Flows in Almost Linear Time via Adaptive Preconditioning

Rasmus Kyng[*†]
Harvard

Richard Peng[*‡]
Georgia Tech /
MSR Redmond

Sushant Sachdeva[*§]
UToronto

Di Wang[*‡]
Georgia Tech

June 26, 2019

Abstract

We present algorithms for solving a large class of flow and regression problems on unit weighted graphs to $(1 + 1/poly(n))$ accuracy in almost-linear time. These problems include ℓ_p-norm minimizing flow for p large ($p \in [\omega(1), o(\log^{2/3} n)]$), and their duals, ℓ_p-norm semi-supervised learning for p close to 1.

As p tends to infinity, ℓ_p-norm flow and its dual tend to max-flow and min-cut respectively. Using this connection and our algorithms, we give an alternate approach for approximating undirected max-flow, and the first almost-linear time approximations of discretizations of total variation minimization objectives.

This algorithm demonstrates that many tools previous viewed as limited to linear systems are in fact applicable to a much wider range of convex objectives. It is based on the the routing-based solver for Laplacian linear systems by Spielman and Teng (STOC '04, SIMAX '14), but require several new tools: adaptive non-linear preconditioning, tree-routing based ultra-sparsification for mixed ℓ_2 and ℓ_p norm objectives, and decomposing graphs into uniform expanders.

[*]Emails: {rjkyng,richard.peng}@gmail.com, sachdeva@cs.toronto.edu, di.wang@cc.gatech.edu

[†]Supported by ONR grant N00014-18-1-2562.

[‡]Supported in part by the National Science Foundation under Grant No. 1718533.

[§]Supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), and a Connaught New Researcher award.

Contents

1 Introduction **1**
- 1.1 Main Results and Applications . . . 2
  - 1.1.1 ℓ_p-Norm Flows . . . 2
  - 1.1.2 Semi-Supervised Learning on Graphs. . . . 3
  - 1.1.3 Use as Oracle in Conjunction with Multiplicative Weight Updates . . . 3
- 1.2 Related Work . . . 4
- 1.3 Overview . . . 5
- 1.4 Open Questions . . . 6

2 Preliminaries **7**
- 2.1 Smoothed ℓ_p-norm functions . . . 7
- 2.2 Flow Problems and Approximation . . . 7
- 2.3 Approximating Smoothed p-norm Instances . . . 8
- 2.4 Orthogonal Decompositions of Flows . . . 9

3 Numerical Methods **10**
- 3.1 Iterative Refinement . . . 10
- 3.2 Vertex Elimination . . . 12
- 3.3 Recursive Preconditioning . . . 13

4 Graph Theoretic Preconditioners **18**
- 4.1 Tree-Portal Routing . . . 19
- 4.2 Partitioning Trees into Subtrees and Portals . . . 21
- 4.3 Graph Sparsification . . . 23
- 4.4 Ultra-sparsification Algorithm and Error Analysis . . . 25

5 Decomposing into Uniform Expanders **35**

Acknowledgements **38**

A Deferred Proofs from Prelims, Section 2.2 **43**

B Deferred Proofs for Numerical Methods from Section 3 **45**

C Elimination of Low-Degree Vertices, and Loops **49**

D Sparsifying Uniform Expanders **53**

E Using Approximate Projections **63**

F ℓ_p-norm Semi-Supervised Learning on Graphs. **64**

1 Introduction

Graphs are among the most ubiquitous representations of data, and efficiently computing on graphs is a task central to operations research, machine learning, and network science. Among graph algorithms, network flows have been extensively studied [EK72, Kar73, ET75, GT88, GR98, Sch02, Hoc08, CKM+11, HO13, Orl13, GT14], and have wide ranges of applications [KBR07, LSBG13, PZZ13]. Over the past decade, the 'Laplacian paradigm' of designing graph algorithms spurred a revolution in the best run-time upper bounds for many fundamental graph optimization problems. Many of these new graph algorithms incorporated numerical primitives: even for the s-t shortest path problem in graphs with negative edge weights, the current best running times [CMTV17] are from invoking linear system solvers.

This incorporation of numerical routines [DS08, CKM+11] in turn led to a dependence on ϵ, the approximation accuracy. While maximum flow and transshipment problems on undirected graphs can now be approximated in nearly-linear time [KLOS14, She13, BKKL17, Pen16, She17b, She17a] (and the distributed setting has also been studied [GKK+15, BKKL17]), these algorithms are *low accuracy* in that their running times have factors of $1/\epsilon$ or higher. This is in contrast to *high accuracy* solvers for linear systems and convex programs, which with $polylog(n)$ overhead give $1/poly(n)$-approximate solutions. Prior to our result, such *high accuracy* runtime bounds for problems beyond linear systems all utilize second order methods [DS08, Mad13, LS14, CMTV17, ALdOW17, BCLL18] from convex optimization.

The main contribution of this paper is giving almost-linear time, *high accuracy* solutions to a significantly wider range of graph optimization problems that can be viewed as interpolations between maximum flow, shortest paths, and graph-structured linear systems. Our unified formulation of these problems is based on the following unified formulation of flow/path problems as norm minimization over a demand vector $\boldsymbol{b} \in \mathbb{R}^V_{\geq 0}$:

$$\min_{\text{flow } \boldsymbol{f} \text{ with residue } \boldsymbol{b}} \|\boldsymbol{f}\|_{\odot}. \tag{1}$$

In particular, when $\|\cdot\|_{\odot}$ is the ℓ_∞-norm, this formulation is equivalent finding the flow of minimum congestion, which is in turn equivalent to computing maximum flows and bipartite matchings in unit capacitated graphs [Mad11]. Our main result is that for any $p \geq 2$, given weights $\boldsymbol{r} \in \mathbb{R}^E_{\geq 0}$, a "gradient" $\boldsymbol{g} \in \mathbb{R}^E$, and a demand vector $\boldsymbol{b} \in \mathbb{R}^V$ (with $\boldsymbol{b}^\top \mathbf{1} = 0$), we can solve

$$\min_{\text{flow } \boldsymbol{f} \text{ with residue } \boldsymbol{b}} \sum_e \boldsymbol{g}_e \boldsymbol{f}_e + \boldsymbol{r}_e \boldsymbol{f}_e^2 + |\boldsymbol{f}_e|^p, \tag{2}$$

to $1/\mathrm{poly}(n)$ additive error in time $2^{O(p^{3/2})} m^{1+O(\frac{1}{\sqrt{p}})}$. We will formally state this result as Theorem 1.1 at the start of Section 1.1, and discuss several of its applications in flows, semi-supervised learning, and total variation minimization.

We believe that our algorithm represents a new approach to designing *high accuracy* solvers for graph-structured optimization problems. A brief survey of relevant works is in Section 1.2: previous *high accuracy* algorithms treat linear systems as the separation between graph theoretic and numerical components: the outer loop adjusts the numerics, while the inner loop quickly solves the resulting linear systems using the underlying graph structures. Our result, in contrast, directly invoke analogs of linear system solving primitives to the non-linear (but still convex) objective functions, and no longer has this clear separation between graph theoretic and numerical components.

We will overview key components of our approach, as well as how they are combined, in Section 1.3. Discussions of possible avenues for addressing shortcomings of our result, namely the exponential dependence on p, the restriction to unweighted graphs, and gap between $\ell_{\sqrt{\log n}}$-norm flow and ℓ_∞ are in Section 1.4.

1.1 Main Results and Applications

The formal formulation of our problem relies on the following objects defined on a graph $G = (V, E)$ with n vertices and m edges:

1. edge-vertex incidence matrix $\boldsymbol{B}$,
2. a vector $\boldsymbol{b}$ indicating the required residues on vertices (satisfying $\mathbf{1}^T \boldsymbol{b} = 0$), and
3. a norm p as finding a flow $\boldsymbol{f}$ with demands $\boldsymbol{b}$ that minimize a specified norm $\|\cdot\|$.

