Unit Capacity Maxflow in Almost $O(m^{4/3})$ Time

Tarun Kathuria
EECS
University of California Berkeley
Berkeley, USA
tarunkathuria@berkeley.edu

Yang P. Liu
Department of Mathematics
Stanford University
Stanford, USA
yangpliu@stanford.edu

Aaron Sidford
MS&E
Stanford University
Stanford, USA
sidford@stanford.edu

***Abstract*—We present an algorithm, which given any m-edge n-vertex directed graph with positive integer capacities at most U computes a maximum s-t flow for any vertices s and t in $O(m^{4/3+o(1)}U^{1/3})$ time. This improves upon the previous best running times of $O(m^{11/8+o(1)}U^{1/4})$ [1], $\widetilde{O}(m\sqrt{n}\log U)$ [2] and $O(mn)$ [3] when the graph is not too dense and doesn't have large capacities. We build upon advances for sparse maxflow based on interior point methods [1], [4], [5]. Whereas these methods increase the energy of local ℓ_2-norm minimizing electrical flows, we instead increase the Bregman divergence value of flows which minimize the Bregman divergence with respect to a weighted log barrier. This allows us to trace the central path with progress depending only on ℓ_∞ norm bounds on the congestion vector as opposed to the ℓ_4 norm, which arises in these prior works. Further, we show that smoothed ℓ_2-ℓ_p flows [6], [7] which were used to maximize energy [1] can also be used to efficiently maximize divergence, thereby yielding our desired runtimes. We believe our approach towards Bregman divergences of barriers may be of further interest.**

***Keywords*-Maximum flow, interior point method, bipartite matching, optimization**

This is an extended abstract corresponding to a a merge of the full paper arxiv.org/abs/2003.08929 [8] and the full paper arxiv.org/abs/2009.03260 [9].

I. Introduction

The s-t maximum flow problem and its dual, the s-t minimum cut on graphs are among the most fundamental problems in combinatorial optimization and have a wide range of applications. They serve as a testbed for new algorithmic concepts which have found uses in other areas of theoretical computer science and optimization. In the well-known s-t maximum flow problem we are given a graph $G = (V, E)$ with m edges and n vertices with positive integer edge capacities $1 \leq u_e \leq U$, and aim to route as much flow as possible from s to t while restricting the magnitude of the flow on each edge to its capacity.

Decades of work in combinatorial algorithms for this problem led to a large set of results culminating in the work of Goldberg-Rao [10] which achieves a $\widetilde{O}(m\min\{m^{1/2}, n^{2/3}\}\log U)$ time algorithm for directed maxflow.[1] This bound remained unimproved for many years, even in the special case of $U = 1$, where these runtimes were obtained much earlier [11], [12].

Continuous Optimization for Maxflow: In a breakthrough paper Christiano *et al.* [13] showed how to compute ε-approximate maxflows in undirected graphs in $\widetilde{O}(mn^{1/3}\log(U)\mathsf{poly}(1/\varepsilon))$.[2] [3] Their new approach used the result that Laplacian linear systems could be solved in nearly linear time [18] to "augment" by electric flows, which minimize the weighted ℓ_2 norm of the flow vector with respect to edge resistances. A straightforward analysis relating the ℓ_2 and ℓ_∞ norm of the congestion vector, i.e., the vector of ratios of the electrical flow on an edge over a type of "residual capacity" of the edge, yields an $O(\sqrt{m})$ iteration algorithm. However, they showed that perturbing edge weights and leverging a potential function based on the energy of the electrical flow, i.e. its weighted ℓ_2 norm, yields an $O(m^{1/3})$ iteration algorithm. This result inspired multiple advances in solving flow problems (See Section I-C).

In the setting of exact directed maxflow, the only improvements to the $O(m\min\{m^{1/2}, n^{2/3}\})$ runtimes of [11], [12] when $U = 1$, are based on interior point methods (IPMs) (with the exception of [19]). IPMs are iterative methods which reduce linear programming to solving a sequence of linear systems. In graphs, for standard IPMs (e.g. [20]) these linear systems are Laplacians and IPMs directly yield an $\widetilde{O}(m^{3/2}\log(U))$ time algorithm for directed maxflow [21]. Building upon [13], Mądry [4] developed a weighted-barrier based IPM which, when $U = 1$, could solve maxflow in $\widetilde{O}(m^{10/7})$. Whereas [21] leverages an ℓ_2 analysis of a log barrier, [5] leveraged that it is possible to analyze progress based on the ℓ_4 norm of the congestion vector of the current electric flow induced by the Laplacian system. Combining this with the perturbation of weights in the weighted log barrier, tracking the perturbation of energy of the electrical flow based of these weight perturbations, and additional

[1]There is a sequence of works resulting in a $O(mn)$ time algorithm for *strongly polynomial time* maxflow [3] – we will not address these here.

[2]There were several earlier combinatorial algorithms giving improvements in the case of bounded flow values and ε-approximate maxflow on undirected graphs [14]–[17], though none achieved a better than $n^{3/2}$ runtime on all sparse graphs.

[3]Several algorithms discussed here, including the original algorithm of [13], are randomized. We will not distinguish randomized versus deterministic algorithms in this introduction.

2575-8454/20/$31.00 ©2020 IEEE
DOI 10.1109/FOCS46700.2020.00020

insights lead to [4] which was improved by [5] to obtain an $\widetilde{O}(m^{10/7}U^{1/7})$ time maxflow algorithm.

Even more recently, Liu and Sidford [1] improved the runtime of Mądry [5] and 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 hence can be solved in almost-linear time using the work of [6], [7]. This leads to a $O(m^{11/8+o(1)}U^{1/4})$-time algorithm for maxflow. However, this result still relied on the amount of progress one can take in each iteration being limited to the bounds one can ensure on the ℓ_4 norm of the congestion of an electrical flow vector, as opposed to the ideal ℓ_∞ norm of the congestion vector which governed [13].[4]

We remark here that there are IPMs for linear programming which only measure centrality in ℓ_∞ norm as opposed to the ℓ_2 or ℓ_4 norm. In particular [23], [24] 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 algorithms for maxflow and this paper takes a different approach based on measuring progress based on the ℓ_∞ norm of the congestion vector.

