Almost-Linear-Time Weighted ℓ_p-Norm Solvers in Slightly Dense Graphs via Sparsification

Deeksha Adil
University of Toronto, Canada

Brian Bullins
Toyota Technological Institute at Chicago, IL, USA

Rasmus Kyng
ETH Zurich, Switzerland

Sushant Sachdeva
University of Toronto, Canada

Abstract

We give almost-linear-time algorithms for constructing sparsifiers with $n\,\mathrm{poly}(\log n)$ edges that approximately preserve weighted $(\ell_2^2 + \ell_p^p)$ flow or voltage objectives on graphs. For flow objectives, this is the first sparsifier construction for such mixed objectives beyond unit ℓ_p weights, and is based on expander decompositions. For voltage objectives, we give the first sparsifier construction for these objectives, which we build using graph spanners and leverage score sampling. Together with the iterative refinement framework of [Adil et al, SODA 2019], and a new multiplicative-weights based constant-approximation algorithm for mixed-objective flows or voltages, we show how to find $(1 + 2^{-\mathrm{poly}(\log n)})$ approximations for weighted ℓ_p-norm minimizing flows or voltages in $p(m^{1+o(1)} + n^{4/3+o(1)})$ time for $p = \omega(1)$, which is almost-linear for graphs that are slightly dense $(m \geq n^{4/3+o(1)})$.

2012 ACM Subject Classification Theory of computation → Sparsification and spanners; Theory of computation → Network flows

Keywords and phrases Weighted ℓ_p-norm, Sparsification, Spanners, Iterative Refinement

Digital Object Identifier 10.4230/LIPIcs.ICALP.2021.9

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2102.06977`

Funding *Deeksha Adil*: Supported by a Post Graduate Doctoral Scholarship awarded by NSERC (Natural Sciences and Engineering Research Council of Canada) and SS's Discovery Grant.
Sushant Sachdeva: Supported by a Discovery Grant awarded by NSERC (Natural Sciences and Engineering Research Council of Canada).

1 Introduction

Network flow problems are some of the most extensively studied problems in optimization (e.g. see [4, 37, 20]). A general network flow problem on a graph $G(V, E)$ with n vertices and m edges can be formulated as

$$\min_{\boldsymbol{B}^\top \boldsymbol{f} = \boldsymbol{d}} \mathtt{cost}(\boldsymbol{f}),$$

where $\boldsymbol{f} \in \mathbb{R}^E$ is a flow vector on edges satisfying net vertex demands $\boldsymbol{d} \in \mathbb{R}^V$, $\boldsymbol{B} \in \mathbb{R}^{E \times V}$ is the signed edge-vertex incidence matrix of the graph, and $\mathtt{cost}(\boldsymbol{f})$ is a cost measure on flows. The weighted ℓ_∞-minimizing flow problem, i.e., $\mathtt{cost}(\boldsymbol{f}) = \|\boldsymbol{S}^{-1}\boldsymbol{f}\|_\infty$, captures

the celebrated maximum-flow problem with capacities $\boldsymbol{S}$; the weighted ℓ_1-minimizing flow problem, $\texttt{cost}(\boldsymbol{f}) = \|\boldsymbol{Sf}\|_1$ captures the transshipment problem generalizing shortest paths with lengths $\boldsymbol{S}$; and $\texttt{cost}(\boldsymbol{f}) = \boldsymbol{f}^\top \boldsymbol{Rf} = \|\boldsymbol{R}^{\frac{1}{2}}\boldsymbol{f}\|_2^2$ captures the electrical flow problem [42].

Dual to flow problems are voltage problems, which can be formulated as

$$\min_{\boldsymbol{d}^\top \boldsymbol{v}=1} \texttt{cost}'(\boldsymbol{Bv}),$$

Analogous to the flow problems, picking $\texttt{cost}'(\boldsymbol{Bv}) = \|\boldsymbol{SBv}\|_1$ captures the capacitated min-cut problem, $\texttt{cost}'(\boldsymbol{Bv}) = \|\boldsymbol{S}^{-1}\boldsymbol{Bv}\|_\infty$ captures vertex-labeling [26], and $\texttt{cost}'(\boldsymbol{Bv}) = (\boldsymbol{Bv})^\top \boldsymbol{R}^{-1}\boldsymbol{Bv} = \|\boldsymbol{R}^{-\frac{1}{2}}\boldsymbol{Bv}\|_2^2$ captures the electrical voltages problem.

The seminal work of Spielman and Teng [42] gave the first nearly-linear-time algorithm for computing $(1 + 1/\operatorname{poly}(n))$-approximate solutions to electrical (weighted ℓ_2-minimizing) flow/voltage problems. This work spurred the "Laplacian Paradigm" for designing faster algorithms for several classic graph optimization problems including maximum flow [11, 38, 22], multi-commodity flow [22], bipartite matching [29], transshipment [39], and graph partitioning [32]; culminating in almost-linear-time or nearly-linear-time low-accuracy algorithms (i.e. $1+\varepsilon$ approximations with $\operatorname{poly}(\frac{1}{\varepsilon})$ running time dependence) for many of these problems.

Progress on high-accuracy algorithms (i.e. algorithms that return $(1 + 1/\operatorname{poly}(n))$-approximate solutions with only a $\operatorname{poly}(\log n)$ factor overhead in time) for solving these problems has been harder to come by, and for many flow problems has been based on interior point methods [18]. E.g. the best running time for maximum flow stands at $\widetilde{O}(\min(m\sqrt{n}, n^\omega + n^{2+1/6}))$ [27, 15] and $\widetilde{O}(m^{4/3})$ for unit-capacity graphs [29, 28, 21]. Other results making progress in this direction include works on shortest paths with small range negative weights [16], and matrix-scaling [13, 5]. Recently, there has been progress on the dense case. In [44], the authors developed an algorithm for weighted bipartite matching and transshipment running in $\widetilde{O}(m + n^{3/2})$ time. This is a nearly-linear-time algorithm in moderately dense graphs.

Bubeck et al. [9] restarted the study of faster high-accuracy algorithms for the weighted ℓ_p-norm objective, $\texttt{cost}(\boldsymbol{f}) = \|\boldsymbol{Sf}\|_p$, a natural intermediate objective between ℓ_2 and ℓ_∞. This result improved the running time significantly over classical interior point methods [31] for p close to 2. Adil et al. [1] gave a high-accuracy algorithm for computing ℓ_p-norm minimizing flows in time $\min\{m^{\frac{4}{3}+o(1)}, n^\omega\}$ for $p \in (2, \sqrt{\log n}]$. Building on their work, Kyng et al. [25] gave an almost-linear-time high-accuracy algorithm for *unit-weight* ℓ_p-norm minimizing flows $\texttt{cost}(\boldsymbol{f}) = \|\boldsymbol{f}\|_p^p$ for large $p \in (\omega(1), \sqrt{\log n}]$. More generally, they give an almost-linear time-high-accuracy algorithm for *mixed* $\ell_2^2 + \ell_p^p$ objectives as long as the ℓ_p^p-norm is unit-weight, i.e.,

$$\texttt{cost}(\boldsymbol{f}) = \|\boldsymbol{R}^{\frac{1}{2}}\boldsymbol{f}\|_2^2 + \|\boldsymbol{f}\|_p^p.$$

Their algorithm for $(\ell_2^2 + \ell_p^p)$-minimizing flows was subsequently used as a key ingredient in recent results improving the running time for high-accuracy/exact maximum flow on unit-capacity graphs to $m^{4/3+o(1)}$ [28, 21].

In this paper, we obtain a nearly-linear running time for weighted $\ell_2^2 + \ell_p^p$ flow/voltage problems on graphs. Our algorithm requires $p(m^{1+o(1)} + n^{4/3+o(1)})$ time for $p = \omega(1)$ which is almost-linear-time for $p \le m^{o(1)}$ in slightly dense graphs, $(m \ge n^{4/3+o(1)})$.

Our running time $m^{1+o(1)} + n^{4/3+o(1)}$ is even better than the $\widetilde{O}(m + n^{3/2})$ time obtained for bipartite matching in [44]. Our result beats the $\Omega(n^{3/2})$ barrier that arises in [44] from the use of interior point methods that maintain a vertex dual solution using dense updates across $\sqrt{n}$ iterations. The progress on bipartite matching relies on highly technical

graph-based inverse maintenance techniques that are tightly interwoven with interior point method analysis. In constrast, our sparsification methods provide a clean interface to iterative refinement, which makes our analysis much more simple and compact.

