den Brand, Y.-T. Lee, D. Nanongkai, R. Peng, T. Saranurak, A. Sidford, Z. Song, and D. Wang, *Bipartite matching in nearly-linear time on moderately dense graphs*, arXiv e-prints, (2020), pp. arXiv–2009.

[41] J. van den Brand, Y. T. Lee, A. Sidford, and Z. Song, *Solving tall dense linear programs in nearly linear time*, in Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020, K. Makarychev, Y. Makarychev, M. Tulsiani, G. Kamath, and J. Chuzhoy, eds., ACM, 2020, pp. 775–788, https://doi.org/10.1145/3357713.3384309, https://doi.org/10.1145/3357713.3384309.

Appendix A. Missing proofs.

A.1. Proof of Observation 1.

Proof. Assume for contradiction that $g(x) < g(x^*)$ for some $x \in S$. For all $\epsilon > 0$ we have

$$g((1-\epsilon)x^* + \epsilon x) \leq (1-\epsilon)g(x^*) + \epsilon g(x) < g(x^*).$$

Because f, g agree on a neighborhood of x^*, we have for sufficiently small ϵ that

$$f((1-\epsilon)x^* + \epsilon x) = g((1-\epsilon)x^* + \epsilon x) < g(x^*) = f(x^*),$$

a contradiction to $f(x^*) = \min_{x \in S} f(x)$, as $(1-\epsilon)x^* + \epsilon x \in S$ by convexity of S. ∎

A.2. Proof of Lemma 5.2.

Lemma A.1. *Let $p \geq 1$ be a real number. Let $h_i : \mathbb{R} \to \mathbb{R}_{\geq 0}$ be convex functions. Then the function $h : \mathbb{R}^n \to \mathbb{R}$ defined by $h(x) = \left(\sum_{i=1}^n h_i(x_i)^p\right)^{1/p}$ is a convex function.*

Proof. For any $x, y \in \mathbb{R}^n$ and $0 \leq t \leq 1$ we have by Minkowski's inequality and convexity that

$$\begin{aligned} h(tx + (1-t)y) &= \|h_i(tx_i + (1-t)y_i)\|_p \leq \|t \cdot h_i(x_i) + (1-t) \cdot h_i(y_i)\|_p \\ &\leq t\|h_i(x_i)\|_p + (1-t)\|h_i(y_i)\|_p = t \cdot h(x) + (1-t) \cdot h(y). \end{aligned}$$ ∎

Proof of Lemma 5.2. D and $\widetilde{D}$ are convex, hence $D_w^V(f)$ and $\widetilde{D}_w^V(f)$ are convex. Also, the functions

$$\left(s_e^+\right)^2 \left(D\left(\frac{f_e}{s_e^+}\right) + \left(\frac{s_e^-}{s_e^+}\right) D\left(-\frac{f_e}{s_e^-}\right)\right) \text{ and } \left(s_e^+\right)^2 \left(\widetilde{D}\left(\frac{f_e}{s_e^+}\right) + \left(\frac{s_e^-}{s_e^+}\right) \widetilde{D}\left(-\frac{f_e}{s_e^-}\right)\right)$$

are convex, hence $\mathrm{val}(f)$ and $\widetilde{\mathrm{val}}(f)$ are convex by Lemma A.1. ∎

Appendix B. Iterative Refinement.

Here, D and $\widetilde{D}$ are defined as in (5.1).

Lemma B.1. *Let $h : \mathbb{R} \to \mathbb{R}$ be a function with $h(0) = h'(0) = 0$, and let $c_1, c_2 > 0$ be constants such that $c_1 \leq h''(x) \leq c_2$ for all x. Then for $x \geq 0$ we have that $c_1 x \leq h'(x) \leq c_2 x$ and $-c_2 x \leq h'(-x) \leq -c_1 x$. Also, $\frac{1}{2}c_1 x^2 \leq h(x) \leq \frac{1}{2}c_2 x^2$ for all x.*

Proof. For $x \geq 0$ we have that $h'(x) = \int_0^x h''(y)dy$ and $c_1 x \leq \int_0^x h''(y)dy \leq c_2 x$. The proof for $x \leq 0$ is equivalent.