A. Our Contributions

Building on the past decade of advances to maxflow, this paper obtains an $O(m^{4/3+o(1)})$ maxflow algorithm on sparse, unit-capacity graphs, thereby closing the longstanding gap between the runtime achieved for approximately solving maxflow using Laplacian system solvers in [13] and the best runtime known for solving maxflow on directed graphs. We also shed light on the energy maximization framework that underlie all previous $O(n^{3/2-\Omega(1)})$ results for exact directed maxflow and depart from it to achieve our bounds. Energy maximization, though straightforward to analyze, is somewhat mysterious from an optimization perspective - it is unclear what (if any) standard optimization technique it corresponds to.[5] In this paper, we show that energy maximization arises naturally when locally optimizing the second order Taylor approximation of a Bregman divergence with respect to a logarithmic barrier. We then show that by optimizing the entire function, instead of its second order Taylor approximation, we can obtain improved convergence. We believe this divergence maximization technique is of intrinsic interest.

[4]Technically, in [1] and [22], weight changes are computed to reduce the ℓ_∞ norm of congestion of an electric flow vector. However, the centrality depends on the ℓ_4 norm, and we see this as why previous works achieved worse than $O(m^{4/3})$ runtime. Since the initial version of this paper [8] was released, [22] was updated to leverage the techniques of this paper and achieved a $O(m^{4/3+o(1)})$ runtime for a broader range of problems.

[5]The recent, independent paper of [22] gives an alternative perspective on energy maximization as regularized Newton steps.

Finally, to achieve our result, we show that divergence maximization can be performed efficiently for graphs. Whereas [1] showed that energy maximization could be performed efficiently by solving smoothed ℓ_2-ℓ_p flows of [6], [7], here we need to solve problems not immediately representable this way. Nevertheless, we show previous solvers can be applied to solve a quadratic extension of the divergence maximization problem in Lemma V.8, which suffices for our algorithms. More generally, we open up the algorithms which underlie ℓ_2-ℓ_p flow solvers and show in Theorem 2 that a range of undirected flow optimization problems, including divergence maximization, can be solved efficiently. We hope this generalized flow solving routine may find further use as well.

B. Our Results

The main result of this paper is the following theorem.

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

Theorem 1 yields an exact maxflow runtime matching the $\widetilde{O}(mn^{1/3}\varepsilon^{-11/3})$ runtime of [13] for $(1-\varepsilon)$-approximate undirected maxflow on sparse graphs. Further, this improves on the recent $m^{11/8+o(1)}U^{1/4}$ time algorithm of [1] when $U \leq m^{1/2-\varepsilon}$ for some $\varepsilon > 0$. When $U \geq m^{1/2}$, the algorithms of Theorem 1 and [1], [4], [5] all have runtime $\widetilde{O}(m^{3/2})$, which was known by [10]. Hence, we assume $U \leq \sqrt{m}$ throughout the paper. An immediate corollary of Theorem 1 is the following result on bipartite matching.

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

We note that Theorem 1 and Corollary I.1 are deterministic. This is achieved by leveraging [26], which recently showed that algorithms for many flow algorithms, including Laplacian system solvers [18], smoothed ℓ_2-ℓ_p flow computation methods [6], [7] and some maxflow IPMs [4], [5], may be derandomized with a $m^{o(1)}$ runtime increase.

C. Previous work

Approximate undirected maxflow: An extensive line of work exploiting continuous optimization techniques for faster maxflow algorithms was inspired by [13]. Lee *et al.* [27] also presented a $O(n^{1/3}\mathsf{poly}(1/\varepsilon))$ iteration algorithm for unit-capacity graphs also using electrical flows. Further, Kelner *et al.* [28] and Sherman [29] presented algorithms achieving $O(m^{1+o(1)}\mathsf{poly}(1/\varepsilon))$ runtimes for maxflow and its variants, [30] improved this to nearly linear time for maximum flow, and [19], [31] further improved the ϵ dependence.

IPMs for maxflow: In order to obtain highly accurate solutions and improved runtimes for directed maxflow, recent work has leveraged IPMs for linear programming [20], [32]. As discussed, classic IPMs [21] and nearly linear time Laplacian systems solvers [18] 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 [1], [4], [5]. Beyond these results, Lee and Sidford [2] also devised a faster IPM using weighted barriers to achieve a $\widetilde{O}(m\sqrt{n}\log(U/\varepsilon))$-runtime for maximum flow. Further, [33] achieved a runtime for minimum cost flow matching the runtimes of [4], [5] and this was recently improved by [22], leveraging the techniques of this paper. Very recently, [34] also provided an IPM-based algorithm which yields an $\widetilde{O}(m+n^{3/2})$ time algorithm for bipartite matching and similar runtimes for related problems.

ℓ_p-flows and beyond: Closely related to this paper, and the preceding lines of research, 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 maximum flow problems $p=\infty$, and transshipment problems $p=1$. Bubeck *et al.* [35] note that although standard self-concordance theory for IPMs cannot improve upon $\sqrt{m}$ iterations for $p\neq 2$, an alternative *homotopy-based* method can solve the problem in $\widetilde{O}_p(m^{\frac{1}{2}-\frac{1}{p}}\log(1/\varepsilon))$ iterations, where O_p hides dependencies on p in the runtime. This lead to improvements on the runtime for constant values of p. Next, Adil *et al.* [36] provided an ℓ_p flow algorithm which reduced the problem to approximately solving a sequence of smoothed ℓ_2-ℓ_p flows, i.e. objectives with both ℓ_2 resistances and ℓ_p weights, and applied a variant of [13] to solve it. This lead to a $\widetilde{O}_p(m^{1+\frac{p-2}{3p-2}}\log^2(1/\varepsilon))$-time algorithm, which is $O_p(m^{4/3}\log^2(1/\varepsilon))$ even for large p. Further, Kyng *et al.* [6] leveraged the tools from Spielman and Teng [18] for ℓ_2-norm flow problems to show 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. Although, these results do not immediately lead to faster directed maxflow algorithm, as naively this would require $p=\mathrm{poly}(m)$, both the maxflow algorithm of this paper and [1] leverage the almost linear time ℓ_2-ℓ_p-solver, for $p=O(\sqrt{\log m})$ of [6], [7].