The normed flow problem that we solve can then be formulated as:

$$\min_{\boldsymbol{B}^\top \boldsymbol{f} = \boldsymbol{b}} \sum_e \boldsymbol{g}_e \boldsymbol{f}_e + \boldsymbol{r}_e \boldsymbol{f}_e^2 + |\boldsymbol{f}_e|^p, \tag{2}$$

Using $\|\boldsymbol{f}\|_{2,\boldsymbol{r}} = \sqrt{\sum_e \boldsymbol{r}_e \boldsymbol{f}_e^2}$ to denote the $\boldsymbol{r}$-weighted 2-norm, the objective can also be viewed as $\boldsymbol{g}^\top \boldsymbol{f} + \|\boldsymbol{f}\|_{2,\boldsymbol{r}}^2 + \|\boldsymbol{f}\|_p^p$. Let $\mathrm{val}(\boldsymbol{f})$ denote value of a flow $\boldsymbol{f}$ according to the above objective, and let OPT denote value of the optimal solution to Problem (2). Our main technical result is the following statement which we prove as corollary of our main technical theorem in Section 3.3.

Theorem 1.1 (Smoothed ℓ_p-norm flows)**.** *For any $p \geq 2$, given weights $\boldsymbol{r} \in \mathbb{R}_{\geq 0}^E$, a "gradient" $\boldsymbol{g} \in \mathbb{R}^E$, a demand vector $\boldsymbol{b} \in \mathbb{R}^V$ (with $\boldsymbol{b}^\top \mathbf{1} = 0$), and an initial solution $\boldsymbol{f}^{(0)}$ such that all parameters are bounded by $2^{poly(\log n)}$, we can compute a flow $\widetilde{\boldsymbol{f}}$ satisfying demands $\boldsymbol{b}$,* i.e.*, $\boldsymbol{B}^{G\top} \widetilde{\boldsymbol{f}} = \boldsymbol{b}$, such that*

$$val(\widetilde{\boldsymbol{f}}) - OPT \leq \frac{1}{poly(m)} \left(val(\boldsymbol{f}^{(0)}) - OPT \right) + \frac{1}{poly(m)}$$

in $2^{O(p^{3/2})} m^{1+O(\frac{1}{\sqrt{p}})}$ time, where m denotes the number of edges in G.

1.1.1 ℓ_p-Norm Flows

From this, we also get a (slightly simpler) statement about ℓ_p-norm flows.

Theorem 1.2 (ℓ_p-norm flows)**.** *For any $p \geq 2$, given an unweighted graph $G(V, E)$ and demands $\boldsymbol{b}$, using the routine* pFlows$(\mathcal{G}, \boldsymbol{b})$ *(Algorithm 2) we can compute a flow $\widetilde{\boldsymbol{f}}$ satisfying satisfying $\boldsymbol{b}$,* i.e.*, $\boldsymbol{B}^{G\top} \widetilde{\boldsymbol{f}} = \boldsymbol{b}$, such that*

$$\left\| \widetilde{\boldsymbol{f}} \right\|_p^p \leq \left(1 + \frac{1}{poly(m)} \right) \min_{\boldsymbol{f} : \boldsymbol{B}^{G\top} \boldsymbol{f} = \boldsymbol{b}} \|\boldsymbol{f}\|_p^p.$$

in $2^{O(p^{3/2})} m^{1+O(\frac{1}{\sqrt{p}})}$, time, where m denotes the number of edges in G.

This corollary is also proven in Section 3.3.

Picking $\boldsymbol{g}, \boldsymbol{r} = \mathbf{0}$ gives us an $2^{O(p^{3/2})} m^{1+O(\frac{1}{\sqrt{p}})}$ time high-accuracy algorithm for ℓ_p-norm minimizing flows on unit weighted undirected graphs ($p \geq 2$). For large p, e.g. $p = \sqrt{\log n}$ this is an $m^{1+o(1)}$ time algorithm, and to our knowledge the first almost linear time high-accuracy algorithm for a flow problem other than Laplacian solvers (ℓ_2) or shortest-paths (ℓ_1).

1.1.2 Semi-Supervised Learning on Graphs.

Semi-supervised learning on graphs in machine learning is often based on solving an optimization problem where voltages (labels) are fixed at some vertices in a graph the voltages at remaning nodes are chosen so that some overall objective is minimized (e.g. ℓ_p-norm of the vector of voltage differences across edges) [AL11, KRSS15, EACR+16]. Formally, given a graph $G = (V, E)$ and a labelled subset of the nodes $T \subset V$ with labels $\boldsymbol{s}_T \in \mathbb{R}^T$, we can write the problem as

$$\min_{\boldsymbol{x} \in \Re^V | \boldsymbol{x}_T = \boldsymbol{s}_T} \sum_{u \sim v} |\boldsymbol{x}_u - \boldsymbol{x}_v|^p. \tag{3}$$

By converting this problem to its dual, we get an almost linear time algorithm for solving it to high accuracy, provided the initial voltage problem uses p close to 1: In this case, voltage solutions are "cut-like". Given $p < 2$, we get a solver that computes a $(1 + 1/\text{poly}(m))$ multiplicative accuracy solution in time $2^{O((\frac{1}{p-1})^{3/2})} m^{1+O(\sqrt{p-1})}$. For $p = 1 + \frac{1}{\sqrt{\log n}}$, this is time is bounded by $m^{1+o(1)}$.

Converting the dual of Problem (3) into a form solvable by our algorithms requires a small transformation, which we describe in Appendix F.

1.1.3 Use as Oracle in Conjunction with Multiplicative Weight Updates

The mixed ℓ_2^2 and ℓ_p^p objective in our Problem (2) is useful for building oracles to use in multiplicative weight update algorithms based on flows, as they appear in [CKM+11, AKPS19]. Assume we are looking to solve some problem to $(1 + \epsilon)$-accuracy measured as multiplicative error, and let us assume $\frac{1}{\text{poly}(m)} < \epsilon < 0.5$. Specifically we can solve for the following objective subject to certain linear constraints.

$$\sum_e \boldsymbol{r}_e \boldsymbol{f}_e^2 + \frac{\epsilon \|\boldsymbol{r}\|_1}{m} |\boldsymbol{f}_e|^p. \tag{4}$$

This gives an oracle for several problems. Algorithms based on oracle solutions to this type of objective work by noting that any $\boldsymbol{f}$ with $\|\boldsymbol{f}\|_\infty \leq 1$ gives an objective value at most

$$\sum_e \boldsymbol{r}_e \boldsymbol{f}_e^2 + \frac{\epsilon \|\boldsymbol{r}\|_1}{m} |\boldsymbol{f}_e|^p \leq (1 + \epsilon) \|\boldsymbol{r}\|_1.$$

Since such a flow must exist in the context where the oracle is applied, the optimum flow must also meet this bound. Now, if we compute a $(1 + 0.01\epsilon)$ approximately optimal solution to this problem, it must satisfy

$$\sum_e \boldsymbol{r}_e \boldsymbol{f}_e^2 + \frac{\epsilon \|\boldsymbol{r}\|_1}{m} |\boldsymbol{f}_e|^p \leq (1 + 1.1\epsilon) \|\boldsymbol{r}\|_1.$$

By Cauchy-Schwarz, we get $\sum_e \boldsymbol{r}_e |\boldsymbol{f}_e| \leq \sqrt{\|\boldsymbol{r}\|_1 \sum_e \boldsymbol{r}_e \boldsymbol{f}_e^2} \leq (1 + 1.1\epsilon) \|\boldsymbol{r}\|_1$, which tells us the oracle is "good-on-average" according to the weights $\boldsymbol{r}$. The objective value also implies for every edge that

$$\frac{\epsilon \|\boldsymbol{r}\|_1}{m} |\boldsymbol{f}_e|^p \leq (1 + 1.1\epsilon) \|\boldsymbol{r}\|_1 \leq 2 \|\boldsymbol{r}\|_1,$$

which simplifies to:

$$|\boldsymbol{f}_e| \leq (m/\epsilon)^{1/p} \leq m^{o(1)} \tag{5}$$

when we set $p = \log^{0.1} n$. This is the width of the oracle, and together these conditions demonstrate that the oracle suffices for a multiplicative weights algorithm and bounds the number of calls to the oracle by $m^{o(1)}\text{poly}(1/\epsilon)$.