For $x \geq 0$ we have that $h(x) = \int_0^x h'(y)dy$ and

$$\frac{1}{2}c_1 x^2 = \int_0^x c_1 y dy \leq \int_0^x h'(y)dy \leq \int_0^x c_2 y dy = \frac{1}{2}c_2 x^2.$$

The proof for $x \leq 0$ is equivalent. ∎

Proof of Lemma 5.1. It suffices to show $1/2 \le \widetilde{D}''(x) \le 2$ and apply Lemma B.1. For $|x| \le 1/10$ we have that $\widetilde{D}''(x) = (1-x)^{-2}$. Now, we can check that for $|x| \le 1/10$ that $1/2 \le (1-x)^{-2} \le 2$. For $x \ge 1/10$ we have $\widetilde{D}''(x) = \widetilde{D}''(1/10)$ and for $x \le -1/10$ we have $\widetilde{D}''(x) = \widetilde{D}''(-1/10)$ as desired. ∎

LEMMA B.2 (Lemmas B.2 and B.3 from [21], arXiv version). *Let $p > 0$ be an even integer. For all real numbers x, Δ we have that*

$$2^{-p}\left(x^{p-2}\Delta^2 + \Delta^p\right) \le (x+\Delta)^p - \left(x^p + p \cdot x^{p-1}\Delta\right) \le p2^{p-1}\left(x^{p-2}\Delta^2 + \Delta^p\right).$$

LEMMA B.3. *Let $h : \mathbb{R} \to \mathbb{R}$ be a function and let $c_2 \ge c_1 > 0$ be constants such that $c_1 \le h''(x) \le c_2$ for all x. Then for all Δ we have that*

$$\frac{1}{2}c_1\Delta^2 \le h(x+\Delta) - (h(x) + h'(x)\Delta) \le \frac{1}{2}c_2\Delta^2. \tag{B.1}$$

If additionally $h(0) = h'(0) = 0$ and $c_1 \le h''(x) \le c_2$ for all x then for all even integers $p > 0$ and Δ we have that

$$\begin{aligned}
&(8c_2)^{-2p}c_1^{3p}\left(x^{2p-2}\Delta^2 + \Delta^{2p}\right) \le h(x+\Delta)^p - \left(h(x)^p + p \cdot h(x)^{p-1}h'(x)\Delta\right) && \text{(B.2)}\\
&\qquad \le (16c_2)^p\left(x^{2p-2}\Delta^2 + \Delta^{2p}\right) && \text{(B.3)}
\end{aligned}$$

Proof. The upper bound of (B.1) follows from

$$h(x+\Delta) - (h(x) + h'(x)\Delta) = \int_0^\Delta (\Delta - y)h''(x+y)dy \le c_2\int_0^\Delta (\Delta - y)dy = \frac{1}{2}c_2\Delta^2$$

by Lemma B.1. The lower bound follows equivalently.

We show the upper bound of (B.2). Use Lemma B.2 with $x \to h(x)$, $\Delta \to h(x+\Delta) - h(x)$ to get

$$\begin{aligned}
&h(x+\Delta)^p - \left(h(x)^p + p \cdot h(x)^{p-1}\left(h(x+\Delta) - h(x)\right)\right)\\
&\le p2^{p-1}\left(h(x)^{p-2}\left(h(x+\Delta) - h(x)\right)^2 + \left(h(x+\Delta) - h(x)\right)^p\right). && \text{(B.4)}
\end{aligned}$$

Using the upper bound of (B.1) and $h(x) \le \frac{1}{2}c_2x^2$ from Lemma B.1 yields

$$\begin{aligned}
&h(x+\Delta)^p - \left(h(x)^p + p \cdot h(x)^{p-1}h'(x)\Delta\right) && \text{(B.5)}\\
&\le h(x+\Delta)^p - \left(h(x)^p + p \cdot h(x)^{p-1}\left(h(x+\Delta) - h(x)\right)\right) + \frac{1}{2}p \cdot h(x)^{p-1}c_2\Delta^2 && \text{(B.6)}\\
&\le h(x+\Delta)^p - \left(h(x)^p + p \cdot h(x)^{p-1}\left(h(x+\Delta) - h(x)\right)\right) + p \cdot c_2^p x^{2p-2}\Delta^2. && \text{(B.7)}
\end{aligned}$$

We now bound (B.4). (B.1) and Lemma B.1 gives us

$$|h(x+\Delta) - h(x)| \le |h'(x)\Delta| + c_2\Delta^2 \le c_2|x\Delta| + c_2\Delta^2.$$

Using this and Lemma B.1 gives us that (B.4) is at most

$$\begin{aligned}
&p2^{p-1}\left(h(x)^{p-2}\left(c_2|x\Delta| + c_2\Delta^2\right)^2 + \left(c_2|x\Delta| + c_2\Delta^2\right)^p\right)\\
&\le p2^{p-1}\left(x^{2p-4}c_2^{p-2}\left(2c_2^2x^2\Delta^2 + 2c_2^2\Delta^4\right) + 2^{p-1}c_2^px^p\Delta^p + 2^{p-1}c_2^p\Delta^{2p}\right)\\
&\le p2^{p-1}c_2^p \cdot \left(4 + 2^{p-1} + 2^{p-1}\right) \cdot \left(x^{2p-2}\Delta^2 + \Delta^{2p}\right).
\end{aligned}$$

Collecting terms and combining this with (B.5), (B.6), (B.7) proves the upper bound of (B.2).