II. 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 $\|v\|_p \stackrel{\text{def}}{=} (\sum_{i=1}^m |v_i|^p)^{1/p}$, and $\|v\|_\infty \stackrel{\text{def}}{=} \max_{i=1}^m |v_i|$. For $n\times n$ positive semidefinite matrices M_1, M_2 we write $M_1 \approx_r M_2$ for $r\geq 1$ if $r^{-1}x^TM_1x \leq x^TM_2x \leq rx^TM_1x$ for all $x\in\mathbb{R}^n$. For a differentiable function $f:\mathbb{R}^n\to\mathbb{R}$ we define its induced *Bregman divergence* as $D_f(x\|y) \stackrel{\text{def}}{=} f(x)-f(y)-\nabla f(y)^T(y-x)$ 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 $\chi_{ab}=1_b-1_a$, a vector with a 1 in position a and -1 in position b. 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 an $m\times n$ matrix B, where the row corresponding to edge $e=(u,v)$ has a 1 (respectively -1) in the column corresponding to v (respectively u). Note that $f\in\mathbb{R}^E$ is a $t\chi$-flow if and only if $B^Tf=t\chi$. More broadly, we call any $d\in\mathbb{R}^V$ a demand vector if $d\perp 1$ and we say $f\in\mathbb{R}^E$ routes d if $B^Tf=d$.

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* 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.

III. Algorithm Derivation and Motivation

In this section we present a principled approach for deriving our new method and the previous energy-based methods [1], [4], [5] for maxflow from an IPM setup.

A. Interior Point Method Setup

The starting point for our method is the broad IPM framework for maxflow of [1], which in turn was broadly inspired by [5]. We consider the setup described in Section II 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 $B^Tf=t\chi$ and f is a high accuracy approximation of the solution $f^*_{t,w}$ of the following optimization problem:

$$f^*_{t,w} \stackrel{\text{def}}{=} \operatorname*{arg\,min}_{B^Tf=t\chi} V(f) \tag{1}$$

where

$$V(f) \stackrel{\text{def}}{=} -\sum_{e \in E} \left(w_e^+ \log(u_e^+ - f_e) + w_e^- \log(u_e^- + f_e) \right)$$

Here, $V(f)$ is known as the *weighted logarithmic barrier* and penalizes how close f is to violating the capacity constraints and t is the amount of flow sent from a to b.

Generally, IPMs proceed towards optimal solutions by iteratively improving the quality, i.e. increasing the parameter t, and decreasing the proximity to the constraints, i.e. decreasing $V(f)$. Previous maxflow IPMs [1], [2], [4], [5] all follow this template. Specifically, [1], [5] alternate between Newton step to improve the optimality of f for Eq. (1) (called *centering steps*) and computing a new flow and weights to approximately solve Eq. (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 preprocessing step known as preconditioning (Section IV-A) directly yields to an $\widetilde{O}(m^{3/2})$ time algorithm. Recent advances [1], [2], [4], [5] were achieved by performing further work to modify the weights and flows used.

B. 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. 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 Eq. (1) give that the gradient of V at the optimum $f^*_{t,w}$ of Eq. (1) is perpendicular to the kernel of B^T, so there is a vector y with $By = \nabla V(f^*_{t,w})$. Define

$$\begin{aligned} \widehat{f} &\stackrel{\text{def}}{=} \arg\min_{B^T f = \delta\chi} D_V(f^*_{t,w} + f \| f^*_{t,w}) \\ &= \arg\min_{B^T f = \delta\chi} V(f^*_{t,w} + f) - V(f^*_{t,w}) - \nabla V(f^*_{t,w})^T f, \end{aligned} \tag{2}$$

i.e. the $\delta\chi$-flow with smallest divergence from $f^*_{t,w}$ against the barrier V. Again, optimality conditions give 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

$$\arg\min_{B^T f = \delta\chi} \frac{1}{2} f^T \nabla^2 V(f^*_{t,w}) f \tag{3}$$

$$= \delta \nabla^2 V(f^*_{t,w})^{-1} B (B^T \nabla^2 V(f^*_{t,w})^{-1} B)^\dagger \chi \, . \tag{4}$$

This can be computed in $O(m)$ time plus the time to solve a Laplacian system, i.e. $\widetilde{O}(m)$ [18]. Choosing δ that routes $\Omega(m^{-1/2})$ fraction of the remaining flow, adding the flow in Eq. (4) to our current point, and taking further Newton steps to re-center yields an $\widetilde{O}(m^{3/2})$ time maxflow algorithm.

C. Energy-based improvements

Improvements to the above $\widetilde{O}(m^{3/2})$ time algorithm [1], [4], [5] arise by a more careful analysis of the step size δ of the Newton step that we may take such that recentering may still be performed in $\widetilde{O}(m)$ time, and by leveraging that the flow in Eq. (4) 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 Section IV-A). 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 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 [4], [5] achieves this by boosting 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 [5], [1] 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:

$$\max_{\|r'\|_1 \le W} \min_{B^T f = \delta\chi} \sum_{e \in E} (r_e + r'_e) f_e^2 . \tag{5}$$

[1] showed that a smoothed version of this objective was solvable in $m^{1+o(1)}$ time using smoothed ℓ_2-ℓ_p flows [6], and that the combinatorial edge boosting framework of [4], [5] can essentially be viewed as greedily taking gradient steps against the objective in Eq. (5).

D. Our new method: beyond electric energy

A disappointing feature of the above discussion is that while the ℓ_∞ norm of congestion governs the amount of flow we may add and still have a feasible flow, we are forced to control the ℓ_4 norm of congestion of the electric flow to allow for efficient centering. This is due to the fact that although the step can be taken without violating 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 Eq. (2). As a result, it suffices to compute weight changes to minimize the ℓ_∞ norm of congestion of this flow.

The next challenge 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 Section IV-B.

Finally, we must compute weights that reduce the ℓ_∞ norm of congestion of the divergence minimizing flow. As the approach of [1] 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 [5] and energy maximization framework of [1] to nonlinear objectives, such as the quadratic extensions described above, and then generalizing the iterative refinement framework introduced by [6], [36] 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.

IV. Technical Ingredients

In this section, we elaborate on several technical aspects discussed in Section III. We give details for setting up the IPM in Section IV-A, discuss preconditioning in Section IV-A, elaborate on quadratic extensions in Section IV-B, and discuss iterative refinement in Section IV-C.

A. 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 (see [4], [37] or [1] 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 Eq. (1) give that the gradient of V at the optimum $f^*_{t,w}$ of Eq. (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

$$B^T f = t\chi \quad \text{and} \tag{6}$$

$$[By]_e = [\nabla V(f)]_e = \frac{w_e^+}{u_e^+ - f_e} - \frac{w_e^-}{u_e^- + f_e} \text{ for all } e \in E \tag{7}$$

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 [1], [4], [5] alternated between progress steps and centering steps. Progress steps increase the path parameter t at the cost of centrality, which refers to the distance of f from satisfying Eq. (7) in the inverse norm of the Hessian of $V(f)$ – see [1] 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 goal of the IPM is to reduce the value of the residual flow $F_t = t^* - t$ below a threshold, at which point we may round and use augmenting paths [27]. 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. The following lemma was shown in [1] Section B.5.