Graph Sparsification. Various notions of graph sparsification – replacing a dense graph with a sparse one, while approximately preserving some key properties of the dense graph – have been key ingredients in faster low-accuracy algorithms. Benczúr and Karger [8] defined cut-sparsifiers that approximately preserve all cuts, and used them to give faster low-accuracy approximation algorithms for maximum-flow. Since then, several notions of sparsification have been studied extensively and utilized for designing faster algorithms [33, 41, 35, 40, 30, 38, 22, 36, 17, 24, 19, 12].

Sparsification has had a smaller direct impact on the design of faster high-accuracy algorithms for graph problems, limited mostly to the design of linear system solvers [42, 23, 34, 24]. Kyng et al. [25] constructed sparsifiers for weighted ℓ_2^2 + unweighted ℓ_p^p-norm objectives for flows. In this paper, we develop almost-linear time algorithms for building sparsifiers for weighted $\ell_2^2 + \ell_p^p$ norm objectives for flows and voltages,

$$\texttt{cost}(\boldsymbol{f}) = \|\boldsymbol{R}^{\frac{1}{2}}\boldsymbol{f}\|_2^2 + \|\boldsymbol{S}\boldsymbol{f}\|_p^p, \text{ and } \texttt{cost}'(\boldsymbol{B}\boldsymbol{v}) = \|\boldsymbol{W}^{\frac{1}{2}}\boldsymbol{B}\boldsymbol{v}\|_2^2 + \|\boldsymbol{U}\boldsymbol{B}\boldsymbol{v}\|_p^p,$$

and utilize them as key ingredients in our faster high-accuracy algorithms for optimizing such objectives on graphs. Our construction of sparsifiers for flow objectives builds on the machinery from [25], and our construction of sparsifiers for voltage objectives builds on graph spanners [6, 33, 7].

2 Our Results

Our main results concern flow and voltage problems for mixed $(\ell_2^2 + \ell_p^p)$-objectives for $p \geq 2$. Since our algorithms work best for large p, we restrict our attention to $p = \omega(1)$ in this overview. Section 3 provides detailed running times for all $p \geq 2$. We emphasize that by setting the quadratic term to zero in our mixed $(\ell_2^2 + \ell_p^p)$-objectives, we get new state of the art algorithms for ℓ_p-norm miniziming flows and voltages.

Mixed ℓ_2-ℓ_p-norm minimizing flow. Consider a graph $G = (V, E)$ along with non-negative diagonal matrices $\boldsymbol{R}, \boldsymbol{S} \in \mathbb{R}^{E\times E}$, and a gradient vector $\boldsymbol{g} \in \mathbb{R}^E$, as well as demands $\boldsymbol{d} \in \mathbb{R}^V$. We refer to the diagonal entries of $\boldsymbol{R}$ and $\boldsymbol{S}$ as ℓ_2-weights and ℓ_p-weights respectively. Let $\boldsymbol{B}$ denote the signed edge-vertex incidence of G (see Appendix in full version). We wish to solve the following minimization problem with the objective $\mathcal{E}(\boldsymbol{f}) = \boldsymbol{g}^\top\boldsymbol{f} + \|\boldsymbol{R}^{1/2}\boldsymbol{f}\|_2^2 + \|\boldsymbol{S}\boldsymbol{f}\|_p^p$

$$\min_{\boldsymbol{B}^\top\boldsymbol{f}=\boldsymbol{d}} \mathcal{E}(\boldsymbol{f}) \tag{1}$$

We require $\boldsymbol{g} \perp \{\ker(\boldsymbol{R}) \cap \ker(\boldsymbol{S}) \cap \ker(\boldsymbol{B})\}$ so that the problem has bounded minimum value, and $\boldsymbol{d} \perp \boldsymbol{1}$ so a feasible solution exists. These conditions can be checked in linear time and have a simple combinatorial interpretation. Note that the choice of graph edge directions in $\boldsymbol{B}$ matters for the value of $\boldsymbol{g}^\top\boldsymbol{f}$,. The flow on an edge is allowed to be both positive or negative.

Mixed ℓ_2-ℓ_p-norm minimizing voltages. Consider a graph $G = (V, E)$ along with non-negative diagonal matrices $\boldsymbol{W} \in \mathbb{R}^{E\times E}$ and $\boldsymbol{U} \in \mathbb{R}^{E\times E}$, and demands $\boldsymbol{d} \in \mathbb{R}^V$. We refer to the diagonal entries of $\boldsymbol{W}$ and $\boldsymbol{U}$ as ℓ_2-conductances and ℓ_p-conductances respectively.

In this case, we want to minimize the objective $\mathcal{E}(\boldsymbol{v}) = \boldsymbol{d}^\top \boldsymbol{v} + \|\boldsymbol{W}^{1/2}\boldsymbol{B}\boldsymbol{v}\|_2^2 + \|\boldsymbol{U}\boldsymbol{v}\|_p^p$ in minimization problem

$$\min_{\boldsymbol{v}} \mathcal{E}(\boldsymbol{v}) \tag{2}$$

In the voltage setting, we only require $\boldsymbol{d} \perp \mathbf{1}$ so the problem has bounded minimum value.

Obtaining good solutions. For both these problems, we study high accuracy approximation algorithms that provide feasible solutions $\boldsymbol{x}$ (a flow or a voltage respectively), that approximately minimize the objective function from some starting point $\boldsymbol{x}^{(0)}$, i.e., for some small $\varepsilon > 0$, we have

$$\mathcal{E}(\boldsymbol{x}) - \mathcal{E}(\boldsymbol{x}^\star) \leq \varepsilon(\mathcal{E}(\boldsymbol{x}^{(0)}) - \mathcal{E}(\boldsymbol{x}^\star))$$

wher $\boldsymbol{x}^\star$ denotes an optimal feasible solution. Our algorithms apply to problems with quasipolynomially bounded parameters, including quasipolynomial bounds on non-zero singular values of matrices we work with. Below we state our main algorithmic results.

▶ **Theorem 1** (Flow Algorithmic Result)**.** *Consider a graph G with n vertices and m edges, equipped with non-negative ℓ_2 and ℓ_p-weights, as well as a gradient and demands, all with quasi-polynomially bounded entries. For $p = \omega(1)$, in $p(m^{1+o(1)} + n^{4/3+o(1)}) \log^2 1/\varepsilon$ time we can compute an ε-approximately optimal flow solution to Problem* (1) *with high probability.*

This improves upon [1, 2, 3] which culminated in a $pm^{4/3+o(1)} \log^2 1/\varepsilon$ time algorithm.

▶ **Theorem 2** (Voltage Algorithmic Result)**.** *Consider a graph G with n vertices and m edges, equipped with non-negative ℓ_2 and ℓ_p-conductances, as well as demands, all with quasi-polynomially bounded entries. For $p = \omega(1)$, in $p(m^{1+o(1)} + n^{4/3+o(1)}) \log^2 1/\varepsilon$ time we can compute an ε-approximately optimal voltage solution to Problem* (2) *with high probability.*

Background: Iterative Refinement for Mixed ℓ_2-ℓ_p-norm Flow Objectives. Adil et al. [1] developed a notion of iterative refinement for mixed $(\ell_2^2 + \ell_p^p)$-objectives which in the flow setting, i.e. Problem (1), corresponds to approximating $\mathcal{E}'(\boldsymbol{\delta}) = \mathcal{E}(\boldsymbol{f} + \boldsymbol{\delta})$ using another $(\ell_2^2 + \ell_p^p)$-objective which roughly speaking corresponds to the 2nd degree Taylor series approximation of $\mathcal{E}'(\boldsymbol{\delta})$ combined with an ℓ_p-norm term $\|\boldsymbol{S}\boldsymbol{\delta}\|_p^p$, while ensuring feasibility of $\boldsymbol{f} + \boldsymbol{\delta}$ through a constraint $\boldsymbol{B}\boldsymbol{\delta} = \mathbf{0}$. We call the resulting problem a *residual* problem. Adil et al. [1] showed that obtaining a constant-factor approximate solution to the residual problem in δ is sufficient to ensure that $\mathcal{E}(\boldsymbol{f} + \boldsymbol{\delta})$ is closer to the optimal solution by a multiplicative factor depending only on p. In [2], this result was sharpened to show that such an approximate solution for the residual problem can be used to make $(1 - \Omega(1/p))$ multiplicative progress to the optimum, so that $O(p \log(m/\varepsilon))$ iterations suffice to produce an ε-accurate solution.

In order to solve the residual problem to a constant approximation, Adil et al. [1] developed an accelerated multiplicative weights method for $(\ell_2^2 + \ell_p^p)$-flow objectives, or more generally, for mixed $(\ell_2^2 + \ell_p^p)$-regression in an underconstrained setting.