This oracle has multiple uses:

Approximate undirected maximum flow. Using the oracle, we can approximate maximum flow using [CKM+11], giving an algorithm for undirected maximum flow that is not based on oblivious routings unlike other fast algorithms for approximate maximum flow [She13, KLOS14, Pen16]. Our algorithm obtains almost-linear time, albeit only for unit weighted graphs.

Isotropic total variation denoising. Using our algorithm, we can give the first almost linear time, low accuracy algorithm for total variation denoising on unit weighted graphs [ROF92, ZWC10]. While there has been significant advances in image processing since the introduction of this objective, it still remains a representative objective in pixel vision tasks. The total variation objectives can be viewed as variants of semi-supervised learning on graphs: Given a "signal" vector $\boldsymbol{s}$ which corresponds to noisy observations of pixels of an image, we want to find a denoised version of $\boldsymbol{s}$, which we refer to as $\boldsymbol{x}$. The denoised output $\boldsymbol{x}$ should minimize an objective that measures both the between pixels in $\boldsymbol{x}$ that are close to each other in the image (which should be small), and the difference between $\boldsymbol{x}$ and $\boldsymbol{s}$ (which should also be small). The most popular version of this problem, known as isotropic total variation denoising, allows the input to specify a collection of groups of pixels with connections inside each group i given by a set of edges E_i, and asks that 1) the denoised pixels are close in an ℓ_2 sense to the measured signal, 2) in each group, the standard deviation between denoised pixels is not too high. These goals are expressed in the objective

$$\sum_u (\boldsymbol{x}_u - \boldsymbol{s}_u)^2 + \sum_i \sqrt{\sum_{e \in E_i} (\boldsymbol{x}_u - \boldsymbol{x}_v)^2}.$$

The dual of this problem is grouped flows, which is finding $\boldsymbol{f}$ such that $\boldsymbol{B}^\top \boldsymbol{f} = \boldsymbol{d}$ and for edge sets E_i,

$$\left\|\boldsymbol{f}_{E_i}\right\|_2^2 \leq 1.$$

Our oracle gives the first routine for approximate isotropic TV denoising that runs in almost linear time. The previous best running time was about $m^{4/3}$ [CMMP13].

1.2 Related Work

Network flow problems have a long history of motivating broader developments in algorithms, including the introduction of strongly polynomial time as a benchmark of algorithmic efficiency [Edm65, EK72], the development of tree data structures [GN79, ST83, ST85], and randomized graph algorithms and graph approximations [KS96, BK96]. For general capacities, the best strongly polynomial time algorithms run in about quadratic time due to the flow decomposition barrier [EK72, GN79, GT88, HO13, Orl13], which says that the there exists graphs where the path decomposition of an optimum flow must have quadratic size.

The flow decomposition barrier suggest that sub-quadratic time algorithms for network flows should decompose solutions numericallly, and this has indeed been the case in the development of such algorithms [GR98, GT14]. These numerical approaches recently culminated in nearly-linear time algorithms for undirected maximum flow and transshipment (the ℓ_1 case of Problem (1)), yielding nearly-linear time algorithms [CKM$^+$11, She13, KLOS14, Pen16, She17b, She17a]. Much of these progress were motivated by the development of nearly-linear time high-accuracy solvers for Laplacian linear systems [ST14, KMP12, KOSZ13, LS13, KS16], whose duals, electrical flows are the ℓ_∞ case of Problem (1). Such solvers can in turn be used to give the current best high accuracy flow algorithms. For graphs with polynomially bounded capacities, the current best running time is $\widetilde{O}(m\sqrt{n})$ due to Lee and Sidford [LS14]. On sparse graphs, this bound still does not break the long-standing $O(n^{1.5})$ barrier dating back to the early 70s [HK73, Kar73, ET75]. Recently Madry [Mad13, Mad16] broke this barrier on unit capacitated graphs, obtaining $\widetilde{O}(m^{10/7})$ running time.

Our result has in common with all previous results on almost-linear time optimization problems on graphs [KLOS14, She13, BKKL17, Pen16, She17b, She17a] in that it is based on white-box modifications of a linear system solver. In particular, our high level algorithmic strategy in creating edge and vertex reductions is identical to the first nearly-linear time solver by Spielman and Teng [ST14]. Much of this similarity is due to the lack of understanding of more general versions of key components: some possibilities for simplifying the result will be discussed in Section 1.4. On the other hand, our algorithms differ from previous adaptations of solvers in that it obtains high accuracy [1]. This requires us to tailor the scheme to the residual problems from the p-norm iterative methods, and results in us taking a more numerical approach, instead of the more routing and path embedding-based approaches utilized in similar adaptations of Spielman and Teng [ST14] to cuts [Mad10], flows [She13, KLOS14], and shortest paths [BKKL17].

The development of high-accuracy algorithms for p-norm minimization that are faster than interior point methods (IPMs) [NN94] was pioneered by the recent work of Bubeck *et al.* [BCLL18] which introduced the γ-functions that were also used in [AKPS19]. However, the methods in [BCLL18] are conceptually similar to interior point methods (IPMs) [NN94] (as in they are *homotopy* methods). Their runtime for large p behaves essentially like IPMs, requiring about $m^{3/2-o(1)}$ time for solving p-norm flow problems, whereas the limiting behavior of our result is about $m^{1+o(1)}$.

1.3 Overview

At a high level, our approach can be viewed as solving a graph optimization problem as a linear system. This is done by combining the numerical methods for ℓ_p-norms by Adil *et al.* [AKPS19] with the recursive preconditioning of graph structured linear systems by Spielman and Teng [ST14]. Many conceptual obstacles arise in trying to realize this vision, preventing us from adopting later Laplacian linear solvers that have greatly simplified the result of Spielman and Teng. The main one is the lack of concentration theory for the smoothed p-norm objectives integral to our algorithms: these concentration arguments are at the core of all subsequent improvements to Laplacian solver algorithms [KMP11, KOSZ13, LPS15, KS16].

Our starting point is a recent result involving a subset of the authors [AKPS19] that significantly generalized the phenomenon of *high-accuracy* numerical methods. In particular, this method is applicable to general ℓ_p-norm optimization problems, for all p that are bounded away from 1 and ∞. It also opens up a major question: can we develop an appropriate notion of *preconditioning*, the

[1] The nearly-linear time matrix scaling algorithm [CMTV17] has a linear dependence on the condition number κ, while convex optimization methods for matrix scaling have dependencies of $\log \kappa$ instead.

other central ingredient of fast solvers for linear systems, applicable to ℓ_p-norms? We resolve this question in the affirmative, and develop a theory of preconditioning that works for a wide class of non-linear problems in Section 3. In particular, we show that the second and p^{th} order terms from the main formulation in Equation 2 form a class of functions that's closed under taking residual problems. We will formally define these as smoothed ℓ_p-norms in Section 2.1.

The crux of our problem then becomes designing preconditioners that interact well with these smoothed ℓ_p-norms. Here it's worth noting that earlier works on preconditioning for non-linear (maximum) flow problems all relied on *oblivious routing* which gives rise to linear preconditioners. Such an approach encounters a significant obstacle with ℓ_p norms: consider changing a single coordinate from, say 1, to $(1 + \delta)$:

- If the update δ is much smaller than 1 in absolute value, the change in the objective from 1^p to $(1 + \delta)^p$ is dominated by terms that scale as δ and δ^2.
- However, if the update is much larger than 1, the change is dominated by a δ^p term.

This means that good preconditioning across small and large updates is inherently highly dependent on the current coordinate value.