Lemma IV.1. *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 IV.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.

B. 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 given that for all $e \in E$ that $c_e^+ \approx_{1.05} (c_e^+)'$ and $c_e^- \approx_{1.05} (c_e^-)'$. 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

$$\begin{aligned}\mathcal{B}(f) &\stackrel{\text{def}}{=} V(f + f^*_{t,w}) - V(f^*_{t,w}) \\ &= -\sum_{e\in E}\left(w_e^+ \log\left(1 - \frac{f_e}{u_e^+ - [f^*_{t,w}]_e}\right)\right. \qquad (8) \\ &\qquad \left. + w_e^- \log\left(1 + \frac{f_e}{u_e^- + [f^*_{t,w}]_e}\right)\right), \qquad (9)\end{aligned}$$

so that $\widehat{f} = \arg\min_{B^T f = \delta\chi} \mathcal{B}(f)$. To compute $\widehat{f}$, we smooth Eq. (9) by replacing each instance of $\log(\cdot)$ with a function $\widetilde{\log}(\cdot)$ defined as

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

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

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

We now claim that $\widehat{f} = \arg\min_{B^T f = \delta\chi} \widetilde{\mathcal{B}}(f)$, 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 $\arg\min_{B^T f = \delta\chi} \widetilde{\mathcal{B}}(f)$ 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 $\widehat{f} = \arg\min_{B^T f = \delta\chi} \widetilde{\mathcal{B}}(f)$ by the following simple observation.

Observation 1. [35] Let $\chi \subseteq \mathbb{R}^n$ be a convex set, and let $f, g : \mathbb{R}^n \to \mathbb{R}$ be convex functions. Let $x^* = \arg\min_{x\in\chi} f(x)$, and assume that f, g agree on a neighborhood of x^* in $\mathbb{R}^n$. Then $g(x^*) = \min_{x\in\chi} g(x)$.

We emphasize that we do not directly do the procedure described here, and instead apply quadratic smoothing in a different form in Section V. 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 Eqs. (11) and (12).

C. Iterative refinement

The idea of *iterative refinement* was introduced in [6], [36] 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, we can show that for a fixed flow f there is a gradient vector g such that for any circulation Δ we have

$$\|f + \Delta\|_p^p - \|f\|_p^p = g^T\Delta + \Theta_p\left(\sum_{e\in E} |f_e|^{p-2}\Delta_e^2 + \|\Delta\|_p^p\right), \qquad (10)$$

so that approximately minimizing Eq. (10) over circulations Δ suffices to make multiplicative progress towards the optimum of $\min_{B^T f = d} \|f\|_p^p$. We sketch in Section V-A how to reduce general class of objectives 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 Eq. (10). We then combine this expansion with the almost linear time smoothed ℓ_2-ℓ_p solver of [6] to solve the necessary objectives.

V. Efficient Divergence Maximization

In this section we present our interior point framework and show Theorem 1. Throughout the section we set $\eta = \log_m\left(m^{1/6-o(1)}U^{1/3}\right)$, in pursuit of a $m^{3/2-\eta+o(1)}$ time algorithm. We assume G is preconditioned Section IV-A and we maintain the invariant $\|w\|_1 \le 5m/2$ before each step, and $\|w\|_1 \le 3m$ at all times. As our algorithm runs in time at least $m^{3/2}$ for $U > \sqrt{m}$ we assume $U \le \sqrt{m}$ throughout this section. We assume that our algorithm knows F_t, as discussed in Section IV-A, and that $F_t \ge m^{1/2-\eta}$ or else we can round to an integral flow and run augmenting paths.

Notational conventions: We largely adopt the same notation as used in Section IV-B. We use D to refer to functions of the logarithmic barrier, 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 $\varepsilon < 1$ define the functions

$$D(x) \stackrel{\text{def}}{=} -\log(1-x) - x \quad \text{and} \tag{11}$$

$$\widetilde{D}_\varepsilon(x) \stackrel{\text{def}}{=} \begin{cases} D(x) \text{ for } |x| \le \varepsilon \\ D(\varepsilon) + D'(\varepsilon)(x-\varepsilon) + \frac{D''(\varepsilon)}{2}(x-\varepsilon)^2 \\ \text{ for } x \ge \varepsilon \\ D(-\varepsilon) + D'(-\varepsilon)(x+\varepsilon) + \frac{D''(-\varepsilon)}{2}(x+\varepsilon)^2 \\ \text{ for } x \le -\varepsilon. \end{cases} \tag{12}$$

Throughout, we will omit ε and write $\widetilde{D}(x) \stackrel{\text{def}}{=} \widetilde{D}_{1/10}(x)$. As discussed in Section IV-B $\widetilde{D}(x)$ is such that it behaves as a quadratic around $x = 0$, and has continuous second derivatives. 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 Section IV-B. Several useful properties of the derivatives and stability of D and $\widetilde{D}$ are collected in Lemma V.1, which we prove in the full version.

Lemma V.1 (Properties of $\widetilde{D}$)**.** *We have that* $1/2 \le \widetilde{D}''(x) \le 2$*. 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 higher order 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.