Sparsification results. Our central technical results in this paper concern sparsification of residual flow and voltage problems, in the sense outlined in the previous paragraph. Concretely, in nearly-linear time, we can take a residual problem on a dense graph and produce a residual problem on a sparse graph with $\widetilde{O}(n)$ edges, with the property that constant factor solutions to the sparse residual problem still make $(1 - \Omega(m^{-\frac{2}{p-1}} p))$ multiplicative

progress on the original problem. This leads to an iterative refinement that converges in $O(pm^{\frac{2}{p-1}}\log(m/\varepsilon))$ steps. However, the accelerated multiplicative weights algorithm that we use for each residual problem now only requires $\widetilde{O}(n^{4/3})$ time to compute a crude solution.

Flow residual problem sparsification. In the flow setting, we show the following:

▶ **Theorem 3** (Informal Flow Sparsification Result)**.** *Consider a graph G with n vertices and m edges, equipped with non-negative ℓ_2 and ℓ_p-weights, as well as a gradient. In $\widetilde{O}(m)$ time, we can compute a graph H with n vertices and $\widetilde{O}(n)$ edges, equipped with non-negative ℓ_2 and ℓ_p-weights, as well as a gradient, such that a constant factor approximation to the flow residual problem on H, when scaled by $m^{\frac{-1}{p-1}}$ results in an $\widetilde{O}(m^{\frac{2}{p-1}})$ approximate solution to the flow residual problem on G. The algorithm works for all $p \geq 2$ and succeeds with high probability.*

Our sparsification techniques build on [25], require a new bucketing scheme to deal with non-uniform ℓ_p-weights, as well as a prepreprocessing step to handle cycles with zero ℓ_2-weight and ℓ_p-weight. This preprocessing scheme in turn necessitates a more careful analysis of additive errors introduced by gradient rounding, and we provide a more powerful framework for this than [25].

Voltage residual problem sparsification. In the voltage setting, we show the following.

▶ **Theorem 4** (Voltage Sparsification Result (Informal))**.** *Consider a graph G with n vertices and m edges, equipped with non-negative ℓ_2 and ℓ_p-conductances. In $\widetilde{O}(m)$ time, we can compute a graph H with n vertices and $\widetilde{O}(n)$ edges, equipped with non-negative ℓ_2 and ℓ_p-conductances, such that constant factor approximation to the voltage residual problem on H, when scaled by $m^{\frac{-1}{p-1}}$ results in an $\widetilde{O}(m^{\frac{1}{p-1}})$ approximate solution to the voltage residual problem on G. The algorithm works for all $p \geq 2$ and succeeds with high probability.*

Note that our voltage sparsification is slightly stronger than our flow sparsification, as the former loses only a factor $\widetilde{O}(m^{\frac{1}{p-1}})$ in the approximation while the latter loses a factor $\widetilde{O}(m^{\frac{2}{p-1}})$. Our voltage sparsification uses a few key observations: In voltage space, surprisingly, we can treat treat the ℓ_2 and ℓ_p costs separately. This behavior is very different than the flow case, and arises becase in voltage space, every edge provides an "obstacle", i.e. adding an edge increases cost, whereas in flow space, every edge provides an "opportunity", i.e. adding an edge decreases cost. This means that in voltage space, we can separately account for the energy costs created by our ℓ_2 and ℓ_p terms, whereas in flow space, the ℓ_2 and ℓ_p weights must be highly correlated in a sparsifier. Armed with this decoupling observation, we preserve ℓ_2 cost using standard tools for spectral graph sparsification, and we preserve ℓ_p cost approximately by a reduction to graph distance preservation, which we in turn achieve using weighted undirected graph spanners.

Voltage space accelerated multiplicative weights solver. The algorithm from [1] for constant approximate solutions to the residual problem works in the flow setting. Using iterative refinement, the algorithm could be used to compute high-accuracy solutions. Because we can use high-accuracy flow solutions to extract high-accuracy solutions to the dual voltage problem, [1] were also able to produce solutions to ℓ_q-norm minimizing voltage problems (where ℓ_q for $q = p/(p-1)$ is the dual norm to ℓ_p). Hence, by solving ℓ_p-flow problems for all $p \in (2, \infty)$, [1] were able to solve ℓ_q-norm minimizing voltage problems for all $q \in (1, 2)$.

Our sparsification of flow and voltage problems works only for $p \geq 2$. Thus, in order to solve for q-norm minimizing voltages for $q > 2$, we require a solver that works directly in voltage space for mixed $(\ell_2^2 + \ell_p^p)$-voltage objectives.

We develop an accelerated multiplicative weights algorithm along the lines of [11, 10, 1] that works directly in voltage space for mixed $(\ell_2^2 + \ell_p^p)$-objectives, or more generally for overconstrained mixed $(\ell_2^2 + \ell_p^p)$-objective regression. Concretely, this directly gives an algorithm for computing crude solutions to the residual problems that arise from applying [1] iterative refinement to Problem (2). Our solver produces an improved $O(1)$-approximation to the residual problem rather than a $p^{O(p)}$-approximation from [1]. This gives an $\widetilde{O}(m^{4/3})$ high-accuracy algorithm for mixed $(\ell_2^2 + \ell_p^p)$-objective voltage problems for $p > 2$, unlike [1], which could only solve pure $p > 2$ voltage problems. We then speed this up to a $p(m^{1+o(1)} + n^{4/3+o(1)})$ time algorithm for $p = \omega(1)$ by developing a sparsification procedure that applies directly to mixed $(\ell_2^2 + \ell_p^p)$-voltage problems for $p > 2$.

Mixed ℓ_2-ℓ_p-norm regression. Our framework can also be applied outside of a graph setting, where our new accelerated multiplicative weights algorithm for overconstrained mixed $(\ell_2^2 + \ell_p^p)$-regression gives new state-of-the-art results in some regimes when combined with new sparsification results. In this setting we develop sparsification techniques based on the Lewis weights sampling from the work of Cohen and Peng [17]. We focus on the case $2 < p < 4$, where [17] provided fast algorithms for Lewis weight sampling.

▶ **Theorem 5** (General Matrices Sparsification Result). *Let $p \in [2, 4)$, let $\boldsymbol{M} \in \mathbb{R}^{m_1 \times n}, \boldsymbol{N} \in \mathbb{R}^{m_2 \times n}$ be matrices, $m_1, m_2 \geq n$, and let $LSS(\boldsymbol{B})$ denote the time to solve a linear system in $\boldsymbol{B}^\top \boldsymbol{B}$. Then, we may compute $\widetilde{\boldsymbol{M}}, \widetilde{\boldsymbol{N}} \in \mathbb{R}^{O(n^{p/2}\log(n)) \times n}$ such that with probability at least $1 - \frac{1}{n^{\Omega(1)}}$, for all $\Delta \in \mathbb{R}^n$,*

$$\|\widetilde{\boldsymbol{M}}\Delta\|_2^2 + \|\widetilde{\boldsymbol{N}}\Delta\|_p^p \approx_{O(1)} \|\boldsymbol{M}\Delta\|_2^2 + \|\boldsymbol{N}\Delta\|_p^p,$$

in time $\tilde{O}\Big(\mathrm{nnz}(\boldsymbol{M}) + \mathrm{nnz}(\boldsymbol{N}) + LSS(\widehat{\boldsymbol{M}}) + LSS(\widehat{\boldsymbol{N}})\Big)$, for some $\widehat{\boldsymbol{M}}$ and $\widehat{\boldsymbol{N}}$ each containing $O(n\log(n))$ rescaled rows of $\boldsymbol{M}$ and $\boldsymbol{N}$, respectively.

▶ **Theorem 6** (General Matrices Algorithmic Result). *For $p \in [2, 4)$, with high probability we can find an ε-approximate solution to (3) in time*

$$\widetilde{O}\Big(\Big(\mathrm{nnz}(\boldsymbol{M}) + \mathrm{nnz}(\boldsymbol{N}) + \Big(LSS(\widetilde{\boldsymbol{M}}) + LSS(\widetilde{\boldsymbol{N}})\Big)n^{\frac{p(p-2)}{6p-4}}\Big)\log^2(1/\varepsilon)\Big),$$

for some $\widetilde{\boldsymbol{M}}$ and $\widetilde{\boldsymbol{N}}$ each containing $O(n^{p/2}\log(n))$ rescaled rows of $\boldsymbol{M}$ and $\boldsymbol{N}$, respectively, where $LSS(\boldsymbol{A})$ is the time required to solve a linear equation in $\boldsymbol{A}^\top \boldsymbol{A}$ to quasipolynomial accuracy.