Now we show the lower bound of (B.2). As above, we use Lemma B.2 to get

$$\begin{aligned} &h(x+\Delta)^p - \left(h(x)^p + p \cdot h(x)^{p-1}\left(h(x+\Delta) - h(x)\right)\right) \\ &\geq 2^{-p}\left(h(x)^{p-2}\left(h(x+\Delta)-h(x)\right)^2 + \left(h(x+\Delta)-h(x)\right)^p\right). \end{aligned} \tag{B.8}$$

Using the lower bound of (B.1) and $h(x) \geq \frac{1}{2}c_1 x^2$ from Lemma B.1 gives us

$$h(x+\Delta)^p - \left(h(x)^p + p \cdot h(x)^{p-1} h'(x)\Delta\right) \tag{B.9}$$

$$\geq h(x+\Delta)^p - \left(h(x)^p + p \cdot h(x)^{p-1}\left(h(x+\Delta)-h(x)\right)\right) + \frac{1}{2}p \cdot h(x)^{p-1} c_1 \Delta^2 \tag{B.10}$$

$$\geq h(x+\Delta)^p - \left(h(x)^p + p \cdot h(x)^{p-1}\left(h(x+\Delta)-h(x)\right)\right) + p2^{-p}c_1^p \cdot x^{2p-2}\Delta^2. \tag{B.11}$$

If $|x| \geq \frac{c_1|\Delta|}{4c_2}$ then

$$\begin{aligned} p2^{-p}c_1^p \cdot x^{2p-2}\Delta^2 &\geq \frac{1}{2} \cdot p2^{-p}c_1^p\left(x^{2p-2}\Delta^2 + \left(\frac{c_1\Delta}{4c_2}\right)^{2p-2}\Delta^2\right) \\ &\geq (8c_2)^{-2p}c_1^{3p} \cdot \left(x^{2p-2}\Delta^2 + \Delta^{2p}\right) \end{aligned}$$

as desired. If $|x| \leq \frac{c_1|\Delta|}{4c_2}$ then applying the lower bound of (B.1) and Lemma B.1 gives us

$$h(x+\Delta) - h(x) \geq h'(x)\Delta + c_1\Delta^2/2 \geq -|c_2 x\Delta| + c_1\Delta^2/2 \geq c_1\Delta^2/4.$$

Therefore, we may lower bound (B.8) by

$$2^{-p}\left(c_1\Delta^2/4\right)^p = 2^{-3p}c_1^p\Delta^{2p} \geq 2^{-3p-1}c_1^p\left(x^{2p-2}\Delta^2 + \Delta^{2p}\right)$$

for $|x| \leq \frac{c_1|\Delta|}{4c_2} \leq \Delta$. Combining this with (B.9), (B.10), (B.11) gives the desired bound. $\square$

LEMMA B.4. *Let p be an even positive integer. Let $h_e(x) : \mathbb{R} \to \mathbb{R}$ be convex functions for $e \in E$, and for $x \in \mathbb{R}^E$ let $h(x) = \sum_{e\in E} h_e(x_e)$. Let $OPT = \min_{B^T f = d} h(f)$. Let f be a flow satisfying $B^T f = d$. Let $C_1, C_2 > 0$ be constants such that for all edges e, there are real numbers $r_e, s_e \geq 0$ and g_e, depending on f_e, such that for all $\Delta_e \in \mathbb{R}$*

$$C_1\left(r_e\Delta_e^2 + s_e\Delta_e^p\right) \leq h_e(f_e+\Delta_e) - \left(h_e(f_e) + g_e\Delta_e\right) \leq C_2\left(r_e\Delta_e^2 + s_e\Delta_e^p\right). \tag{B.12}$$

Let

$$\widehat{\Delta} = \operatorname*{argmin}_{B^T\Delta = 0} g^T\Delta + C_1\left(\sum_{e\in E} r_e\Delta_e^2 + \sum_{e\in E} s_e\Delta_e^p\right).$$

Then

$$\left(h\left(f + \frac{C_1}{C_2}\widehat{\Delta}\right) - OPT\right) \leq \left(1 - \frac{C_1}{C_2}\right)\left(h(f) - OPT\right).$$

Proof. Define $f^* = \operatorname{argmin}_{B^T f = d} h(f)$. Define $\Delta = f^* - f$. By the first inequality of (B.12), we get

$$g^T \widehat{\Delta} + C_1 \left(\sum_{e \in E} r_e \widehat{\Delta}_e^2 + \sum_{e \in E} s_e \widehat{\Delta}_e^p \right) \leq g^T \Delta + C_1 \left(\sum_{e \in E} r_e \Delta_e^2 + \sum_{e \in E} s_e \Delta_e^p \right)$$
$$\leq h(f + \Delta) - h(f) = OPT - h(f).$$