This example captures the core difficulties of our preconditioned iterative methods for smoothed ℓ_p-norm problems, which heavily rely on both the second and pth power terms the objective functions. It means our graph theoretic components must simultaneously control terms of different degrees (namely scaling as δ, δ^2, and δ^p) related to the flows on graphs. Here our key idea is that unit-weighted graphs have "multi-objective low-stretch trees" that simultaneously preserve the δ^2 and δ^p terms, while the linear (gradient) terms can be preserved exactly when routing along these trees. Here a major difficulty is that the tree depends on the second order derivatives of the current solution point, and thus must continuously change as the algorithm proceeds. Additionally, after rerouting graph edges along the tree, we need to sparsify the graph according to weights defined by the same second derivatives at the current solution, which makes the adaptive aspect of the algorithm even more important. We defer the construction of our adaptive preconditioner to Section 4, after first formally defining our objective functions in Section 2, and introducing numerical methods based on them in Section 3.

1.4 Open Questions

We expect that our algorithm can be greatly simplified and adapted to non-unit weight graphs in ways similar to the sampling based solvers for Laplacian linear systems [KMP14, KOSZ13, KS16]. The current understanding of concentration theory for ℓ_p norms rely heavily on tools from functional analysis [CP15]: generalizing these tools to smoothed ℓ_p-norm objectives is beyond the scope of this paper.

A major limitation of our result is the restriction to unit capacitated graphs. We believe this limitation is inherent to our approach of constructing preconditioners from trees: for general weights, there are cases where no tree can simultaneously have small stretch w.r.t. ℓ_2-norm and ℓ_p-norm weights. We believe that by developing a more complete theory of elimination and sparsification for these objectives, it will be possible to sparsify non-unit weight graphs, and develop solvers following the patterns of non-tree based Laplacian solvers [PS14, LPS15, KS16].

We also believe that the overall algorithmic approach established here is applicable far beyond the class of objective functions studied in this paper. Here a direct question is whether the dependency on p can be improved to handling ℓ_m flows, which in unit weighted graphs imply maximum flows. The exponential dependence on p has already been shown to be improvable to about $\widetilde{O}(p^2)$ [Sac19]. For even larger values of p, a natural approach is to use homotopy methods that modify the p values gradually. Here it is also plausible that our techniques, or their possible generalizations to weighted cases, can be used as algorithmic building blocks.

2 Preliminaries

2.1 Smoothed ℓ_p-norm functions

We consider p-norms smoothed by the addition of a quadratic term. First we define such a smoothed p^{th}-power on $\mathbb{R}$.

Definition 2.1 (Smoothed p^{th}-power)**.** Given $r, x \in \mathbb{R}, r \geq 0$ define the r-smoothed s-weighted p^{th}-power of x to be

$$h_p(r, s, x) = rx^2 + s|x|^p.$$

This definition can be naturally extended to vectors to obtained smoothed ℓ_p-norms.

Definition 2.2 (Smoothed ℓ_p-norm)**.** Given vectors $\boldsymbol{x} \in \mathbb{R}^m, \boldsymbol{r} \in \mathbb{R}^m_{\geq 0}$, and a positive scalar $s \in \mathbb{R}_{\geq 0}$, define the $\boldsymbol{r}$-smooth s-weighted p-norm of $\boldsymbol{x}$ to be

$$h_p(\boldsymbol{r}, s, \boldsymbol{x}) = \sum_{i=1}^{m} h_p(\boldsymbol{r}_i, s, \boldsymbol{x}_i) = \sum_{i=1}^{m} (\boldsymbol{r}_i \boldsymbol{x}_i^2 + s|\boldsymbol{x}_i|^p).$$

2.2 Flow Problems and Approximation

We will consider problems where we seek to find flows minimizing smoothed p-norms. We first define these problem instances.

Definition 2.3 (Smoothed p-norm instance)**.** A *smoothed p-norm instance* is a tuple $\mathcal{G}$,

$$\mathcal{G} \stackrel{\text{def}}{=} (V^{\mathcal{G}}, E^{\mathcal{G}}, \boldsymbol{g}^{\mathcal{G}}, \boldsymbol{r}^{\mathcal{G}}, s^{\mathcal{G}}),$$

where $V^{\mathcal{G}}$ is a set of vertices, $E^{\mathcal{G}}$ is a set of undirected edges on $V^{\mathcal{G}}$, the edges are accompanied by a gradient, specified by $\boldsymbol{g}^{\mathcal{G}} \in \mathbb{R}^{E^{\mathcal{G}}}$, the edges have ℓ_2^2-resistances given by $\boldsymbol{r}^{\mathcal{G}} \in \mathbb{R}^{E^{\mathcal{G}}}_{\geq 0}$, and $s \in \mathbb{R}_{\geq 0}$ gives the p-norm scaling.

Definition 2.4 (Flows, residues, and circulations)**.** Given a smoothed p-norm instance $\mathcal{G}$, a vector $\boldsymbol{f} \in \mathbb{R}^{E^{\mathcal{G}}}$ is said to be a flow on $\mathcal{G}$. A flow vector $\boldsymbol{f}$ satisfies residues $\boldsymbol{b} \in \mathbb{R}^{V^{\mathcal{G}}}$ if $\left(\boldsymbol{B}^{\mathcal{G}}\right)^{\top} \boldsymbol{f} = \boldsymbol{b}$, where $\boldsymbol{B}^{\mathcal{G}} \in \mathbb{R}^{E^{\mathcal{G}} \times V^{\mathcal{G}}}$ is the edge-vertex incidence matrix of the graph $(V^{\mathcal{G}}, E^{\mathcal{G}})$, *i.e.*, $\left(\boldsymbol{B}^{\mathcal{G}}\right)^{\top}_{(u,v)} = \mathbf{1}_u - \mathbf{1}_v$.

A flow $\boldsymbol{f}$ with residue $\mathbf{0}$ is called a circulation on $\mathcal{G}$.

Note that our underlying instance and the edges are undirected. However, for every undirected edge $e = (u, v) \in E$, we assign an arbitrary fixed direction to the edge, say $u \to v$, and interpret $\boldsymbol{f}_e \geq 0$ as flow in the direction of the edge from u to v, and $\boldsymbol{f}_e < 0$ as flow in the reverse direction. For convenience, we assume that for any edge $(u, v) \in E$, we have $\boldsymbol{f}_{(u,v)} = -\boldsymbol{f}_{(v,u)}$.

Definition 2.5 (Objective, $\mathcal{E}^{\mathcal{G}}$)**.** Given a smoothed p-norm instance $\mathcal{G}$, and a flow $\boldsymbol{f}$ on $\mathcal{G}$, the associated objective function, or the energy, of $\boldsymbol{f}$ is given by

$$\mathcal{E}^{\mathcal{G}}(\boldsymbol{f}) = \left(\boldsymbol{g}^{\mathcal{G}}\right)^{\top}\boldsymbol{f} - h_p(\boldsymbol{r}, s, \boldsymbol{f}).$$

Definition 2.6 (Smoothed p-norm flow / circulation problem)**.** Given a smoothed p-norm instance $\mathcal{G}$ and a residue vector $\boldsymbol{b} \in \mathbb{R}^{E^{\mathcal{G}}}$, the *smoothed p-norm flow problem* $(\mathcal{G}, \boldsymbol{b})$, finds a flow $\boldsymbol{f} \in \mathbb{R}^{E^{\mathcal{G}}}$ with residues $\boldsymbol{b}$ that maximizes $\mathcal{E}^{\mathcal{G}}(\boldsymbol{f})$, *i.e.*,

$$\max_{\boldsymbol{f}:(\boldsymbol{B}^{\mathcal{G}})^{\top}\boldsymbol{f}=\boldsymbol{b}} \mathcal{E}^{\mathcal{G}}(\boldsymbol{f}).$$

If $\boldsymbol{b} = \boldsymbol{0}$, we call it a *smoothed p-norm circulation problem.*

Note that the optimal objective of a smoothed p-norm circulation problem is always non-negative, whereas for a smoothed p-norm flow problem, it could be negative.