Note that for all $p \in (2, 4)$, we have that the exponent $\frac{p(p-2)}{6p-4} \leq 0.4$.

▶ Remark 7. By [14], a linear equation in $\boldsymbol{A}^\top \boldsymbol{A}$, where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ can be solved to quasipolynomial accuracy in time $\widetilde{O}(\mathrm{nnz}(\boldsymbol{A}) + n^\omega)$.

Using the above result for solving the required linear systems, we get a running time of $\widetilde{O}(\mathrm{nnz}(\boldsymbol{M}) + \mathrm{nnz}(\boldsymbol{N}) + (n^{p/2} + n^\omega)n^{\frac{p(p-2)}{6p-4}})$, matching an earlier input sparsity result by Bubeck et al. [9] that achieves $\widetilde{O}((\mathrm{nnz}(\boldsymbol{M}) + \mathrm{nnz}(\boldsymbol{N}))(1 + n^{\frac{1}{2}}m^{-\frac{1}{p}}) + m^{\frac{1}{2}-\frac{1}{p}}n^2 + n^\omega)$, where $\boldsymbol{M} \in \mathbb{R}^{m_1 \times n}, \boldsymbol{N} \in \mathbb{R}^{m_2 \times n}$ and $m = \max\{m_1, m_2\}$.

3 Main Algorithm

In this section, we prove Theorems 1, 2, and 6. We first design an algorithm to solve the following general problem:

▶ **Definition 8.** *For matrices* $\boldsymbol{M} \in \mathbb{R}^{m_1 \times n}$, $\boldsymbol{N} \in \mathbb{R}^{m_2 \times n}$ *and* $\boldsymbol{A} \in \mathbb{R}^{d \times n}$, $m_1, m_2 \geq n, d \leq n$, *and vectors* $\boldsymbol{b} \perp \{\ker(\boldsymbol{M}) \cap \ker(\boldsymbol{N}) \cap \ker(\boldsymbol{A})\}$ *and* $\boldsymbol{c} \in im(\boldsymbol{A})$, *we want to solve*

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{b}^\top \boldsymbol{x} + \|\boldsymbol{M}\boldsymbol{x}\|_2^2 + \|\boldsymbol{N}\boldsymbol{x}\|_p^p \\ s.t. \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{c}. \end{aligned} \tag{3}$$

In order to solve the above problem, we use the iterative refinement framework from [2] to obtain a residual problem which is defined as follows.

▶ **Definition 9.** *For any* $p \geq 2$, *we define the residual problem* $res(\Delta)$, *for* (3) *at a feasible* $\boldsymbol{x}$ *as,*

$$\max_{\boldsymbol{A}\Delta=0} \quad res(\Delta) \stackrel{\text{def}}{=} \boldsymbol{g}^\top \Delta - \Delta^\top \boldsymbol{R} \Delta - \|\boldsymbol{N}\Delta\|_p^p, \text{ where,}$$

$$\boldsymbol{g} = \frac{1}{p}\boldsymbol{b} + \frac{2}{p}\boldsymbol{M}^\top \boldsymbol{M}\boldsymbol{x} + |\boldsymbol{N}\boldsymbol{x}|^{p-2}\boldsymbol{N}\boldsymbol{x} \quad \text{and} \quad \boldsymbol{R} = \frac{2}{p^2}\boldsymbol{M}^\top \boldsymbol{M} + 2\boldsymbol{N}^\top Diag(|\boldsymbol{N}\boldsymbol{x}|^{p-2})\boldsymbol{N}.$$

This residual problem can further be reduced by moving the term linear in $\boldsymbol{x}$ to the constraints via a binary search. This leaves us with a problem of the form,

$$\begin{aligned} \min_{\Delta} \quad & \Delta^\top \boldsymbol{R}\Delta + \|\boldsymbol{N}\Delta\|_p^p \\ \text{s.t.} \quad & \boldsymbol{g}^\top \Delta = a, \boldsymbol{A}\Delta = 0, \end{aligned}$$

for some constant a.

In order to solve the above problem with $\ell_2^2 + \ell_p^p$ objective, we reduce the instance size via a sparsification routine, and then solve the smaller problem by a multiplicative weights algorithm. We adapt the multiplicative-weights algorithm from [1] to work in the voltage space while improving the p dependence of the runtime from $p^{O(p)}$ to p, and the approximation quality from $p^{O(p)}$ to $O(1)$. The precise sparsification routines are described in later sections.

For large p, i.e., $p > \log m$, in order to get a linear dependence on the running time on p, we need to reduce the residual problem in ℓ_p-norm to a residual problem in $\log m$-norm by using the framework from [3].

The entire meta-algorithm is described formally in Algorithm 1, and its guarantees are described by the next theorem. Most proof details are deferred to the full version.

▶ **Theorem 10.** *For an instance of Problem* (3), *suppose we are given a starting solution* $\boldsymbol{x}^{(0)}$ *that satisfies* $\boldsymbol{A}\boldsymbol{x}^{(0)} = \boldsymbol{c}$ *and is a* κ *approximate solution to the optimum. Consider an iteration of the while loop, line 8 of Algorithm 1 for the* ℓ_p*-norm residual problem at* $\boldsymbol{x}^{(t)}$. *We can define* μ_1 *and* κ_1 *such that if* $\bar{\Delta}$ *is a* β *approximate solution to a corresponding* p'*-norm residual problem, then* $\mu_1\bar{\Delta}$ *is a* κ_1*-approximate solution to the* p*-residual problem. Further, suppose we have the following procedures,*

1. SPARSIFY*: Runs in time* K, *takes as input any matrices* $\boldsymbol{R}, \boldsymbol{N}$ *and vector* $\boldsymbol{g}$ *and returns* $\widetilde{\boldsymbol{R}}, \widetilde{\boldsymbol{N}}, \widetilde{\boldsymbol{g}}$ *having sizes at most* $\tilde{n} \times n$ *for the matrices , such that if* $\widetilde{\Delta}$ *is a* β *approximate solution to,*

$$\max_{\boldsymbol{A}\Delta=0} \quad \widetilde{\boldsymbol{g}}^\top \Delta - \|\widetilde{\boldsymbol{R}}\Delta\|_2^2 - \|\widetilde{\boldsymbol{N}}\Delta\|_{p'}^{p'},$$

for any $p' \geq 2$, *then* $\mu_2\widetilde{\Delta}$, *for a computable* μ_2 *is a* $\kappa_2\beta$*-approximate solution for,*

$$\max_{\boldsymbol{A}\Delta=0} \quad res(\Delta) \stackrel{\text{def}}{=} \boldsymbol{g}^\top \Delta - \|\boldsymbol{R}^{1/2}\Delta\|_2^2 - \|\boldsymbol{N}\Delta\|_{p'}^{p'}.$$

2. SOLVER: *Approximately solves* (4) *to return* $\bar{\Delta}$ *such that* $\|\widetilde{\boldsymbol{R}}\bar{\Delta}\|_2^2 \le \kappa_3\nu$ *and* $\|\widetilde{\boldsymbol{N}}\Delta\|_p^p \le \kappa_4\nu$ *in time* $\tilde{K}(\tilde{n})$ *for instances of size at most* $\tilde{n}$.

Algorithm 1 finds an ε*-approximate solution for Problem* (3) *in time*

$$\widetilde{O}\left(p\kappa_4^{1/(p-1)}\kappa_3\kappa_2\kappa_1(K+\tilde{K}(\tilde{n}))\log\left(\frac{\kappa p}{\varepsilon}\right)^2\right).$$

Algorithm 1 Meta-Algorithm for ℓ_p Flows and Voltages.