This and the right side inequality of (B.12) give us

$$h\left(f + \frac{C_1}{C_2}\widehat{\Delta}\right) - h(f) \leq \frac{C_1}{C_2} g^T \widehat{\Delta} + C_2 \left(\left(\frac{C_1}{C_2}\right)^2 \sum_{e \in E} r_e \widehat{\Delta}_e^2 + \left(\frac{C_1}{C_2}\right)^p \sum_{e \in E} s_e \widehat{\Delta}_e^p \right)$$
$$\leq \frac{C_1}{C_2} \left(g^T \widehat{\Delta} + C_1 \left(\sum_{e \in E} r_e \widehat{\Delta}_e^2 + \sum_{e \in E} s_e \widehat{\Delta}_e^p \right) \right) \leq \frac{C_1}{C_2} \left(OPT - h(f)\right).$$

Rearranging this gives the desired inequality. ∎

Appendix C. Additional Preliminaries. In this section, we state some preliminaries for convex optimization. These will be used in Appendix D. We assume all functions in this section to be convex. We also work in the ℓ_2 norm exclusively. Proofs for the results stated can be found in [30].

Matrices and norms. We say that a $m \times m$ matrix M is positive semidefinite if $x^T M x \geq 0$ for all $x \in \mathbb{R}^m$. We say that M is positive definite if $x^T M x > 0$ for all nonzero $x \in \mathbb{R}^m$. For $m \times m$ matrices A, B we write $A \succeq B$ if $A - B$ is positive semidefinite, and $A \succ B$ is $A - B$ is positive definite. For $m \times m$ positive semidefinite matrix M and vector $x \in \mathbb{R}^m$ we define $\|x\|_M = \sqrt{x^T M x}$. For $m \times m$ positive semidefinite matrices M_1, M_2 and $C > 0$ we say that $M_1 \approx_C M_2$ if $\frac{1}{C} x^T M_1 x \leq x^T M_2 x \leq C x^T M_1 x$ for all $x \in \mathbb{R}^m$.

Lipschitz functions. Here we define what it means for a function f to be Lipschitz and provide a lemma showing its equivalence to a bound on the norm of the gradient.

DEFINITION C.1 (Lipschitz Function). *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a function, and let $\mathcal{X} \subseteq \mathbb{R}^n$ be an open convex set. We say that f is L_1-Lipschitz on $\mathcal{X}$ (in the ℓ_2 norm) if for all $x, y \in \mathcal{X}$ we have that $|f(x) - f(y)| \leq L_1 \|x - y\|_2$.*

LEMMA C.2 (Gradient Characterization of Lipschitz Function). *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a differentiable function, and let $\mathcal{X} \subseteq \mathbb{R}^n$ be an open convex set. Then f is L_1-Lipschitz on $\mathcal{X}$ if and only if for all $x \in \mathcal{X}$ we have that $\|\nabla f(x)\|_2 \leq L_1$.*

Smoothness and strong convexity. We define what it means for a function f to be convex, smooth, and strongly convex. We say that a function f is convex on $\mathcal{X}$ if for all $x, y \in \mathcal{X}$ and $0 \leq t \leq 1$ that $f(tx + (1-t)y) \leq tf(x) + (1-t)f(y)$. We say that f is L_2-smooth on $\mathcal{X}$ if $\|\nabla f(x) - \nabla f(y)\|_2 \leq L_2 \|x - y\|_2$ for all $x, y \in \mathcal{X}$. We say that f is μ-strongly convex on $\mathcal{X}$ if for all $x, y \in \mathcal{X}$ and $0 \leq t \leq 1$ that

$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y) - t(1-t) \cdot \frac{\mu}{2} \|x - y\|_2^2.$$

LEMMA C.3. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a differentiable function, and let $\mathcal{X} \subseteq \mathbb{R}^n$ be an open convex set. Then f is μ-strongly convex on $\mathcal{X}$ if and only if for all $x, y \in \mathcal{X}$ we have that*

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) + \frac{\mu}{2} \|y - x\|_2^2.$$

Also, f is L_2-smooth on $\mathcal{X}$ if and only if for all $x, y \in \mathcal{X}$ we have that

$$f(y) \leq f(x) + \nabla f(x)^T (y - x) + \frac{L_2}{2} \|y - x\|_2^2.$$

We can equivalently view smoothness and strong convexity as spectral bounds on the Hessian of f.