2.3 Approximating Smoothed p-norm Instances

Since we work with objective functions that are non-standard (and not even homogeneous), we need to carefully define a new notion of approximation for these instances.

Definition 2.7 ($\mathcal{H} \preceq_{\kappa} \mathcal{G}$)**.** For two smoothed p-norm instances, $\mathcal{G}, \mathcal{H}$, we write $\mathcal{H} \preceq_{\kappa} \mathcal{G}$ if there is a linear map $\mathcal{M}_{\mathcal{H}\to\mathcal{G}} : \mathbb{R}^{E^{\mathcal{H}}} \to \mathbb{R}^{E^{\mathcal{G}}}$ such that for every flow $\boldsymbol{f}^{\mathcal{H}}$ on $\mathcal{H}$, we have that $\boldsymbol{f}^{\mathcal{G}} = \mathcal{M}_{\mathcal{H}\to\mathcal{G}}(\boldsymbol{f}^{\mathcal{H}})$ is a flow on $\mathcal{G}$ such that

1. $\boldsymbol{f}^{\mathcal{G}}$ has the same residues as $\boldsymbol{f}^{\mathcal{H}}$ *i.e.*, $(\boldsymbol{B}^{\mathcal{G}})^{\top}\boldsymbol{f}^{\mathcal{G}} = (\boldsymbol{B}^{\mathcal{H}})^{\top}\boldsymbol{f}^{\mathcal{H}}$, and
2. has energy bounded by:
$$\frac{1}{\kappa}\mathcal{E}^{\mathcal{H}}(\boldsymbol{f}^{\mathcal{H}}) \leq \mathcal{E}^{\mathcal{G}}\left(\frac{1}{\kappa}\boldsymbol{f}^{\mathcal{G}}\right).$$

For some of our transformations on graphs, we will be able to prove approximation guarantees only for circulations. Thus, we define the following notion restricted to circulations.

Definition 2.8 ($\mathcal{H} \preceq_{\kappa}^{\text{cycle}} \mathcal{G}$)**.** For two smoothed p-norm instances, $\mathcal{G}, \mathcal{H}$, we write $\mathcal{H} \preceq_{\kappa}^{\text{cycle}} \mathcal{G}$ if there is a linear map $\mathcal{M}_{\mathcal{H}\to\mathcal{G}} : \mathbb{R}^{E^{\mathcal{H}}} \to \mathbb{R}^{E^{\mathcal{G}}}$ such that for any circulation $\boldsymbol{f}^{\mathcal{H}}$ on $\mathcal{H}$, *i.e.*, $(\boldsymbol{B}^{\mathcal{H}})^{\top}\boldsymbol{f}^{\mathcal{H}} = \boldsymbol{0}$, the flow $\boldsymbol{f}^{\mathcal{G}} = \mathcal{M}_{\mathcal{H}\to\mathcal{G}}(\boldsymbol{f}^{\mathcal{H}})$ is a circulation, *i.e.*, $(\boldsymbol{B}^{\mathcal{G}})^{\top}\boldsymbol{f}^{\mathcal{G}} = \boldsymbol{0}$, and satisfies

$$\frac{1}{\kappa}\mathcal{E}^{\mathcal{H}}(\boldsymbol{f}^{\mathcal{H}}) \leq \mathcal{E}^{\mathcal{G}}\left(\frac{1}{\kappa}\boldsymbol{f}^{\mathcal{G}}\right).$$

Observe that $\mathcal{H} \preceq_{\kappa} \mathcal{G}$ implies $\mathcal{H} \preceq_{\kappa}^{\text{cycle}} \mathcal{G}$.

These definitions satisfy most properties that we want from comparisons.

Lemma 2.9 (Reflexivity)**.** *For every smoothed p-norm instance $\mathcal{G}$, and every $\kappa \geq 1$, $\mathcal{G} \preceq_{\kappa} \mathcal{G}$ and $\mathcal{G} \preceq_{\kappa}^{cycle} \mathcal{G}$ with the identity map.*

It behaves well under composition.

Lemma 2.10 (Composition)**.** *Given two smoothed p-norm instances, $\mathcal{G}_1, \mathcal{G}_2$, such that $\mathcal{G}_1 \preceq_{\kappa_1} \mathcal{G}_2$ with the map $\mathcal{M}_{\mathcal{G}_1 \to \mathcal{G}_2}$ and $\mathcal{G}_2 \preceq_{\kappa_2} \mathcal{G}_3$ with the map $\mathcal{M}_{\mathcal{G}_2 \to \mathcal{G}_3}$, then $\mathcal{G}_1 \preceq_{\kappa_1\kappa_2} \mathcal{G}_3$ with the map $\mathcal{M}_{\mathcal{G}_1 \to \mathcal{G}_3} = \mathcal{M}_{\mathcal{G}_2 \to \mathcal{G}_3} \circ \mathcal{M}_{\mathcal{G}_1 \to \mathcal{G}_2}$.*

Similarly, for any $\mathcal{G}_1, \mathcal{G}_2$, if $\mathcal{G}_1 \preceq_{\kappa_1}^{cycle} \mathcal{G}_2$ with the map $\mathcal{M}_{\mathcal{G}_1 \to \mathcal{G}_2}$ and $\mathcal{G}_2 \preceq_{\kappa_2}^{cycle} \mathcal{G}_3$ with the map $\mathcal{M}_{\mathcal{G}_2 \to \mathcal{G}_3}$, then $\mathcal{G}_1 \preceq_{\kappa_1\kappa_2}^{cycle} \mathcal{G}_3$ with the map $\mathcal{M}_{\mathcal{G}_1 \to \mathcal{G}_3} = \mathcal{M}_{\mathcal{G}_2 \to \mathcal{G}_3} \circ \mathcal{M}_{\mathcal{G}_1 \to \mathcal{G}_2}$.

The most important property of this is that this notion of approximation is also additive, *i.e.*, it works well with graph decompositions.

Definition 2.11 (Union of two instances)**.** Consider smoothed p-norm instances, $\mathcal{G}_1, \mathcal{G}_2$, with the same set of vertices, *i.e.* $V^{\mathcal{G}_1} = V^{\mathcal{G}_2}$. Define $\mathcal{G} = \mathcal{G}_1 \cup \mathcal{G}_2$ as the instance on the same set of vertices obtained by taking a disjoint union of the edges (potentially resulting in multi-edges). Formally,

$$\mathcal{G} = (V^{\mathcal{G}_1}, E^{\mathcal{G}_1} \cup E^{\mathcal{G}_2}, (\boldsymbol{g}^{\mathcal{G}_1}, \boldsymbol{g}^{\mathcal{G}_2}), (\boldsymbol{r}^{\mathcal{G}_1}, \boldsymbol{r}^{\mathcal{G}_2}), (\boldsymbol{s}^{\mathcal{G}_1}, \boldsymbol{s}^{\mathcal{G}_2})).$$

Lemma 2.12 ($\preceq_\kappa$ under union)**.** *Consider four smoothed p-norm instances, $\mathcal{G}_1, \mathcal{G}_2, \mathcal{H}_1, \mathcal{H}_2$, on the same set of vertices,* i.e. $V^{\mathcal{G}_1} = V^{\mathcal{G}_2} = V^{\mathcal{H}_1} = V^{\mathcal{H}_2}$*, such that for $i = 1, 2$, $\mathcal{H}_i \preceq_\kappa \mathcal{G}_i$ with the map $\mathcal{M}_{\mathcal{H}_i \to \mathcal{G}_i}$. Let $\mathcal{G} \stackrel{\text{def}}{=} \mathcal{G}_1 \cup \mathcal{G}_2$, and $\mathcal{H} \stackrel{\text{def}}{=} \mathcal{H}_1 \cup \mathcal{H}_2$. Then, $\mathcal{H} \preceq_\kappa \mathcal{G}$ with the map*

$$\mathcal{M}_{\mathcal{H} \to \mathcal{G}}\left(\boldsymbol{f}^{\mathcal{H}} = (\boldsymbol{f}^{\mathcal{H}_1}, \boldsymbol{f}^{\mathcal{H}_2})\right) \stackrel{\text{def}}{=} \left(\mathcal{M}_{\mathcal{H}_1 \to \mathcal{G}_1}(\boldsymbol{f}^{\mathcal{H}_1}), \mathcal{M}_{\mathcal{H}_2 \to \mathcal{G}_2}(\boldsymbol{f}^{\mathcal{H}_2})\right),$$

where $(\boldsymbol{f}^{H_1}, \boldsymbol{f}^{H_2})$ is the decomposition of $\boldsymbol{f}^H$ onto the supports of H_1 and H_2.