procedure SPARSIFIED-P-PROBLEMS($\boldsymbol{A}, \boldsymbol{M}, \boldsymbol{N}, \boldsymbol{c}, \boldsymbol{b}, p$)
  $\boldsymbol{x} \leftarrow \boldsymbol{x}^{(0)}$, such that $\boldsymbol{f}(\boldsymbol{x}^{(0)}) \le \kappa\text{OPT}$
  $T \leftarrow \widetilde{O}\left(p\kappa_1\kappa_2\kappa_3\log\left(\frac{\kappa}{\varepsilon}\right)\right)$
  for $t = 0$ to T **do**
    At $\boldsymbol{x}^{(t)}$ define $\boldsymbol{g}, \boldsymbol{R}, \boldsymbol{N}$ and $res(\Delta)$, the residual problem (Definition 9)
    $a \leftarrow \frac{1}{2}, b \leftarrow 1, \mu_1 \leftarrow 1, \kappa_1 \leftarrow 1$
    $\nu \leftarrow \boldsymbol{f}(\boldsymbol{x}^{(0)})$
    while $\nu \ge \varepsilon\frac{\boldsymbol{f}(\boldsymbol{x}^{(0)})}{\kappa p}$ **do**
      if $p > \log m$ **then** ▷ Convert ℓ_p-norm residual to $\log m$-norm residual
        $p' \leftarrow \log m$
        $\boldsymbol{N}' \leftarrow \frac{1}{2^{1/p'}}\left(\frac{\nu}{m}\right)^{\frac{1}{p'}-\frac{1}{p}}\boldsymbol{N}$
        $a \leftarrow \frac{1}{33}, b \leftarrow O(1)m^{o(1)}$
        $\mu_1 \leftarrow m^{-o(1)}, \kappa_1 \leftarrow m^{o(1)}$ ▷ Lose κ_1 in approx. when scaled by μ_1
        $(\widetilde{\boldsymbol{g}}, \widetilde{\boldsymbol{R}}, \widetilde{\boldsymbol{N}}) \leftarrow$ SPARSIFY$(\boldsymbol{g}, \boldsymbol{R}, \boldsymbol{N}')$ ▷ Lose κ_2 in approx. when scaled by μ_2
      else
        $(\widetilde{\boldsymbol{g}}, \widetilde{\boldsymbol{R}}, \widetilde{\boldsymbol{N}}) \leftarrow$ SPARSIFY$(\boldsymbol{g}, \boldsymbol{R}, \boldsymbol{N})$ ▷ Lose κ_2 in approx. when scaled by μ_2
        $p' \leftarrow p$
      Use SOLVER to compute κ_3, κ_4 approximate solution to

$$\widetilde{\Delta}^{(\nu)} \leftarrow \arg\min_{\Delta} \quad \|\widetilde{\boldsymbol{R}}^{1/2}\Delta\|_2^2 + \|\widetilde{\boldsymbol{N}}\Delta\|_{p'}^{p'} \quad \text{s.t.} \quad \widetilde{\boldsymbol{g}}^\top\Delta = a\nu, \quad \boldsymbol{A}\Delta = 0. \tag{4}$$

      $\bar{\Delta}^{(\nu)} \leftarrow \frac{a}{2b\kappa_3\kappa_4^{1/(p'-1)}}\mu_2\mu_1\widetilde{\Delta}^{(\nu)}$
      $\nu \leftarrow \nu/2$
    $\Delta \leftarrow \arg\min_{\bar{\Delta}^{(\nu)}} \boldsymbol{f}\left(\boldsymbol{x} - \frac{\bar{\Delta}^{(\nu)}}{p}\right)$
    $\boldsymbol{x} \leftarrow \boldsymbol{x} - \frac{\Delta}{p}$
  return $\boldsymbol{x}$

3.1 Algorithms for ℓ_p-norm Problems

The problems discussed in Section 2 are special cases of Problem (3), which means we can use Algorithm 1. To prove our results, we will utilize Theorem 10, with the respective sparsification procedures and the following multiplicative-weights based algorithm for solving problems of the form,

$$\min_{\Delta} \quad \Delta^\top\boldsymbol{M}^\top\boldsymbol{M}\Delta + \|\boldsymbol{N}\Delta\|_p^p \quad \text{s.t.} \quad \boldsymbol{A}\Delta = \boldsymbol{c}. \tag{5}$$

We describe our solver formally and prove the following theorem about its guarantees in the full version.

▶ **Theorem 11.** *Let $p \geq 2$. Consider an instance of Problem (5) described by matrices $\boldsymbol{A} \in \mathbb{R}^{d\times n}, \boldsymbol{N} \in \mathbb{R}^{m_1\times n}, \boldsymbol{M} \in \mathbb{R}^{m_2\times n}$, $d \leq n \leq m_1, m_2$, and vector $\boldsymbol{c} \in \mathbb{R}^d$. If the optimum of this problem is at most ν, Procedure* Residual-Solver *returns an $\boldsymbol{x}$ such that $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{c}$, and $\boldsymbol{x}^\top \boldsymbol{M}^\top \boldsymbol{M}\boldsymbol{x} \leq O(1)\nu$ and $\|\boldsymbol{N}\boldsymbol{x}\|_p^p \leq O(3^p)\nu$. The algorithm makes $O\left(pm_1^{\frac{p-2}{(3p-2)}}\right)$ calls to a linear system solver.*

We utilize Procedure Residual-Solver as the Procedure Solver in Algorithm Sparsified-p-Problems. The algorithm uses the procedure only for solving problems instances with $p \leq \log m$. Thus, its running time is $\tilde{K}(\tilde{n}) = \widetilde{O}\left(\tilde{n}^{\frac{p-2}{3p-2}} \cdot \mathrm{LSS}(\tilde{n})\right) \leq \widetilde{O}\left(\tilde{n}^{1/3} \cdot \mathrm{LSS}(\tilde{n})\right)$, where $\mathrm{LSS}(\tilde{n})$ denotes the time required to solve a linear system in matrices of size $\tilde{n}$. We also have, $\kappa_3 = O(1), \kappa_4^{1/(p-1)} = O(1)$.

We next estimate the values of κ_1 and μ_1. If $p \leq \log m$, we have $\mu_1 = 1$ and $\kappa_1 = 1$. Otherwise, $\mu_1 = \widetilde{O}(1)$ and $\kappa_1 = O(m^{o(1)})$ (Refer to the full version).

In order to obtain an initial solution, we usually solve an ℓ_2-norm problem. This gives an $m^{p/2}$ approximate initial solution which results in a factor of p^2 in the running time. To avoid this, we can do a homotopy on p similar to [3], i.e., start with an ℓ_2 solution and solve the ℓ_{2^2} problem to a constant approximation, followed by $\ell_{2^3}, ..\ell_p$. We note that a constant approximate solution to the $\ell_{p/2}$-norm problem gives an $O(m)$ approximation to the ℓ_p problem and thus, we can solve $\log p$ problems where we can assume $\kappa = O(m)$.

We now complete the proof of our various algorithmic results by utilizing sparsification procedures specific to each problem.

ℓ_p Flows

We will prove Theorem 1 (Flow Algorithmic Result), with explicit p dependencies.

Proof. From Theorem 3, we obtain a sparse graph in $K = \widetilde{O}(m)$ time with $\tilde{n} = \widetilde{O}(n)$ edges. A constant factor approximation to the flow residual problem on this sparse graph when scaled by $\mu_2 = m^{-\frac{1}{p-1}}$ gives a $\kappa_2 = \widetilde{O}\left(m^{\frac{2}{p-1}}\right)$-approximate solution to the flow residual problem on the original graph. We can solve linear systems on the sparse graph in $\widetilde{O}(\tilde{n}) = \widetilde{O}(n)$ time using fast Laplacian solvers. Using all these values in Theorem 10, we get the final runtime to be $pm^{\frac{2}{p-1}+o(1)}\left(m + n^{1+\frac{p-2}{3p-2}}\right)\log^2\left(\frac{pm}{\varepsilon}\right)$ as claimed. We prove Theorem 3 in the full version. ◀

ℓ_p Voltages

We will prove Theorem 2 (Voltage Algorithmic Result), with explicit p dependencies.

Proof. From Theorem 4, we obtain a sparse graph in $K = \widetilde{O}(m)$ time with $\tilde{n} = \widetilde{O}(n)$ edges. A constant factor approximation to the voltage residual problem on this sparse graph when scaled by $\mu_2 = m^{-\frac{1}{p-1}}$ gives a $\kappa_2 = \widetilde{O}\left(m^{\frac{1}{p-1}}\right)$-approximate solution to the voltage residual problem on the original graph. We can solve linear systems on the sparse graph in $\widetilde{O}(\tilde{n}) = \widetilde{O}(n)$ time using fast Laplacian solvers. Using these values in Theorem 10, we get the final runtime to be $pm^{\frac{1}{p-1}+o(1)}\left(m + n^{1+\frac{p-2}{3p-2}}\right)\log^2\left(\frac{pm}{\varepsilon}\right)$ as claimed. We prove Theorem 4 in Section 4. ◀

General Matrices

We will now prove Theorem 6.