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

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

Define val and $\widetilde{\text{val}}$ as follows, where $p = 2\lceil\sqrt{\log m}\rceil$, and $W = m^{6\eta}$ is a constant. For clarity, we express the vector inside the $\|\cdot\|_p$ piece of Eqs. (13) and (14) coordinate-wise, where the coordinate corresponding to edge e is written.

$$\text{val}(f) \stackrel{\text{def}}{=} D_w^V(f) + W \left\| (c_e^+)^2 \left(D\left(\frac{f_e}{c_e^+}\right) + \left(\frac{c_e^-}{c_e^+}\right) D\left(-\frac{f_e}{c_e^-}\right)\right)\right\|_p \tag{13}$$

$$\widetilde{\text{val}}(f) \stackrel{\text{def}}{=} \widetilde{D_w^V}(f) + W \left\| (c_e^+)^2 \left(\widetilde{D}\left(\frac{f_e}{c_e^+}\right) + \left(\frac{c_e^-}{c_e^+}\right) \widetilde{D}\left(-\frac{f_e}{c_e^-}\right)\right)\right\|_p \tag{14}$$

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

$$\begin{aligned} &\max_{\substack{\|(c_e^+)^{-2}\nu_e^+\|_q \le W \\ \nu \in \mathbb{R}^{2E}_{\ge 0} \\ \frac{\nu_e^+}{c_e^+} = \frac{\nu_e^-}{c_e^-} \text{ for all } e \in E}} \min_{B^T f = d} D_{w+\nu}^V(f) \\ &= \min_{B^T f = d} \max_{\substack{\|(c_e^+)^{-2}\nu_e^+\|_q \le W \\ \nu \in \mathbb{R}^{2E}_{\ge 0} \\ \frac{\nu_e^+}{c_e^+} = \frac{\nu_e^-}{c_e^-} \text{ for all } e \in E}} D_{w+\nu}^V(f) \\ &= \min_{B^T f = d} \max_{\substack{\|(c_e^+)^{-2}\nu_e^+\|_q \le W \\ \nu \in \mathbb{R}^{2E}_{\ge 0} \\ \frac{\nu_e^+}{c_e^+} = \frac{\nu_e^-}{c_e^-} \text{ for all } e \in E}} D_w^V(f) + D_\nu^V(f) \\ &= \min_{B^T f = d} \max_{\substack{\|(c_e^+)^{-2}\nu_e^+\|_q \le W \\ \nu^+ \in \mathbb{R}^{2E}_{\ge 0}}} D_w^V(f) \\ &+ \sum_{e \in E} \left(\nu_e^+ D\left(\frac{f_e}{c_e^+}\right) + \frac{c_e^-}{c_e^+} \nu_e^+ D\left(-\frac{f_e}{c_e^-}\right)\right) \\ &= \min_{B^T f = d} \text{val}(f) \end{aligned} \tag{15}$$

and similarly for $\widetilde{D_w^V}(f)$ and $\widetilde{\text{val}}(f)$. 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, 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 $\widetilde{\text{val}}(f)$ is ensured to have approximately unit weights on the f_e. Precisely, by Lemma V.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 in order to apply the smoothed ℓ_2-ℓ_p flow solvers in Theorem 3.

As $\min_{B^T f = d} \text{val}(f)$ 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 procedure to μ^*.

The remaining arguments in this section heavily use local optimality of convex functions. For this reason, we note that $\text{val}(f)$ and $\widetilde{\text{val}}(f)$ are convex, proof given in the full version.

Lemma V.2. $\text{val}(f)$ *and* $\widetilde{\text{val}}(f)$ *are convex.*

From now on, we fix a step size $\delta = \frac{F_t}{10^5 m^{1/2-\eta}}$. 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 $\widetilde{D_w^V}(f)$ 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 V.3 and V.4 generalize

corresponding bounds for electric flows shown in [5] and [1] Lemma 4.5 and 5.2.

Lemma V.3. *Let* $\delta = \frac{F_t}{10^5 m^{1/2-\eta}}$. *Then* $\min_{B^T f=\delta\chi} \widetilde{D_w^V}(f) \le 5 \cdot 10^{-7} m^{2\eta}$.

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 IV.1 and the invariant $\|w\|_1 \le 3m$ tells us that

$$\frac{f'_{e_p}}{c_{e_p}} \le \frac{\delta/m}{F_t/7\|w\|_1} \le \frac{21}{10^5 m^{1/2-\eta}} .$$

Therefore, applying Lemma V.1 to P, the set of m preconditioning edges, and again applying that $\|w\|_1 \le 3m$ gives the desired bound, as

$$\begin{aligned}\widetilde{D_w^V}(f') &= \sum_{e\in P}\left(w_e^+ \widetilde{D}\left(\frac{f'_e}{c_e^+}\right) + w_e^- \widetilde{D}\left(-\frac{f'_e}{c_e^-}\right)\right)\\ &\le \left(\frac{f'_{e_p}}{c_{e_p}}\right)^2 \sum_{e\in P}\left(w_e^+ + w_e^-\right)\\ &\le \left(\frac{21}{10^5 m^{1/2-\eta}}\right)^2 \|w\|_1 \le 5\cdot 10^{-7} m^{2\eta}\end{aligned}$$

■

Lemma V.4. *Let* $\delta = \frac{F_t}{10^5 m^{1/2-\eta}}$, *and* $\widehat{f} = \arg\min_{B^T f=\delta\chi} \widetilde{D_w^V}(f)$. *Then* $|\widehat{f}_e| c_e^{-2} \le m^{2\eta}$.

Proof: Local optimality tells us that there is a vector $z \in \mathbb{R}^V$ satisfying $Bz = \nabla \widetilde{D_w^V}(\widehat{f})$. This, Lemma V.3, and Lemma V.1, specifically that $x\widetilde{D}'(x) \le 2x^2 \le 8\widetilde{D}(x)$ gives

$$\begin{aligned}\delta\chi^T z = \widehat{f}^T B z &= \widehat{f}^T \nabla \widetilde{D_w^V}(\widehat{f})\\ &= \sum_{e\in E} \widehat{f}_e \left(\frac{w_e^+}{c_e^+}\widetilde{D}'\left(\frac{\widehat{f}_e}{c_e^+}\right) - \frac{w_e^-}{c_e^-}\widetilde{D}'\left(-\frac{\widehat{f}_e}{c_e^-}\right)\right)\\ &\le 8\widetilde{D_w^V}(\widehat{f}).\end{aligned}$$

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, as decreasing the flow along a cycle reduces the objective value. 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 V.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 gives

$$|\widehat{f}_e| c_e^{-2} \le 2|[Bz]_e| \le 2\chi^T z \le 16\delta^{-1}\widetilde{D_w^V}(\widehat{f}) \le m^{2\eta}$$

after using Lemma V.3 and $\delta = \frac{F_t}{10^5 m^{1/2-\eta}} \ge 10^{-5}$. ■