This notion of approximation also behaves nicely with scaling of ℓ_2 and ℓ_p resistances.

Lemma 2.13. *For all $\kappa \geq 1$, and for all pairs of smoothed p-norm instances, $\mathcal{G}, \mathcal{H}$, on the same underlying graphs,* i.e., $(V^{\mathcal{G}}, E^{\mathcal{G}}) = (V^{\mathcal{H}}, E^{\mathcal{H}})$*, such that,*

1. *the gradients are identical, $\boldsymbol{g}^{\mathcal{G}} = \boldsymbol{g}^{\mathcal{H}}$,*
2. *the ℓ_2^2 resistances are off by at most κ,* i.e., $\boldsymbol{r}_e^{\mathcal{G}} \leq \kappa \boldsymbol{r}_e^{\mathcal{H}}$ *for all edges e, and*
3. *the p-norm scaling is off by at most κ^{p-1},* i.e., $s^{\mathcal{G}} \leq \kappa^{p-1} s^{\mathcal{H}}$,

then $\mathcal{H} \preceq_\kappa \mathcal{G}$ with the identity map.

2.4 Orthogonal Decompositions of Flows

At the core of our graph decomposition and sparsification procedures is a decomposition of the gradient $\boldsymbol{g}$ of $\mathcal{G}$ into its cycle space and potential flow space. We denote such a splitting using

$$\boldsymbol{g}^{\mathcal{G}} = \widehat{\boldsymbol{g}}^{\mathcal{G}} + \boldsymbol{B}^{\mathcal{G}} \boldsymbol{\psi}^{\mathcal{G}}, \text{ s.t. } \boldsymbol{B}^{\mathcal{G}^\top} \widehat{\boldsymbol{g}}^{\mathcal{G}} = \boldsymbol{0}. \quad (6)$$

Here $\widehat{\boldsymbol{g}}$ is a circulation, while $\boldsymbol{B}\boldsymbol{\psi}$ gives a potential induced edge value. We will omit the superscripts when the context is clear.

The following minimization based formulation of this splitting of $\boldsymbol{g}$ is critical to our method of bounding the overall progress of our algorithm

Fact 2.14. *The projection of $\boldsymbol{g}$ onto the cycle space is obtained by minimizing the energy added to a potential flow to $\boldsymbol{g}$. Specifically,*

$$\|\widehat{\boldsymbol{g}}\|_2^2 = \min_{\boldsymbol{x}} \|\boldsymbol{g} + \boldsymbol{B}\boldsymbol{x}\|_2^2.$$

Lemma 2.15. *Given a graph/gradient instance $\mathcal{G}$, consider $\mathcal{H}$ formed from a subset of its edges. The projections of $\boldsymbol{g}^{\mathcal{G}}$ and $\boldsymbol{g}^{\mathcal{H}}$ onto their respective cycle spaces, $\widehat{\boldsymbol{g}}^{\mathcal{G}}$ and $\widehat{\boldsymbol{g}}^{\mathcal{H}}$ satsify:*

$$\left\|\widehat{\boldsymbol{g}}^{\mathcal{H}}\right\|_2^2 \leq \left\|\widehat{\boldsymbol{g}}^{\mathcal{G}}\right\|_2^2 \leq \left\|\boldsymbol{g}^{\mathcal{G}}\right\|_2^2.$$

3 Numerical Methods

The general idea of (preconditioned) numerical methods, which are at the core of solvers for graph-structured linear systems [ST14] is to repeatedly update a current solution in ways that multiplicative reduce the difference in objective value to optimum. In the setting of flows, suppose we currently have some tentative solution $\boldsymbol{f}$ to the minimization problem

$$\min_{\boldsymbol{B}\boldsymbol{f}=\boldsymbol{b}} \|\boldsymbol{f}\|_p^p \tag{7}$$

by performing the step

$$\boldsymbol{f} \leftarrow \boldsymbol{f} + \boldsymbol{\delta},$$

with the goal of improving the objective value substantially.

The work of Adil *et al.* [AKPS19] proved that ℓ_p-norm minimization problems could be iteratively refined. While that result hinted at a much more general theory of numerical iterative methods for minimizing convex objectives, this topic is very much under development. In this section, we will develop the tools necessary for preconditioning ℓ_p-norm based functions, and formalize the requirements for preconditioners necessary for recursive preconditioning algorithms.

3.1 Iterative Refinement

The following key Lemma from [AKPS19] allows us to approximate the change in the smoothed p-norm of $\boldsymbol{x}+\boldsymbol{\delta}$ relative to the norm of $\boldsymbol{x}$, in terms of another smoothed p-norm of $\boldsymbol{\delta}$.

Lemma 3.1 ([AKPS19])**.** *For all $\boldsymbol{r}, \boldsymbol{x}, \boldsymbol{\delta} \in \mathbb{R}^m$, with $\boldsymbol{r} \in \mathbb{R}^m_{\geq 0}$, and $s \geq 0$, we have*

$$2^{-p} \cdot h_p(\boldsymbol{r} + |\boldsymbol{x}|^{p-2}, s, \boldsymbol{\delta}) \leq h_p(\boldsymbol{r}, s, \boldsymbol{x}+\boldsymbol{\delta}) - h_p(\boldsymbol{r}, s, \boldsymbol{x}) - \boldsymbol{\delta}^\top \nabla_{\boldsymbol{x}} h_p(\boldsymbol{r}, s, \boldsymbol{x}) \leq 2^{2p} \cdot h_p(\boldsymbol{r} + |\boldsymbol{x}|^{p-2}, s, \boldsymbol{\delta}).$$

The above lemma gives us the following theorem about iteratively refining smoothed ℓ_p-norm minimization problems. While the lemma was essentially proven in [AKPS19], they used slightly different definitions, and for completeness we prove the lemma in Appendix B.

The following theorem also essentially appeared in [AKPS19], but again as slightly different definitions were used in that paper, we prove the theorem in Appendix B for completeness.

Theorem 3.2 ([AKPS19])**.** *Given the following optimization problem,*

$$\begin{array}{rl} \max_{\boldsymbol{x}} & \mathcal{E}_1(\boldsymbol{x}) \stackrel{\text{def}}{=} \boldsymbol{g}^\top \boldsymbol{x} - h_p(\boldsymbol{r}, s, \boldsymbol{x}) \\ s.t. & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \end{array} \tag{P1}$$

and an initial feasible solution $\boldsymbol{x}_0$, we can construct the following residual problem:

$$\begin{array}{rl} \max_{\boldsymbol{\delta}} & \mathcal{E}_2(\boldsymbol{\delta}) \stackrel{\text{def}}{=} (\boldsymbol{g}')^\top \boldsymbol{\delta} - h_p(\boldsymbol{r}', s, \boldsymbol{\delta}) \\ s.t. & \boldsymbol{A}\boldsymbol{\delta} = \boldsymbol{0}, \end{array} \tag{R1}$$

where $\boldsymbol{g}' = 2^p(\boldsymbol{g} - \nabla_{\boldsymbol{x}} h(\boldsymbol{r}, s, \boldsymbol{x})|_{\boldsymbol{x}=\boldsymbol{x}_0})$, *and* $\boldsymbol{r}' = \boldsymbol{r} + s|\boldsymbol{x}_0|^{p-2}$.