Proof. We assume Theorem 5, which we prove in Appendix (refer to full version). From the theorem, we have $\kappa_2 = O(1)$ and $\mu_2 = O(1)$. Note that $K = \mathrm{LSS}(\widehat{\boldsymbol{M}}) + \mathrm{LSS}(\widehat{\boldsymbol{N}})$ for some $\widehat{\boldsymbol{M}}, \widehat{\boldsymbol{N}} \in \mathbb{R}^{O(n\log(n))\times n}$, which is the time required to solve linear systems in $\widehat{\boldsymbol{M}}^\top \widehat{\boldsymbol{M}}$ and $\widehat{\boldsymbol{N}}^\top \widehat{\boldsymbol{N}}$, respectively. Since, by Theorem 5, the size of $\widetilde{\boldsymbol{M}}$ and $\widetilde{\boldsymbol{N}}$ is $\tilde{n} = O(n^{p/2}\log(n))$, the cost from the solver in Theorem 11 is $\tilde{O}_p\Big(\Big(\mathrm{LSS}(\widetilde{\boldsymbol{M}}) + \mathrm{LSS}(\widetilde{\boldsymbol{N}})\Big) n^{\frac{p(p-2)}{6p-4}}\Big)$. ◀

4 Construction of Sparsifiers for $\ell_2^2 + \ell_p^p$ Voltages

In this section, we prove a formal version of the voltage sparsification result (Theorem 4):

▶ **Theorem 12.** *Consider a graph $G = (V, E)$ with non-negative 2-weights $\boldsymbol{w} \in \mathbb{R}^E$ and non-negative p-weights $\boldsymbol{s} \in \mathbb{R}^E$, with m and n vertices. We can produce a graph $H = (V, F)$ with edges $F \subseteq E$, ℓ_2-weights $\boldsymbol{u} \in \mathbb{R}^F$, and ℓ_p-weights $\boldsymbol{t} \in \mathbb{R}^F$, such that with probability at least $1 - \delta$ the graph H has $O(n\log(n/\varepsilon))$ edges and*

$$\frac{1}{1.5}\|\boldsymbol{W}\boldsymbol{B}_G\boldsymbol{x}\|_2 \le \|\boldsymbol{U}\boldsymbol{B}_H\boldsymbol{x}\|_2 \le 1.5\|\boldsymbol{W}\boldsymbol{B}_G\boldsymbol{x}\|_2 \tag{6}$$

and for any $p \in [1, \infty]$

$$\frac{1}{m^{1/p}\log(n)}\|\boldsymbol{S}\boldsymbol{B}_G\boldsymbol{x}\|_p \le \|\boldsymbol{T}\boldsymbol{B}_H\boldsymbol{x}\|_p \le \|\boldsymbol{S}\boldsymbol{B}_G\boldsymbol{x}\|_p \tag{7}$$

where $\boldsymbol{W} = \textsc{Diag}(\boldsymbol{w})$, $\boldsymbol{U} = \textsc{Diag}(\boldsymbol{u})$, $\boldsymbol{S} = \textsc{Diag}(\boldsymbol{s})$, $\boldsymbol{T} = \textsc{Diag}(\boldsymbol{t})$. We denote the routine computing H and $\boldsymbol{u}, \boldsymbol{t}$ by SpannerSparsify, so that $(H, \boldsymbol{u}, \boldsymbol{t}) = \textsc{SpannerSparsify}(G, \boldsymbol{w}, \boldsymbol{s})$. This algorithm runs in $\widetilde{O}(m\log(1/\delta))$ time.

We will first define some terms required for our result. Given a undirected graph $G = (V, E)$, with edge lengths $\boldsymbol{l} \in \mathbb{R}^E$, and $u, v \in V$, we let $d_G(u, v)$ denote the shortest path distance in G w.r.t $\boldsymbol{l}$, so that if P is the shortest path w.r.t $\boldsymbol{l}$ then

$$d_{G,\boldsymbol{l}}(u, v) = \sum_{e\in P} \boldsymbol{l}(e)$$

▶ **Definition 13.** *Given a undirected graph $G = (V, E)$ with edge lengths $\boldsymbol{l} \in \mathbb{R}^E$, a K-spanner is a subgraph H of G with the same edge lengths s.t. $d_H(u, v) \le K d_G(u, v)$.*

Baswana and Sen showed the following result on spanners [7].

▶ **Theorem 14.** *Given an undirected graph $G = (V, E, \boldsymbol{l})$ with m edges and n vertices, and an integer $k > 1$, we can compute a $(2k-1)$-spanner H of G with $O(n^{1+1/k})$ edges in expected time $O(km)$.*

▶ **Lemma 15.** *Given an undirected graph $G = (V, E)$ with positive edge lengths $\boldsymbol{l} \in \mathbb{R}^E$, and a K-spanner $H = (V, F)$ of G, for all $\boldsymbol{x} \in \mathbb{R}^V$ we have*

$$\max_{(u,v)\in F} \frac{1}{\boldsymbol{l}(u,v)}|\boldsymbol{x}(u) - \boldsymbol{x}(v)| \le \max_{(u,v)\in E} \frac{1}{\boldsymbol{l}(u,v)}|\boldsymbol{x}(u) - \boldsymbol{x}(v)| \le K \max_{(u,v)\in F} \frac{1}{\boldsymbol{l}(u,v)}|\boldsymbol{x}(u) - \boldsymbol{x}(v)|$$

Proof. The inequality $\max_{(u,v)\in F} \frac{1}{\boldsymbol{l}(u,v)}|\boldsymbol{x}(u)-\boldsymbol{x}(v)| \leq \max_{(u,v)\in E} \frac{1}{\boldsymbol{l}(u,v)}|\boldsymbol{x}(u)-\boldsymbol{x}(v)|$ is immediate from $F \subseteq E$.

To prove the second inequality, we note that if $(u,v) \in E$ has shortest path P in H then

$$\frac{1}{\boldsymbol{l}(u,v)}|\boldsymbol{x}(u)-\boldsymbol{x}(v)| \leq \frac{K}{\sum_{(z,y)\in P}\boldsymbol{l}(z,y)}\left|\sum_{(z,y)\in P}\boldsymbol{x}(z)-\boldsymbol{x}(y)\right| \leq \max_{(z,y)\in P}\frac{K}{\boldsymbol{l}(z,y)}|\boldsymbol{x}(z)-\boldsymbol{x}(y)|.$$

◀

▶ **Definition 16.** *Given a undirected graph $G=(V,E)$ with m edges and n vertices with positive edge ℓ_2-weights $\boldsymbol{w} \in \mathbb{R}^E$, a spectral ε-approximation of G is a graph $H=(V,F)$ with $F \subseteq E$ with positive edge ℓ_2-weights $\boldsymbol{u} \in \mathbb{R}^F$ s.t.*

$$\frac{1}{1+\varepsilon}\|\boldsymbol{W}\boldsymbol{B}_G\boldsymbol{x}\|_2 \leq \|\boldsymbol{U}\boldsymbol{B}_H\boldsymbol{x}\|_2 \leq (1+\varepsilon)\|\boldsymbol{W}\boldsymbol{B}_G\boldsymbol{x}\|_2$$

where $\boldsymbol{W} = \textsc{Diag}(\boldsymbol{w})$ and $\boldsymbol{U} = \textsc{Diag}(\boldsymbol{u})$.

The following result on spectral sparsifiers was shown by Spielman and Srivastava [40] (see also [43]).

▶ **Theorem 17.** *Given a graph $G=(V,E)$ with positive ℓ_2-weights $\boldsymbol{w} \in \mathbb{R}^E$ with m edges and n vertices, for any $\varepsilon \in (0,1/2]$, we can produce a graph $H=(V,F)$ with edges $F \subseteq E$ and ℓ_2-weights $\boldsymbol{u} \in \mathbb{R}^F$ such that H has $O(n\varepsilon^{-2}\log(n/\delta))$ edges and with probability at least $1-\delta$ we have that $(H,\boldsymbol{u})$ is a spectral ε-approximation of $(G,\boldsymbol{w})$. We denote the routine computing H and $\boldsymbol{u}$ by SpectralSparsify, so that $(H,\boldsymbol{u}) = \textsc{SpectralSparsify}(G,\boldsymbol{s},\varepsilon,\delta)$. This algorithm runs in $\widetilde{O}(m)$ time. Furthermore, if the weights $\boldsymbol{w}$ are quasipolynomially bounded, then so are the weights of $\boldsymbol{u}$.*

We can now prove our main result.