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

Algorithm 1 AUGMENT$(G, w, F_t, f^*_{t,w})$. Takes a preconditioned undirected graph G with maximum capacity U, weights $w \in \mathbb{R}^{2E}_{\ge 1}$ with $\|w\|_1 \le 5m/2$, 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: $\eta \leftarrow \log_m(m^{1/6-o(1)}U^{-1/3})$
2: $\delta \leftarrow \frac{F_t}{10^5 m^{1/2-\eta}}$ ▷ Step size.
3: $c_e^+ \leftarrow u_e^+ - [f^*_{t,w}]_e, c_e^- \leftarrow u_e^- + [f^*_{t,w}]_e$. ▷ Residual capacities.
4: $W \leftarrow m^{6\eta}$ ▷ Weight budget.
5: $\widehat{f} \leftarrow \arg\min_{B^T f=\delta\chi} \widetilde{\mathrm{val}}(f)$ ▷ $\widetilde{\mathrm{val}}(f)$ implicitly depends on W, c_e^+, c_e^-.
6: $v \in \mathbb{R}^E$ 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 $e \in E$.
7: $\mu \in \mathbb{R}^{2E}_{\ge 0}$ 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.
8: Initialize $\nu \in \mathbb{R}^{2E}_{\ge 0}$. ▷ Reduced weight change, satisfies $\frac{\nu_e^+}{c_e^+ - \widehat{f}_e} - \frac{\nu_e^-}{c_e^- + \widehat{f}_e} = \frac{\mu_e^+}{c_e^+ - \widehat{f}_e} - \frac{\mu_e^-}{c_e^- + \widehat{f}_e}$
9: **for** $e \in E$ **do**
10: **if** $\frac{\mu_e^+}{c_e^+ - \widehat{f}_e} - \frac{\mu_e^-}{c_e^- + \widehat{f}_e} \ge 0$ **then**
11: $\nu_e^+ \leftarrow (c_e^+ - \widehat{f}_e)\left(\frac{\mu_e^+}{c_e^+ - \widehat{f}_e} - \frac{\mu_e^-}{c_e^- + \widehat{f}_e}\right), \nu_e^- \leftarrow 0$
12: **else**
13: $\nu_e^+ \leftarrow 0, \nu_e^- \leftarrow -(c_e^- + \widehat{f}_e)\left(\frac{\mu_e^+}{c_e^+ - \widehat{f}_e} - \frac{\mu_e^-}{c_e^- + \widehat{f}_e}\right)$
14: **end if**
15: **end forreturn** $(\delta, \widehat{f}, \nu)$.

Now, we analyze Algorithm 1. We start by showing that $\widehat{f} = \arg\min_{B^T f=\delta\chi} \widetilde{D_{w+\mu}^V}(f)$ for the weight change μ in line 7 of Algorithm 1, and that μ has bounded ℓ_1 norm. This essentially follows from duality in our setup in Eq. (15).

Lemma V.5. *Let parameters* $\eta, W, c_e^+, c_e^-, \delta$, *flow* $\widehat{f}$, *and weight change* μ *be defined as in Algorithm 1, and assume that* $F_t \ge m^{1/2-\eta}$. *Then we have that* $\|\mu\|_1 \le m/2$, $f^*_{t,w} = f^*_{t,w+\mu}$, *and* $\widehat{f} = \arg\min_{B^T f=\delta\chi} \widetilde{D_{w+\mu}^V}(f)$.

Proof: Let $v \in \mathbb{R}^E$ be the vector as defined in line 6 in Algorithm 1. By local optimality of $\widehat{f}$, we have that there is a vector z satisfying for all $e \in E$ that

$$\begin{aligned}[Bz]_e &= \left[\nabla \widetilde{\mathrm{val}}(\widehat{f})\right]_e\\ &= \left(\frac{w_e^+}{c_e^+} + \frac{v_e^{p-1}}{\|v\|_p^{p-1}} \cdot \frac{W(c_e^+)^2}{c_e^+}\right)\widetilde{D}'\left(\frac{\widehat{f}_e}{c_e^+}\right)\\ &- \left(\frac{w_e^-}{c_e^-} + \frac{v_e^{p-1}}{\|v\|_p^{p-1}} \cdot \frac{W c_e^+ c_e^-}{c_e^-}\right)\widetilde{D}'\left(-\frac{\widehat{f}_e}{c_e^-}\right).\end{aligned} \tag{16}$$

For clarity, we rewrite line 7 of Algorithm 1 here as

$$\mu_e^+ = W(c_e^+)^2 \cdot \frac{v_e^{p-1}}{\|v\|_p^{p-1}}$$
$$\text{and } \mu_e^- = \frac{c_e^-}{c_e^+}\mu_e^+ = Wc_e^+c_e^- \cdot \frac{v_e^{p-1}}{\|v\|_p^{p-1}}. \tag{17}$$

Note that $\mu_e^+/c_e^+ = \mu_e^-/c_e^-$, hence $f^*_{t,w} = f^*_{t,w+\mu}$ by Eq. (7). Combining Eqs. (16) and (17) and optimality conditions of the objective $\min_{B^Tf=\delta_\chi} \widetilde{D^V_{w+\mu}}(f)$ shows that $\widehat{f} = \arg\min_{B^Tf=\delta_\chi} \widetilde{D^V_{w+\mu}}(f)$. Additionally, if q is the dual of p, i.e. $1/q + 1/p = 1$, then

$$\|\mu\|_1 \le m^{o(1)}\|\mu\|_q \le 2m^{o(1)}WU^2 \left\| \frac{v_e^{p-1}}{\|v\|_p^{p-1}} \right\|_q$$
$$= 2m^{o(1)}WU^2 = m^{6\eta+o(1)}U^2 \le m/2$$

by our choice of $\eta = \log_m\left(m^{1/6-o(1)}U^{1/3}\right)$. ∎

We now show congestion bounds on $\widehat{f}$ by imitating the proof of Lemma V.3 and applying Lemma V.4. Recall that $c_e = \min(c_e^+, c_e^-)$.