There exists a feasible solution $\widetilde{\boldsymbol{\delta}}$ *to the residual problem R1 that achieves an objective of* $\mathcal{E}_2\left(\widetilde{\boldsymbol{\delta}}\right) \geq 2^p(\mathcal{E}_1(\boldsymbol{x}^\star) - \mathcal{E}_1(\boldsymbol{x}_0))$, *where* $\boldsymbol{x}^\star$ *is an optimal solution to problem P1.*

Moreover, given any feasible solution $\boldsymbol{\delta}$ *to Program R1, the vector* $\boldsymbol{x}_1 \stackrel{\text{def}}{=} \boldsymbol{x}_0 + 2^{-3p}\boldsymbol{\delta}$ *is a feasible solution to the Program P1 and obtains the objective*

$$\mathcal{E}_1(\boldsymbol{x}_1) \geq \mathcal{E}_1(\boldsymbol{x}_0) + 2^{-4p}\mathcal{E}_2(\boldsymbol{\delta}).$$

Importantly, we can apply the above theorem to smoothed p-norm flow/circulation problems.

Corollary 3.3 (Iterative refinement for smoothed p-norm flow/circulation problems)**.** *Given any smoothed p-norm flow problem* $(\mathcal{G}, \boldsymbol{b})$ *with optimal objective* $\mathcal{E}^\star(\mathcal{G})$, *and any initial circulation* $\boldsymbol{f}_0$, *we can build, in* $O(|E^{\mathcal{G}}|)$ *time, a smoothed p-norm circulation problem* $\mathcal{H}$ *with the same underlying graph* $(V^{\mathcal{H}}, E^{\mathcal{H}}) = (V^{\mathcal{G}}, E^{\mathcal{G}})$, *such that* $\mathcal{E}^\star(\mathcal{H}) \geq 2^p(\mathcal{E}^\star(\mathcal{G}) - \mathcal{E}^{\mathcal{G}}(\boldsymbol{f}_0))$ *and for any circulation* $\boldsymbol{f}^{\mathcal{H}}$ *on* $\mathcal{H}$, *the flow* $\boldsymbol{f}_1 \stackrel{\text{def}}{=} \boldsymbol{f}_0 + 2^{-3p}\boldsymbol{f}^{\mathcal{H}}$ *satisfies residues* $\boldsymbol{b}$ *on* $\mathcal{G}$ *and has an objective*

$$\mathcal{E}^{\mathcal{G}}(\boldsymbol{f}_1) \geq \mathcal{E}^{\mathcal{G}}(\boldsymbol{f}_0) + 2^{-4p}\mathcal{E}^{\mathcal{H}}(\boldsymbol{f}^{\mathcal{H}}).$$

This means if we find even a crude approximate minimizer $\widetilde{\boldsymbol{\delta}}$ of this update problem, we can move to a new point $\boldsymbol{f}' = \boldsymbol{f} + \widetilde{\boldsymbol{\delta}}$, so that the gap to the optimum in the original optimization problem (7) will decrease by a constant factor (depending only on p) from $\|\boldsymbol{f}\|_p^p - \mathrm{OPT}$ to $\|\boldsymbol{f}'\|_p^p - \mathrm{OPT} \leq (1 - 2^{-O(p)})(\|\boldsymbol{f}\|_p^p - \mathrm{OPT})$. In other words, we have a kind of iterative refinement: crude solutions to an update problem directly give constant factor progress in the original objective.

Note that $\|\boldsymbol{f}\|_p^p = \sum_i \boldsymbol{f}_i^p$. This will help us understand the objective function of the update problem coordinate-wise. Our update problem objective function is motivated by the following observations. Our function differs slightly from the function used in [AKPS19], which in turn was based on functions from [BCLL18], but our function still uses a few special properties of the [BCLL18] functions. Suppose $p \geq 2$ is an even integer (only to avoid writing absolute values), then

$$\boldsymbol{f}_i^p + p\boldsymbol{f}_i^{p-1}\boldsymbol{\delta}_i + 2^{-O(p)} \underbrace{(\boldsymbol{f}_i^{p-2}\boldsymbol{\delta}_i^2 + \boldsymbol{\delta}_i^p)}_{\text{write as } h_p(\boldsymbol{f}_i^{p-2}, \boldsymbol{\delta}_i)} \leq (\boldsymbol{f}_i + \boldsymbol{\delta}_i)^p \leq \boldsymbol{f}_i^p + p\boldsymbol{f}_i^{p-1}\boldsymbol{\delta}_i + 2^{O(p)} \underbrace{(\boldsymbol{f}_i^{p-2}\boldsymbol{\delta}_i^2 + \boldsymbol{\delta}_i^p)}_{h_p(\boldsymbol{f}_i^{p-2}, \boldsymbol{\delta}_i)}$$

Of course, the exact expansion gives

$$(\boldsymbol{f}_i + \boldsymbol{\delta}_i)^p = \boldsymbol{f}_i^p + p\boldsymbol{f}_i^{p-1}\boldsymbol{\delta}_i + \frac{p(p-1)}{2}\boldsymbol{f}_i^{p-2}\boldsymbol{\delta}_i^2 + \frac{p(p-1)(p-2)}{6}\boldsymbol{f}_i^{p-3}\boldsymbol{\delta}_i^3 + \ldots + \boldsymbol{\delta}_i^p \tag{8}$$

So essentially we can approximate this expansion using only the zeroth, first, second, and *last* term in the expansion. We use $\boldsymbol{g}(\boldsymbol{f})$ to denote the vector with $\boldsymbol{g}_i(\boldsymbol{f}) = p\boldsymbol{f}_i^{p-1}$ (i.e. the gradient), and let $\boldsymbol{f}^{p-2}$ denote the entrywise powered vector, and define $h_p(\boldsymbol{f}^{p-2}, \boldsymbol{\delta}) = \sum_i h_p(\boldsymbol{f}_i^{p-2}, \boldsymbol{\delta}_i)$. Thus we have

$$\|\boldsymbol{f}\|_p^p + \boldsymbol{g}(\boldsymbol{f})^\top\boldsymbol{\delta} + 2^{-O(p)}h_p(\boldsymbol{f}^{p-2}, \boldsymbol{\delta}) \leq \|\boldsymbol{f} + \boldsymbol{\delta}\|_p^p \leq \|\boldsymbol{f}\|_p^p + \boldsymbol{g}(\boldsymbol{f})^\top\boldsymbol{\delta} + 2^{O(p)}h_p(\boldsymbol{f}^{p-2}, \boldsymbol{\delta})$$

Note that for any scalar $0 < \lambda < 1$,

$$\lambda^p h_p(\boldsymbol{f}^{p-2}, \boldsymbol{\delta}) \leq h_p(\boldsymbol{f}^{p-2}, \lambda\boldsymbol{\delta}) \leq \lambda^2 h_p(\boldsymbol{f}^{p-2}, \boldsymbol{\delta})$$

Together, these observations are enough to ensure that if we have $\widetilde{\boldsymbol{\delta}}$ which is a constant factor approximate solution to the follow optimization problem, which we define as our *update problem*

$$\min_{\boldsymbol{B\delta}=\mathbf{0}} \boldsymbol{g}(\boldsymbol{f})^{\top}\boldsymbol{\delta} + h_p(\boldsymbol{f}^{p-2}, \boldsymbol{\delta}) \tag{9}$$

then we can find a λ s.t. $\left\|\boldsymbol{f} + \lambda\widetilde{\boldsymbol{\delta}}\right\|_p^p - \mathrm{OPT} \leq (1 - 2^{-O(p)})(\|\boldsymbol{f}\|_p^p - \mathrm{OPT})$.

But what have we gained? Why is Problem (9) more tractable than the one we started with?