LEMMA C.4. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a twice differentiable function, and let $\mathcal{X} \subseteq \mathbb{R}^n$ be an open convex set. Then f is μ-strongly convex on a convex set $\mathcal{X}$ if and only if $\nabla^2 f(x) \succeq \mu I$ for all $x \in \mathcal{X}$. f is L_2-smooth on $\mathcal{X}$ if and only if $\nabla^2 f(x) \preceq L_2 I$ for all $x \in \mathcal{X}$.*

Smoothness allows us to relate function error and the norm of the gradient.

LEMMA C.5. *Let $\mathcal{X} \subseteq \mathbb{R}^n$ be an open convex set, and let $f : \mathbb{R}^n \to \mathbb{R}$ be L_2-smooth on $\mathcal{X}$. Define $x^* = \operatorname{argmin}_{x \in \mathbb{R}^n} f(x)$, and assume that x^* exists and $x^* \in \mathcal{X}$. Then for all $x \in X$ we have that*

$$\|\nabla f(x)\|_2^2 \leq 2L_2(f(x) - f(x^*)).$$

Strong convexity allows us to relate function error and distance to the optimal point.

LEMMA C.6. *Let $\mathcal{X} \subseteq \mathbb{R}^n$ be an open convex set, and let $f : \mathbb{R}^n \to \mathbb{R}$ be μ-strongly convex on $\mathcal{X}$. Define $x^* = \operatorname{argmin}_{x \in \mathbb{R}^n} f(x)$, and assume that x^* exists and $x^* \in \mathcal{X}$. Then for all $x \in \mathcal{X}$ we have that*

$$\|x - x^*\|_2^2 \leq \frac{2(f(x) - f(x^*))}{\mu}.$$

Appendix D. Proof of Theorem 6.3. In this section we show Theorem 6.3 through a series of reductions, where we work with a more general space of regression problems. Our first reduction is from smoothed quadratic and ℓ_p regression problems with bounded entries to solving smoothed quadratic and ℓ_p^p regression, so that the ℓ_p norm piece is instead raised to the p power. The latter formulation is more amenable to iterative refinement inequalities shown in Appendix B. The proof of this reduction is essentially a binary search on W to ensure that the desired KKT conditions of (D.1) are satisfied to high precision, using [27, Lemma B.3] and carefully tracking errors.

LEMMA D.1. *Let $A \in \mathbb{R}^{n \times m}$ be a matrix and $d \in \mathbb{R}^n$ a vector, all with entries bounded by $2^{\mathrm{poly}(\log m)}$. Assume that all nonzero singular values of A are between $2^{-\mathrm{poly}(\log m)}$ and $2^{\mathrm{poly}(\log m)}$. For $1 \leq i \leq m$ let $0 \leq a_i \leq 2^{\mathrm{poly}(\log m)}$ be constants and $q_i : \mathbb{R} \to \mathbb{R}$ be functions such that $|q_i(0)|, |q_i'(0)| \leq 2^{\mathrm{poly}(\log m)}$ and $a_i/4 \leq q_i''(x) \leq 4a_i$ for all $x \in \mathbb{R}$. For $1 \leq i \leq m$ let $0 \leq b_i \leq 2^{\mathrm{poly}(\log m)}$ be constants and $h_i : \mathbb{R} \to \mathbb{R}$ be functions such that $h_i(0) = h_i'(0) = 0$ and $b_i/4 \leq h_i''(x) \leq 4b_i$ for all $x \in \mathbb{R}$. For an even integer $p \leq \log m$ define*

$$\text{(D.1)} \qquad \mathrm{val}(x) \stackrel{\mathrm{def}}{=} \sum_{i=1}^m q_i(x_i) + \left(\sum_{i=1}^m h_i(x_i)^p\right)^{1/p} \text{ and } OPT \stackrel{\mathrm{def}}{=} \min_{Ax=d} \mathrm{val}(x).$$

We can compute an x' with $Ax' = d$ and $\mathrm{val}(x') \leq OPT + \frac{1}{2^{\mathrm{poly}(\log m)}}$ in $\widetilde{O}(1)$ oracle calls which for $0 \leq W \leq 2^{\mathrm{poly}(\log m)}$ and

$$\text{(D.2)} \qquad \mathrm{val}_{p,W} \stackrel{\mathrm{def}}{=} \sum_{i=1}^m q_i(x_i) + W \sum_{i=1}^m h_i(x_i)^p \text{ and } OPT_{p,W} \stackrel{\mathrm{def}}{=} \min_{Ax=d} \mathrm{val}_{p,W}(x)$$

computes a x' with $Ax' = d$ and $\mathrm{val}_{p,W}(x') \le OPT_{p,W} + \frac{1}{2^{\mathrm{poly}(\log m)}}$.