Proof of Theorem 12. We consider a graph $G=(V,E)$ with m edges and n vertices, and with non-negative ℓ_p-weights $\boldsymbol{r} \in \mathbb{R}^E$, non-negative ℓ_2-weights $\boldsymbol{s} \in \mathbb{R}^E$. We define $\hat{E} \subseteq E$ to be the edges s.t. $\boldsymbol{s}(e) > 0$, and then let $\boldsymbol{l} \in \mathbb{R}^{\hat{E}}$ by $\boldsymbol{l}(e) = 1/\boldsymbol{s}(e)$, and $\hat{G} = (V,\hat{E})$. We then apply Theorem 14 to $\hat{G}$ with $\boldsymbol{l}$ as edge lengths, and with $k = \log(n)$. We turn the algorithm of Theorem 14 into running time $\widetilde{O}(m\log(1/\delta))$, instead of expected time $\widetilde{O}(m)$, by applying the standard Las Vegas to Monte-Carlo reduction. With probability $1-\delta/2$, this gives us a $\log n$-spanner H_1 of $\hat{G}$, and we define $\boldsymbol{t}$ by restricting $\boldsymbol{s}$ to the edges of H_1. By Lemma 15, we then have

$$\|\boldsymbol{T}\boldsymbol{B}_{H_1}\boldsymbol{x}\|_\infty \leq \|\boldsymbol{S}\boldsymbol{B}_G\boldsymbol{x}\|_\infty \leq \log(n)\|\boldsymbol{T}\boldsymbol{B}_{H_1}\boldsymbol{x}\|_\infty$$

Because $\boldsymbol{T}\boldsymbol{B}_{H_1}\boldsymbol{x}$ is a restriction of $\boldsymbol{S}\boldsymbol{B}_G\boldsymbol{x}$ to a subset of the coordinates, we always have for any $p \geq 1$ that $\|\boldsymbol{T}\boldsymbol{B}_{H_1}\boldsymbol{x}\|_p \leq \|\boldsymbol{S}\boldsymbol{B}_G\boldsymbol{x}\|_p$.

At the same time, we also have

$$\|\boldsymbol{S}\boldsymbol{B}_G\boldsymbol{x}\|_p \leq m^{1/p}\|\boldsymbol{S}\boldsymbol{B}_G\boldsymbol{x}\|_\infty \leq m^{1/p}\log(n)\|\boldsymbol{T}\boldsymbol{B}_{H_1}\boldsymbol{x}\|_\infty \leq m^{1/p}\log(n)\|\boldsymbol{T}\boldsymbol{B}_{H_1}\boldsymbol{x}\|_p$$

We define $\tilde{E} \subseteq E$ to be the edges s.t. $\boldsymbol{r}(e) > 0$, and the let $\tilde{G} = (V,\tilde{E})$. Now, appealing to Theorem 17, we let $(H_2,\boldsymbol{u}) = \textsc{SpectralSparsify}(\tilde{G},\boldsymbol{r},1/2,\varepsilon/2)$.

Finally, we form H by taking the union of the edge sets of H_1 and H_2 and extending $\boldsymbol{u}$ and $\boldsymbol{t}$ to the new edge set by adding zero entries as needed. By a union bound, the approximation guarantees of Equations (6) and (7) simultaneously hold with probability at least $1-\delta$.

The edge set remains bounded in size by $O(n\log n)$. ◀

To see Theorem 4, note that from Theorem 12, we get,

$$m^{-\frac{1}{p-1}}\left(m^{-\frac{1}{p-1}}\|\boldsymbol{W}\boldsymbol{B}_G\boldsymbol{x}\|_2^2+m^{-1}\|\boldsymbol{S}\boldsymbol{B}_G\boldsymbol{x}\|_p^p\right)\leq m^{-\frac{1}{p-1}}\left(\|\boldsymbol{U}\boldsymbol{B}_H\boldsymbol{x}\|_2^2+\|\boldsymbol{T}\boldsymbol{B}_H\boldsymbol{x}\|_p^p\right)$$

The other direction is easy to see.

5 Extensions of Our Results and Open Problems

Solving dual problems: q-norm minimizing flows and voltages for $q < 2$

When the mixed $(\ell_2^2+\ell_p^p)$-objective flow problem (Problem (1)) is restricted to the case $\boldsymbol{g}=\boldsymbol{0}$ and $\boldsymbol{R}=\boldsymbol{0}$, it becomes a pure ℓ_p-norm minimizing flow problem, and its dual problem can be slightly rearranged to give

$$\min_{\boldsymbol{v}}\boldsymbol{d}^\top\boldsymbol{v}+\left\|\boldsymbol{S}^{-1}\boldsymbol{B}\boldsymbol{v}\right\|_q^q \tag{8}$$

where $q=p/(p-1)=1+1/(p-1)$. We refer to the diagonal entries of $\boldsymbol{S}^{-1}$ as ℓ_q-conductances. Because we can solve Problem (1) to high-accuracy in near-linear time for $p=\omega(1)$, this allows us to solve Problem (8), the dual voltage ℓ_q-norm minimization, in time $p(m^{1+o(1)}+n^{4/3+o(1)})\log^2 1/\varepsilon$ (see [1, Section 7] for the reduction). We summarize this in the theorem below.

▶ **Theorem 18** (Voltage Algorithmic Result, $q<2$ (Informal)). *Consider a graph G with n vertices and m edges, equipped with positive ℓ_q-conductances, as well as a demand vector. For $1<q<2$, when $q=1+o(1)$, in $\mathrm{poly}\left(\frac{1}{q-1}\right)(m^{1+o(1)}+n^{4/3+o(1)})\log^2 1/\varepsilon$ time, we can compute an ε-approximately optimal voltage solution to Problem (8) with high probability.*

Similarly, we can solve ℓ_q-norm minimizing flows for $q<2$ as dual to the ℓ_p-voltage problem, a special case of the mixed $(\ell_2^2+\ell_p^p)$-voltage problem. Picking $\boldsymbol{W}=\boldsymbol{0}$ in Problem (2), we obtain a pure ℓ_p-norm minimizing voltage problem, and its dual problem can be slightly rearranged to give

$$\min_{\boldsymbol{B}^\top\boldsymbol{f}=\boldsymbol{d}}\left\|\boldsymbol{U}^{-1}\boldsymbol{f}\right\|_q^q \tag{9}$$

where $q=p/(p-1)=1+1/(p-1)$. We refer to the diagonal entries of $\boldsymbol{U}^{-1}$ as q-weights. Again, because we can solve Problem (2) to high-accuracy in near-linear time for $p=\omega(1)$, this allows us to solve Problem (9), the dual flow ℓ_q-norm minimization, in time $p(m^{1+o(1)}+n^{4/3+o(1)})\log^2 1/\varepsilon$.

▶ **Theorem 19** (Flow Algorithmic Result, $q<2$ (Informal)). *Consider a graph G with n vertices and m edges, equipped with positive q-weights, as well as a demand vector. For $1<q<2$, when $q=1+o(1)$, in $\mathrm{poly}\left(\frac{1}{q-1}\right)(m^{1+o(1)}+n^{4/3+o(1)})\log^2 1/\varepsilon$ time, we can compute an ε-approximately optimal flow solution to Problem (9) with high probability.*

Open Questions

Mixed ℓ_2,ℓ_q problems for small $q<2$. In this work, we provided new state-of-the-art algorithms for weighted mixed ℓ_2,ℓ_p-norm minimizing flow and voltage problems for $p>>2$, and for pure ℓ_q-norm minimizing flow and voltage problems for q near 1.

A reasonable definition of mixed ℓ_2,ℓ_q-norm problems for $q<2$ is based on gamma-functions as introduced in [9] and used in [1]. We believe that with minor adjustments to our multiplicative weights solver, these objectives could be handled too, by solving their dual ℓ_2,ℓ_p-gamma function problem for $p>2$.

Directly sparsifying mixed ℓ_2, ℓ_q problems for $q < 2$. A second approach to developing a fast ℓ_2, ℓ_q-gamma function solver for $q < 2$ would be to directly develop sparsification in this setting. We believe this might be possible, and in the general matrix setting might provide better algorithms than alternative approaches.

Removing the $m^{\frac{O(1)}{p-1}}$ loss in sparsification. Our current approaches to graph mixed ℓ_2, ℓ_p-sparsification lose a factor $m^{\frac{O(1)}{p-1}}$ in their quality of approximation, which leads to a $m^{\frac{O(1)}{p-1}}$ factor slowdown in running time, and makes our algorithms less useful for small p. We believe a more sophisticated graph sparsification routine could remove this loss and result in significantly faster algorithms for p close to 2.

Using mixed ℓ_2, ℓ_p-objectives as oracles for ℓ_∞ regression. The current state-of-the-art algorithm for computing maximum flow in unit capacity graphs runs in $\widetilde{O}(m^{4/3})$ time [21], and uses the almost-linear-time algorithm from [25] for solving *unweighted* $\ell_2^2 + \ell_p^p$ instances as a key ingredient.