Lemma V.6. *Let parameters $\eta, W, c_e^+, c_e^-, \delta$, flow $\widehat{f}$, and weight change μ be defined as in Algorithm 1, and assume that $F_t \ge m^{1/2-\eta}$. Then we have $|\widehat{f}_e| \le \frac{1}{500}m^{-2\eta}$ and $|\widehat{f}_e| \le \frac{1}{20}c_e$ for all edges e. It follows that $\widehat{f} = \arg\min_{B^Tf=\delta_\chi} \mathrm{val}(f)$.*

Proof: We first show $\widetilde{\mathrm{val}}(\widehat{f}) \le 10^{-6}m^{2\eta}$. Let f' be the flow which routes $\frac{\delta}{m}$ units of flow on each of the m preconditioning edges. As in Lemma V.3 we have that $\widetilde{D^V_w}(f') \le 5\cdot 10^{-7}m^{2\eta}$. For a preconditioning edge e, using Lemma V.1 and Lemma IV.1 gives that

$$(c_e^+)^2\left(\widetilde{D}\left(\frac{f'_e}{c_e^+}\right) + \left(\frac{c_e^-}{c_e^+}\right)\widetilde{D}\left(-\frac{f'_e}{c_e^-}\right)\right)$$
$$\le (c_e^+)^2\left(\left(\frac{f'_e}{c_e^+}\right)^2 + \left(\frac{c_e^-}{c_e^+}\right)\left(\frac{f'_e}{c_e^-}\right)^2\right) \le 2(f'_e)^2$$
$$\le 2(\delta/m)^2 \le 2m^{-2}\left(\frac{F_t}{10^5 m^{1/2-\eta}}\right)^2 \le 10^{-8}m^{2\eta-1}U^2.$$

as $F_t \le 3mU$. For the choice $W = m^{6\eta}$ we get that

$$\widetilde{\mathrm{val}}(f') \le \widetilde{D^V_w}(f') + W\left\|(c_e^+)^2\left(\widetilde{D}\left(\frac{f'_e}{c_e^+}\right) + \left(\frac{c_e^-}{c_e^+}\right)\widetilde{D}\left(-\frac{f'_e}{c_e^-}\right)\right)\right\|_p$$
$$\le 5\cdot 10^{-7}m^{2\eta} + 10^{-8}m^{8\eta-1+o(1)}U^2 \le 10^{-6}m^{2\eta}$$

where we have used $\|x\|_p \le m^{o(1)}\|x\|_\infty$ for the choice $p = 2\lceil\sqrt{\log m}\rceil$, and the choice of $\eta = \log_m\left(m^{1/6-o(1)}U^{1/3}\right)$ to get $m^{8\eta-1+o(1)}U^2 \le m^{2\eta}$.

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

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

Using the choice $W = m^{6\eta}$ and rearranging gives $|\widehat{f}_e| \le \frac{1}{500}m^{-2\eta}$.

Let μ be the weight increases given by line 7 of Algorithm 1, and Eq. (17). As $\widehat{f} = \min_{B^Tf=\delta_\chi} \widetilde{D^V_{w+\mu}}(f)$ and $\|w+\mu\|_1 \le 5m/2 + m/2 \le 3m$ by our invariant and Lemma V.5, using Lemma V.4 gives

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

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 that $\widehat{f}$ is also a minimizer of $\min_{B^Tf=\delta_\chi}\mathrm{val}(f)$ 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, the only instances which are known to admit almost linear runtimes [6], which we use in Section V-A.

Lemma V.7. *Let parameters $\eta, W, c_e^+, c_e^-, \delta$, flow $\widehat{f}$, and weight changes μ, ν be defined as in Algorithm 1, and assume that $F_t \ge m^{1/2-\eta}$. Then we have that $\|\nu\|_1 \le m^{4\eta+o(1)}U$ 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 V.6, we have $\widehat{f} = \arg\min_{B^Tf=\delta_\chi}\mathrm{val}(f)$. Let the vector v be defined as in line 6 of Algorithm 1. There exist vectors $y, z \in \mathbb{R}^E$ such that

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

Here, the first line follows from local optimality of $\widehat{f} = \arg\min_{B^Tf=\delta_\chi}\mathrm{val}(f)$, 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

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

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 8 to 13 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 Eq. (7) tell us that $f^*_{t+\delta,w+\nu} = f^*_{t+\delta,w+\mu}$.

We now bound $\|\nu\|_1$. Line 13 of Algorithm 1 and $\mu_e^+/c_e^+ = \mu_e^-/c_e^-$ gives that

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

where we have used that $c_e^{-1}|\widehat{f}_e| \le 1/20$. A similar analysis of line 11 gives that

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

In both cases, we have that $\nu_e^+ + \nu_e^- \le 3c_e^{-1}|\widehat{f}_e|(\mu_e^+ + \mu_e^-)$. Using the choice $W = m^{6\eta}$, Eq. (17) and Lemma V.6 yield

$$\|\nu\|_1 \le \sum_{e\in E} 3c_e^{-1}|\widehat{f}_e|(\mu_e^- + \mu_e^+)$$
$$\le 6W\sum_{e\in E}|\widehat{f}_e|c_e^- \cdot \frac{v_e^{p-1}}{\|v\|_p^{p-1}}$$
$$\le 12W \cdot \frac{1}{500}m^{-2\eta}U\left\|\frac{v_e^{p-1}}{\|v\|_p^{p-1}}\right\|_1$$
$$\le m^{4\eta+o(1)}U\left\|\frac{v_e^{p-1}}{\|v\|_p^{p-1}}\right\|_q = m^{4\eta+o(1)}U.$$

■

A. Efficient Divergence Maximization

Lemma V.7 shows that our algorithm just needs to compute $\arg\min_{B^T f=\delta\chi}\widetilde{\mathrm{val}}(f)$ in line 5 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 V.8. *There is an algorithm that in $m^{1+o(1)}$ time computes a flow f' with $B^T f' = \delta\chi$ and $\widetilde{\mathrm{val}}(f') \le \min_{B^T f=\delta\chi}\widetilde{\mathrm{val}}(f) + \frac{1}{2^{\mathrm{poly}(\log m)}}$.*

To prove Lemma V.8, we first show a generalization Theorem 2.

Theorem 2. *For graph $G = (V,E)$ and all $e \in E$, let $0 \le a_e \le 2^{\mathrm{poly}(\log m)}$ 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 $a_e/4 \le q_e''(x) \le 4a_e$ 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 $1/4 \le h_e''(x) \le 4$ for all $x \in \mathbb{R}$. For demand d 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 define*

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

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

We do this by carefully applying and extending the analysis of the following result of [6] on smoothed ℓ_2-ℓ_p flows.

Theorem 3 (Theorem 1.1 in [6], 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 $B^T f_0 = d$. For a flow f, define*

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

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

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

The remaining details of the proof of Lemma V.8 and Theorem 2 are deferred to the full version.

B. Algorithm

Here we state Algorithm 2 to show Theorem 1. It repeatedly takes steps computed with Algorithm 1, and when the remaining flow is $m^{1/2-\eta}$, we stop the algorithm, round to an integral flow and run augmenting paths.

Proof of Theorem 1: We show that MaxFlow(G) computes a maximum flow on G in time $m^{3/2-\eta+o(1)} = m^{4/3+o(1)}U^{1/3}$ by the choice of η. Correctness follows from Lemmas V.7 and V.8.