Proof. We will apply [27, Lemma B.3]. Thus we must define the functions f, g, h and choose constants $C_0, \chi, T_1, T_2, \mu_f, L_g, L_h, Z_1, Z_2, H_1, H_2, \epsilon_1$ satisfying its constraints. For simplicity, we assume that A has rank n, so that AA^T is invertible. In the case where $A = B^T$, a graph incidence matrix, the nullspace is simply the 1 vector, and the analysis can proceed similarly. We pick $C_0 = 1$ and $\chi = \{x \in \mathbb{R}^n : \|x\|_\infty < 2^{\mathrm{poly}(\log m)}\}$, which is valid because all derivatives of q_i, h_i and condition number of A are bounded by $2^{\mathrm{poly}(\log m)}$.

Regularizing the objective. Set $\nu = 2^{-\mathrm{poly}(\log m)}$ to be some sufficiently small parameter, and replace each $q_i(x) \to q_i(x) + \nu x^2$. Clearly, for all $x \in \chi$, the value of the objective is affected by at most $\nu\|x\|_2^2 = 2^{-\mathrm{poly}(\log m)}$ for sufficiently small ν. From this point forwards, we assume that $q_i''(x) \ge 2\nu$ for all x. We also assume that the demand d has all components at least ν, as changing $d \to d + \nu 1$ affects the objective value by at most $2^{-\mathrm{poly}(\log m)}$.

Reduction to unconstrained problem and choice of f, g, h. We first reduce to the unconstrained case by removing the constraint $Ax = d$. Define $x_0 = A^T(AA^T)^{-1}d$, so that $Ax_0 = d$, $P \in \mathbb{R}^{m\times(m-n)}$ be an isomorphism onto the nullspace of A, which may be computed by inverting an arbitrary $n \times n$ minor of A. Specifically, if $A = \begin{bmatrix} X & Y \end{bmatrix}$ where $X \in \mathbb{R}^{n\times(m-n)}$ and $Y \in \mathbb{R}^{n\times n}$ is invertible, we set $P = \begin{bmatrix} I_{m-n} \\ Y^{-1}X \end{bmatrix}$. In the case $A = B^T$, we may take P to be determined by the case where Y corresponds to a tree. Thus, we may replace the condition $Ax = d$ with $x = Py + x_0$ for some $y \in \mathbb{R}^{m-n}$. Our choice of f, g, h are there

$$f(y) \stackrel{\mathrm{def}}{=} \sum_{i=1}^m q_i([Py + x_0]_i) \quad \text{and} \quad g(y) \stackrel{\mathrm{def}}{=} \left(\sum_{i=1}^m h_i([Py + x_0]_i)^p\right)^{1/p} \quad \text{and} \quad h(x) = x^p.$$

Choice of remaining parameters. As we have regularized each $q_i(x)$, we have that $\nabla^2 f(x) \succeq 2P^T \nu P \succeq 2^{-\mathrm{poly}(\log m)} I$, as $\nu \ge 2^{-\mathrm{poly}(\log m)}$ and $P^T P \succeq 2^{-\mathrm{poly}(\log m)} I$ by the condition number bound on A. Thus, we may set $\mu_f = 2^{-\mathrm{poly}(\log m)}$ by Lemma C.4. As all entries of A are bounded by $2^{\mathrm{poly}(\log m)}$ and all entries of d are at least ν in absolute value by our reduction, all x with $Ax = d$ are at least $2^{-\mathrm{poly}(\log m)}$ in some coordinate. Thus $f(y), g(y) \ge 2^{-\mathrm{poly}(\log m)}$, so we may set $T = 2^{-\mathrm{poly}(\log m)}$. By our choice of χ, $f(y), g(y) \le 2^{\mathrm{poly}(\log m)}$ for all $y \in \chi$, so we set $T_2 = 2^{\mathrm{poly}(\log m)}$.

We may set $L_g = 2^{\mathrm{poly}(\log m)}$, and for $p \le \log m$ and our choice $h(x) = x^p$ and T_1, T_2 we may set $L_h = 2^{\mathrm{poly}(\log m)}$. We set $H_1 = h(T_1) \ge 2^{-\mathrm{poly}(\log m)}$ and $H_2 = h(T_2) \le 2^{\mathrm{poly}(\log m)}$. Finally, we set $\epsilon_1 = 2^{-\mathrm{poly}(\log m)}$. We set $Z_1 = 0$ and $Z_2 = \frac{C_0}{h'(T_1)} \le 2^{\mathrm{poly}(\log m)}$. These parameters satisfy all desired properties, and

$$\log \max\{L_g, L_h, Z_2, H_2\} = \widetilde{O}(1) \quad \text{and} \quad \log \min\{\mu_f, \epsilon_1\} = -\widetilde{O}(1).$$

So, [27, Lemma B.3] Equation (21) tells us it suffices to make $O\left(\log \frac{H_2 Z_2 L_g L_h}{\mu_f \epsilon_1}\right) = \widetilde{O}(1)$ oracle calls with accuracy parameter $\frac{\mu_f \epsilon_1^2}{100 Z_2^2 L_g^4 L_h^2} \ge 2^{-\mathrm{poly}(\log m)}$, as desired.