References

1. Deeksha Adil, Rasmus Kyng, Richard Peng, and Sushant Sachdeva. Iterative refinement for ℓ_p-norm regression. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1405–1424. SIAM, 2019.
2. Deeksha Adil, Richard Peng, and Sushant Sachdeva. Fast, provably convergent irls algorithm for p-norm linear regression. In *Advances in Neural Information Processing Systems*, pages 14189–14200, 2019.
3. Deeksha Adil and Sushant Sachdeva. Faster p-norm minimizing flows, via smoothed q-norm problems. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 892–910. SIAM, 2020.
4. Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network flows - theory, algorithms and applications*. Prentice Hall, 1993.
5. Zeyuan Allen-Zhu, Yuanzhi Li, Rafael Oliveira, and Avi Wigderson. Much faster algorithms for matrix scaling. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 890–901. IEEE, 2017.
6. Ingo Althöfer, Gautam Das, David Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 9(1):81–100, 1993. doi:10.1007/BF02189308.
7. Surender Baswana and Sandeep Sen. A simple and linear time randomized algorithm for computing sparse spanners in weighted graphs. *Random Struct. Algorithms*, 30(4):532–563, 2007.
8. András A. Benczúr and David R. Karger. Approximating s-t minimum cuts in $\tilde{O}(n^2)$ time. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, STOC '96, pages 47–55, New York, NY, USA, 1996. ACM. doi:10.1145/237814.237827.
9. Sébastien Bubeck, Michael B. Cohen, Yin Tat Lee, and Yuanzhi Li. An homotopy method for lp 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, pages 1130–1137, New York, NY, USA, 2018. ACM. doi:10.1145/3188745.3188776.
10. Hui Han Chin, Aleksander Madry, Gary L. Miller, and Richard Peng. Runtime guarantees for regression problems. In *Proceedings of the 4^{th} conference on Innovations in Theoretical Computer Science*, ITCS '13, pages 269–282, New York, NY, USA, 2013. ACM.
11. Paul Christiano, Jonathan A. Kelner, Aleksander Madry, Daniel A. Spielman, and Shang-Hua Teng. Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs. In *Proceedings of the 43rd annual ACM symposium on Theory of computing*, STOC '11, pages 273–282, New York, NY, USA, 2011. ACM. doi:10.1145/1993636.1993674.

12 T. Chu, Y. Gao, R. Peng, S. Sachdeva, S. Sawlani, and J. Wang. Graph sparsification, spectral sketches, and faster resistance computation, via short cycle decompositions. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 361–372, October 2018. doi:10.1109/FOCS.2018.00042.

13 M. B. Cohen, A. Madry, D. Tsipras, and A. Vladu. Matrix scaling and balancing via box constrained newton's method and interior point methods. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 902–913, October 2017. doi:10.1109/FOCS.2017.88.

14 Michael B. Cohen, Yin Tat Lee, Cameron Musco, Christopher Musco, Richard Peng, and Aaron Sidford. Uniform sampling for matrix approximation. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, ITCS '15, page 181–190, New York, NY, USA, 2015. Association for Computing Machinery. doi:10.1145/2688073.2688113.

15 Michael B. Cohen, Yin Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, page 938–942, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3313276.3316303.

16 Michael B. Cohen, Aleksander Madry, Piotr Sankowski, and Adrian Vladu. Negative-weight shortest paths and unit capacity minimum cost flow in $\tilde{O}(m^{10/7} \log w)$ 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*, pages 752–771, 2017.

17 Michael B. Cohen and Richard Peng. ℓ_p row sampling by lewis weights. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '15, page 183–192, New York, NY, USA, 2015. Association for Computing Machinery. doi:10.1145/2746539.2746567.

18 Samuel I. Daitch and Daniel A. Spielman. Faster approximate lossy generalized flow via interior point algorithms. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, STOC '08, pages 451–460, New York, NY, USA, 2008. ACM. Available at arXiv:0803.0988. doi:10.1145/1374376.1374441.

19 David Durfee, John Peebles, Richard Peng, and Anup B. Rao. Determinant-preserving sparsification of SDDM matrices with applications to counting and sampling spanning trees. In *FOCS*, pages 926–937. IEEE Computer Society, 2017.

20 Andrew V. Goldberg and Robert Endre Tarjan. Efficient maximum flow algorithms. *Commun. ACM*, 57(8):82–89, 2014.

21 Tarun Kathuria, Yang P. Liu, and Aaron Sidford. Unit capacity maxflow in almost $m^{4/3}$ time. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 119–130, 2020. doi:10.1109/FOCS46700.2020.00020.

22 Jonathan A. Kelner, Yin Tat Lee, Lorenzo Orecchia, and Aaron 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*, pages 217–226, 2014.

23 Ioannis Koutis, Gary L. Miller, and Richard Peng. A nearly-$m \log n$ time solver for SDD linear systems. In *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, FOCS '11, pages 590–598, Washington, DC, USA, 2011. IEEE Computer Society. doi:10.1109/FOCS.2011.85.

24 Rasmus Kyng, Yin Tat Lee, Richard Peng, Sushant Sachdeva, and Daniel A Spielman. Sparsified cholesky and multigrid solvers for connection laplacians. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing*, pages 842–850. ACM, 2016.

25 Rasmus Kyng, Richard Peng, Sushant Sachdeva, and Di Wang. Flows in almost linear time via adaptive preconditioning. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, pages 902–913, New York, NY, USA, 2019. ACM. doi:10.1145/3313276.3316410.

26 Rasmus Kyng, Anup Rao, Sushant Sachdeva, and Daniel A. Spielman. Algorithms for lipschitz learning on graphs. In Peter Grünwald, Elad Hazan, and Satyen Kale, editors, *Proceedings of The 28th Conference on Learning Theory*, volume 40 of *Proceedings of Machine Learning Research*, pages 1190–1223, Paris, France, July 2015. PMLR.

27 Y. T. Lee and A. Sidford. Path finding methods for linear programming: Solving linear programs in $\tilde{O}(\text{vrank})$ iterations and faster algorithms for maximum flow. In *FOCS*, 2014.

28 Yang P. Liu and Aaron 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*, pages 803–814. ACM, 2020. doi:10.1145/3357713.3384247.

29 A. Madry. Navigating central path with electrical flows: From flows to matchings, and back. In *FOCS*, 2013.

30 Aleksander Madry. Fast approximation algorithms for cut-based problems in undirected graphs. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 245–254. IEEE, 2010.

31 Y. Nesterov and A. Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming*. Society for Industrial and Applied Mathematics, 1994. doi:10.1137/1.9781611970791.

32 Lorenzo Orecchia, Sushant Sachdeva, and Nisheeth K. Vishnoi. Approximating the exponential, the lanczos method and an $\tilde{O}(m)$-time spectral algorithm for balanced separator. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, STOC '12, pages 1141–1160, New York, NY, USA, 2012. ACM. doi:10.1145/2213977.2214080.

33 David Peleg and Alejandro A. Schäffer. Graph spanners. *Journal of Graph Theory*, 13(1):99–116, 1989. doi:10.1002/jgt.3190130114.

34 Richard Peng and Daniel A. Spielman. An efficient parallel solver for SDD linear systems. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, STOC '14, pages 333–342, New York, NY, USA, 2014. ACM.

35 Harald Racke. Optimal hierarchical decompositions for congestion minimization in networks. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, STOC '08, pages 255–264, New York, NY, USA, 2008. ACM.

36 Harald Racke, Chintan Shah, and Hanjo Taubig. Computing cut-based hierarchical decompositions in almost linear time. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '14, pages 227–238, 2014.

37 Alexander Schrijver. On the history of the transportation and maximum flow problems. *Math. Program.*, 91(3):437–445, 2002.

38 Jonah 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*, pages 263–269, 2013.

39 Jonah Sherman. Generalized preconditioning and undirected minimum-cost flow. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '17, pages 772–780, 2017.

40 D. Spielman and N. Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011. doi:10.1137/080734029.

41 D. Spielman and S. Teng. Spectral sparsification of graphs. *SIAM Journal on Computing*, 40(4):981–1025, 2011. doi:10.1137/08074489X.

42 D.A. Spielman and S. Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *STOC*, 2004.

43 Daniel A. Spielman. Spectral graph theory lectures: Sparsification by effective resistance sampling, 2015.

44 Jan van den Brand, Yin-Tat Lee, Danupon Nanongkai, Richard Peng, Thatchaphol Saranurak, Aaron Sidford, Zhao Song, and Di Wang. Bipartite matching in nearly-linear time on moderately dense graphs. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 919–930. IEEE, 2020.