It suffices to control the weights. Our choice of δ guarantees that we route $\Omega(m^{-1/2+\eta})$ fraction of the remaining flow per iteration, hence line 3 executes $\widetilde{O}(m^{1/2-\eta})$ times.

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

1: $\eta \leftarrow \log_m(m^{1/6-o(1)}U^{-1/3})$.
2: $f \leftarrow 0, t \leftarrow 0, w \leftarrow 1$.
3: **while** $F_t \geq m^{1/2-\eta}$ **do**
4: $\quad(\delta, \widehat{f}, \nu) \leftarrow$ AUGMENT$(G, w, t^* - t, f)$.
5: $\quad f \leftarrow f + \widehat{f}, w \leftarrow w + \nu$, and $t \leftarrow t + \delta$.
6: **end while**
7: Round to an integral flow and use augmenting paths until done.

$\|\nu\|_1 \leq m^{4\eta+o(1)}U$ always by Lemma V.7, hence at the end of the algorithm by the choice of η we have

$$\|w\|_1 \leq 2m + \widetilde{O}\left(m^{1/2-\eta} \cdot m^{4\eta+o(1)}U\right) \leq 5m/2 \ .$$

To analyze the runtime, first note that by Lemma V.8 line 4 takes $m^{1+o(1)}$ time, so the total runtime of these throughout the algorithm is $m^{3/2-\eta+o(1)}$. Rounding to an integral flow takes $\widetilde{O}(m)$ time [4], [27]. Augmenting paths takes $O(m^{3/2-\eta})$ time also, as desired. ■

VI. 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 that the weights only increase and do not be become super linear and that 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 precisely computes the 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 have to depart from this template and e.g., backtrack on weight changes on the central path so that they are not additive throughout the algorithm, show better amortized bounds on weight change than shown here, or provide a novel algorithm to step to faraway points on the central path, outside a ball where the congestions are bounded.

Potential future directions: An interesting question is potentially achieving a $mn^{1/3+o(1)}$ time algorithm for maxflow through the approach of [2] and robust central paths [23], [24]. It would also be interesting to understand whether IPMs can achieve $m^{3/2-\Omega(1)}\log^{\widetilde{O}(1)}U$ runtimes for maxflow or understand whether $m^{4/3}$ barrier descriibed above can be broken.

ACKNOWLEDGMENT

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.

REFERENCES

[1] Y. P. Liu and A. Sidford, "Faster energy maximization for faster maximum flow," *STOC*, 2020.

[2] Y. T. Lee and A. Sidford, "Path finding methods for linear programming: Solving linear programs in õ(vrank) iterations and faster algorithms for maximum flow," in *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, 2014.

[3] 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. [Online]. Available: https://doi.org/10.1145/2488608.2488705

[4] 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.

[5] ——, "Computing maximum flow with augmenting electrical flows," in *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS*, 2016.

[6] 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.

[7] D. Adil and S. Sachdeva, "Faster p-norm minimizing flows, via smoothed q-norm problems," in *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, 2020.

[8] Y. Liu and A. Sidford, "Faster divergence maximization for faster maximum flow," in *arXiv preprints, 2003.08929*, 2020.

[9] T. Kathuria, "A potential reduction inspired algorithm for exact max flow in almost $\widetilde{O}(m^{4/3})$time," in *arXiv preprints, 2009.03260*, 2020.

[10] A. V. Goldberg and S. Rao, "Beyond the flow decomposition barrier," *J. ACM*, vol. 45, no. 5, pp. 783–797, 1998.

[11] A. V. Karzanov, "O nakhozhdenii maksimal'nogo potoka v setyakh spetsial'nogo vida i nekotorykh prilozheniyakh," *Mathematicheskie Voprosy Upravleniya Proizvodstvom*, 1973.

[12] S. Even and R. E. Tarjan, "Network flow and testing graph connectivity," *SIAM journal on computing*, vol. 4, no. 4, pp. 507–518, 1975.

[13] 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.

[14] 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.

[15] ——, "Better random sampling algorithms for flows in undirected graphs," in *SODA*, vol. 98, 1998, pp. 490–499.

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

[17] D. R. Karger, "Random sampling in cut, flow, and network design problems," in *Mathematics of Operations Research*, vol. 24(2), 1999, pp. 383–413.

[18] D. A. Spielman and S. Teng, "Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems," *SIAM J. Matrix Analysis Applications*, vol. 35, no. 3, pp. 835–885, 2014.

[19] 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. [Online]. Available: https://doi.org/10.1109/FOCS.2018.00091

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

[21] S. I. Daitch and D. A. Spielman, "Faster approximate lossy generalized flow via interior point algorithms," in *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, C. Dwork, Ed., 2008.

[22] K. Axiotis, A. Mądry, and A. Vladu, "Circulation control for faster minimum cost flow in unit-capacity graphs," *Arxiv preprints*, vol. abs/2003.04863, 2020.

[23] 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.

[24] J. van den Brand, Y. T. Lee, A. Sidford, and Z. Song, "Solving tall dense linear programs in nearly linear time," *STOC*, 2020.

[25] K. Anstreicher, "Potential reduction algorithms," 1996.

[26] 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," in *CoRR, abs/1910.08025*, 2019.

[27] 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*, 2013.

[28] 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.

[29] J. Sherman, "Nearly maximum flows in nearly linear time," in *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS*, 2013.

[30] 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*, R. Krauthgamer, Ed. SIAM, 2016, pp. 1862–1867. [Online]. Available: https://doi.org/10.1137/1.9781611974331.ch130

[31] 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.

[32] Y. E. Nesterov and A. Nemirovskii, *Interior-point polynomial algorithms in convex programming*, ser. Siam studies in applied mathematics. SIAM, 1994, vol. 13.

[33] M. B. Cohen, A. Mądry, P. Sankowski, and A. Vladu, "Negative-weight shortest paths and unit capacity minimum cost flow in $\tilde{o}$ ($m^{10/7}$ log W) time," in *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, 2017.

[34] 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*, pp. arXiv–2009, 2020.

[35] 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.

[36] D. Adil, R. Kyng, R. Peng, and S. Sachdeva, "Iterative refinement for lp-norm regression," in *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, 2019.

[37] H. Lin, "Reducing directed max flow to undirected max flow," in *Unpublished Manuscript, 4(2)*, 2009.