Finishing the proof. We now argue that the oracle described in Lemma D.1 satisfies Equations (19) and (20) of [27, Lemma B.3]. Equation (19) follows by definition, and Equation (20) follows by Equation (19), $2^{\mathrm{poly}(\log m)}$ smoothness of the objective, and Lemma C.5. Thus, applying Lemma B.3 gives a y with

$$\|\nabla f(y) + \nabla g(y)\|_2 \le \epsilon_1.$$

Let $y^* = \operatorname{argmin}_y f(y) + g(y)$. By convexity we have that

$$(f(y) + g(y)) - (f(y^*) + g(y^*)) \leq (\nabla f(y) + \nabla g(y))^T (y - y^*)$$
$$\leq \|\nabla f(y) + \nabla g(y)\|_2 \|y - y^*\|_2 \leq 2^{-\mathrm{poly}(\log m)},$$

by our choice $\epsilon = 2^{-\mathrm{poly}(\log m)}$ that $\|y - y^*\|_2 \leq 2^{\mathrm{poly}(\log m)}$ from our choice of χ. □

We now show that objectives as in (D.2) may be iteratively refined.

Lemma D.2. *Let $A \in \mathbb{R}^{n \times m}$ be a matrix and $d \in \mathbb{R}^n$ a vector, all with entries bounded by $2^{\mathrm{poly}(\log m)}$. Assume that all nonzero singular values of A are between $2^{-\mathrm{poly}(\log m)}$ and $2^{\mathrm{poly}(\log m)}$. For $1 \leq i \leq m$ let $0 \leq a_i \leq 2^{\mathrm{poly}(\log m)}$ be constants and $q_i : \mathbb{R} \to \mathbb{R}$ be functions such that $|q_i(0)|, |q_i'(0)| \leq 2^{\mathrm{poly}(\log m)}$ and $a_i/4 \leq q_i''(x) \leq 4a_i$ for all $x \in \mathbb{R}$. For $1 \leq i \leq m$ let $0 \leq b_i \leq 2^{\mathrm{poly}(\log m)}$ be constants and $h_i : \mathbb{R} \to \mathbb{R}$ be functions such that $h_i(0) = h_i'(0) = 0$ and $b_i/4 \leq h_i''(x) \leq 4b_i$ for all $x \in \mathbb{R}$. For an even integer $p \leq \log m$ define $\mathrm{val}_{p,W}(x)$ and $OPT_{p,W}$ as in (D.2). We can compute a x' with $Ax' = d$ and $\mathrm{val}_{p,W}(x') \leq OPT_{p,W} + \frac{1}{2^{\mathrm{poly}(\log m)}}$ with one call to a solver for $(A^T A)^\dagger d$ and $\widetilde{O}(2^{22p})$ oracle calls which for $g \in \mathbb{R}^m, r \in \mathbb{R}^m_{\geq 0}$, all entries bounded by $2^{\mathrm{poly}(\log m)}$, and*

$$\mathrm{val}_{g,r,b}(x) \stackrel{\mathrm{def}}{=} \sum_{i=1}^m g_i x_i + \left(\sum_{i=1}^m r_i x_i^2\right) + \sum_{i=1}^m b_i^p x_i^{2p} \quad \text{and} \quad OPT_{g,r,b} = \min_{Ax=0} \mathrm{val}_{g,r,b}(x)$$

computes an x' with $Ax' = 0$ and $\mathrm{val}_{g,r,b}(x') \leq OPT_{g,r,b} + \frac{1}{2^{\mathrm{poly}(\log m)}}$.

Proof. We use Algorithm 3, which reduces (D.2) to oracle calls to Oracle2P.

Algorithm 3: ReduceTo2P(A, d, q, h, a, b). Takes matrix $A \in \mathbb{R}^{n \times m}$, vector d, constants a_i, b_i, and functions q_i, h_i for $1 \leq i \leq m$. Computes x with $Ax = d$ and $\mathrm{val}_{p,W}(x) \leq OPT_{p,W} + \frac{1}{2^{\mathrm{poly}(\log m)}}$ with $\widetilde{O}(2^{22p})$ calls to Oracle2P(A, g, r, b), which computes an x with $Ax = 0$ and $\mathrm{val}_{g,r,b}(x') \leq OPT_{g,r,b} + \frac{1}{2^{\mathrm{poly}(\log m)}}$.

1 Initialize $x \leftarrow A^T (AA^T)^\dagger d$.
2 **for** $1 \leq t \leq \widetilde{O}(2^{22p})$ **do**
3   Initialize r.
4   **for** $1 \leq i \leq m$ **do**
5         $r_i = a_i + b_i^p x_i^{2p-2}$.
6     $g \leftarrow \nabla \mathrm{val}_{p,W}(x)$.
7     $\widehat{\Delta} \leftarrow$ Oracle2P$(A, g, 2^{-16p} r, 2^{-16} b)$.
8     $x \leftarrow x + 2^{-22p} \widehat{\Delta}$.
9 Return x.

Note that all entries of $x = A^T (AA^T)^\dagger d$ are bounded by $2^{\mathrm{poly}(\log m)}$ because all nonzero singular values of A are between $2^{-\mathrm{poly}(\log m)}$ and $2^{\mathrm{poly}(\log m)}$. Therefore, $\mathrm{val}_{p,W}(x) \leq 2^{\mathrm{poly}(\log m)}$.

We show that iteration in Line 8 decreases the value of the objective on x multiplicatively towards OPT. Applying Lemma B.3 and (B.1) to q_i with $c_1 = a_i/4$ and $c_2 = 4a_i$ gives

$$\text{(D.3)} \qquad \frac{1}{8} a_i \Delta_i^2 \leq q_i(x_i + \Delta_i) - q_i(x_i) - q_i'(x_i)\Delta_i \leq 2a_i \Delta_i^2.$$

Applying Lemma B.3 and (B.2) to h_i with $c_1 = b_i/4$ and $c_2 = 4b_i$ gives

$$2^{-16p} b_i^p \left(x_i^{2p-2}\Delta_i^2 + \Delta_i^{2p}\right) \le h_i(x_i+\Delta_i)^p - h_i(x_i)^p - p \cdot h_i(x_i)^{p-1} h_i'(x_i) \tag{D.4}$$

$$\le 2^{6p} b_i^p \left(x_i^{2p-2}\Delta_i^2 + \Delta_i^{2p}\right). \tag{D.5}$$

Adding (D.3), (D.4), (D.5) for all i and using our choice of g, r in Line 6 and Line 5 gives that for Δ

$$2^{-16p} \left(\sum_{i=1}^m r_i \Delta_i^2 + \sum_{i=1}^m b_i^p \Delta_i^{2p}\right) \le \mathrm{val}_{p,W}(x+\Delta) - \mathrm{val}_{p,W}(x) - g^T\Delta \tag{D.6}$$

$$\le 2^{6p} \left(\sum_{i=1}^m r_i \Delta_i^2 + \sum_{i=1}^m b_i^p \Delta_i^{2p}\right) \tag{D.7}$$

Applying Lemma B.4 to (D.6), (D.7) gives us that

$$\mathrm{val}_{p,W}(x + 2^{-22p}\Delta) - OPT_{p,W} \le (1 - 2^{-22p}) \left(\mathrm{val}_{p,W}(x) - OPT_{p,W}\right) + \frac{1}{2^{\mathrm{poly}(\log m)}}.$$

As $\mathrm{val}_{p,W}(x) \le 2^{\mathrm{poly}(\log m)}$ initially, performing this iteration $\widetilde{O}(2^{22p})$ times as in line 2 results in $\mathrm{val}_{p,W}(x) \le OPT_{p,W} + \frac{1}{2^{\mathrm{poly}(\log m)}}$ at the end, as desired. ∎

Now, the proof of Theorem 6.3 follows from verifying the conditions of Lemmas D.1 and D.2 and applying Theorem 6.2.

Proof of Theorem 6.3. We check the conditions of Lemmas D.1 and D.2. By Lemma D.1 of [27] all resistances and residual capacities are polynomially bounded, hence all quantities encountered during the algorithm are $2^{\mathrm{poly}(\log m)}$. For $A = B^T$, we have that $AA^T = B^T B$ is a graph Laplacian, and hence has polynomially bounded singular values. We may compute an initial point $x = B(B^T B)^\dagger d$ as in Line 1 by computing an electric flow. All remaining conditions follow directly for the choice $b_i = 1$.

We now analyze the runtime. We set $b_i = 1$ to apply Lemmas D.1 and D.2 to Theorem 6.3. Theorem 6.2 allows us to solve objectives desired by ORACLE2P(B^T, g, r, b) in $m^{1+o(1)}$ time for $b = 1$ and $p = 2\left\lceil \sqrt{\log m} \right\rceil$. Additionally, $\widetilde{O}(2^{22p}) = m^{o(1)}$ for $p = 2\left\lceil \sqrt{\log m}\right\rceil$, hence the total runtime is $m^{1+o(1)}$ as desired. ∎