A key reason is that unlike the exact expansion of an update as given by Equation (8), all the higher order terms in the objective function of (9) are coordinate-wise even functions, i.e. flipping the sign of a coordinate does not change the value of the function. [AKPS19] used a different but still even function instead of our h_p. This symmetrization allowed them to develop a multiplicative weights update algorithm motivated by [CKM+11] for their version of Problem (9), reducing the problem to solving a sequence of linear equations.

For our choice of h_p, it is particularly simple to show another very important property: Consider solving Problem (9) by *again* applying iterative refinement to this problem. That is, at some intermediate step with δ being the current solution, we aim to find an update $\widehat{\boldsymbol{\delta}}$ s.t. $\boldsymbol{B}\widehat{\boldsymbol{\delta}} = \mathbf{0}$ and $\boldsymbol{g}(\boldsymbol{f})^{\top}(\boldsymbol{\delta} + \widehat{\boldsymbol{\delta}}) + h_p(\boldsymbol{f}^{p-2}, \boldsymbol{\delta} + \widehat{\boldsymbol{\delta}})$ is smaller than $\boldsymbol{g}(\boldsymbol{f})^{\top}(\boldsymbol{\delta}) + h_p(\boldsymbol{f}^{p-2}, \boldsymbol{\delta})$. Then by expanding the two non-linear terms of $h_p(\boldsymbol{f}_i^{p-2}, \boldsymbol{\delta}_i)$, i.e. $(\boldsymbol{\delta}_i + \widehat{\boldsymbol{\delta}})^p$ *and* $(\boldsymbol{\delta}_i + \widehat{\boldsymbol{\delta}})^2$, similar to Equation (8), we get a sequence of terms with powers of $\boldsymbol{\delta}_i$ ranging from 2 to p. If we approximate this sequence again using only the $\widehat{\boldsymbol{\delta}}_i^2$ and $\widehat{\boldsymbol{\delta}}_i^p$ terms, we get another update problem. This update problem is an instance of Problem (2). And in general, we can set up iterative refinement update problems for instances of Problem (2), and get back another problem of the that class (after our approximation based on dropping intermediate terms). Thus, the problem class (2) is closed under repeatedly creating iterative update problems. This observation is central because it allows us to develop recursive algorithms.

3.2 Vertex Elimination

Following the template of the Spielman-Teng Laplacian solver, we recursively solve a problem on m edges by solving about κ problems on graphs with $n-1+m/\kappa$ edges. These ultra-sparse graphs allow us to eliminate degree 1 and 2 vertices and obtain a smaller problem. Because our update problem (Problem (9)) corresponds to a flow-circulation problem with some objective, we are able to understand elimination on these objectives in a relatively simple way: the flow on degree 1 and 2 vertices is easily related to flow in a smaller graph created by elimination. Unlike Spielman-Teng, every recursive call must rely on a new graph sparsifier, because the "graph" we sparsify depends heavily on the current solution that we are seeking to update: we have to simultaneously preserve linear, quadratic and p-th order terms, whose weights depend on the current solution.

A critical component of this schema is the mapping of flows back and forth between the original graph and the new graph so a good solution on a smaller graph can be transformed into a good solution on the larger graph. These mappings are direct analogs of eliminating degrees 1 and 2 vertices. In Appendix C, we generalize these processes to smoothed ℓ_p-norm objectives, proving the following statements:

Theorem 3.4 (Eliminating vertices with degree 1 and 2)**.** *Given a smoothed p-norm instance $\mathcal{G}$, the algorithm* Eliminate($\mathcal{G}$) *returns another smoothed p-norm instance $\mathcal{G}'$, along with the map $\mathcal{M}_{\mathcal{G}'\to\mathcal{G}}$ in $O(\left|V^{\mathcal{G}}\right| + \left|E^{\mathcal{G}}\right|)$ time, such that the graph $G' = (V^{\mathcal{G}'}, E^{\mathcal{G}'})$ is obtained from the graph*

$G = (V^{\mathcal{G}}, E^{\mathcal{G}})$ *by first repeatedly removing vertices with non-selfloop degree*[2] *1 in G, and then replacing every path $u \rightsquigarrow v$ in G where all internal path vertices have non-selfloop degree exactly 2 in G, with a new edge (u, v).*

Moreover,

$$\mathcal{G}' \preceq^{cycle}_{\frac{1}{n^{p-1}}} \mathcal{G} \preceq^{cycle}_{1} \mathcal{G}',$$

where $n = \left|V^{\mathcal{G}}\right|$, and the map $\mathcal{M}_{\mathcal{G}' \to \mathcal{G}}$ can be applied in $O(\left|V^{\mathcal{G}}\right| + \left|E^{\mathcal{G}}\right|)$ time.

Lemma 3.5 (Eliminating Self-loops)**.** *There is an algorithm* RemoveLoops *such that, given a smoothed p-norm instance $\mathcal{G}$ with self-loops in $E^{\mathcal{G}}$, in $O(\left|V^{\mathcal{G}}\right| + \left|E^{\mathcal{G}}\right|)$ time, it returns instances $\mathcal{G}_1, \mathcal{G}_2$, such that $\mathcal{G} = \mathcal{G}_1 \cup \mathcal{G}_2$, where $\mathcal{G}_1$ is obtained from $\mathcal{G}$ by eliminating all self-loops from $E^{\mathcal{G}}$, and $\mathcal{G}_2$ is an instance consisting of just the self-loops from $\mathcal{G}$. Thus, any flow $\boldsymbol{f}^{\mathcal{G}_2}$ on $\mathcal{G}_2$ is a circulation.*

Moreover, there is an algorithm SolveLoops *that, given $\mathcal{G}_2$, for any $\delta \leq 1/p$, in time $O(|E^{\mathcal{G}_2}| \log 1/\delta)$, finds a circulation $\widetilde{\boldsymbol{f}}^{\mathcal{G}_2}$ on $\mathcal{G}_2$ such that*

$$\mathcal{E}^{\mathcal{G}_2}(\widetilde{\boldsymbol{f}}^{\mathcal{G}_2}) \geq (1 - \delta) \max_{\boldsymbol{f}^{\mathcal{G}} : (\boldsymbol{B})^{\mathcal{G}} \boldsymbol{f}^{\mathcal{G}} = \boldsymbol{0}} \mathcal{E}^{\mathcal{G}_2}(\boldsymbol{f}^{\mathcal{G}_2}).$$

We remark that only the map from the smaller graph to the larger has to be constructive.

3.3 Recursive Preconditioning

We can now present our main recursive preconditioning algorithm, RecursivePreconditioning (Algorithm 1). Earlier work on preconditioning for non-linear (maximum) flow problems all relied on *oblivious routing* which gives rise to linear preconditioners. These inherently cannot work well for high-accuracy iterative refinement, and the issue is not merely linearity: Consider Problem (9): the optimal $\boldsymbol{\delta}$ is highly dependent on the current $\boldsymbol{f}$, and when a coordinate $\boldsymbol{\delta}_i$ is large compared to the current $|\boldsymbol{f}_i|$, the function depends on it as $\boldsymbol{\delta}_i^p$, while when $\boldsymbol{\delta}_i$ is small compared to $|\boldsymbol{f}_i|$, it behaves as $\boldsymbol{\delta}_i^2$. Thus the behavior is highly dependent on the current solution. This necessitates adaptive and non-linear preconditioners.

To develop adaptive preconditioners, we employ recursive chains of alternating calls to non-linear iterative refinement and a new type of (ultra-)sparsification that is more general and stronger, allowing us to simultanously preserve multiple different properties of our problem. And crucially, every time our solution is updated, our preconditioners change. The central theorem governing the combinatorial components of our algorithm, which is the main result proven in Section 4, is:

Theorem 3.6 (Ultra-Sparsification)**.** *Given any instance $\mathcal{G} = (V^{\mathcal{G}}, E^{\mathcal{G}}, \boldsymbol{g}^{\mathcal{G}}, \boldsymbol{r}^{\mathcal{G}}, s^{\mathcal{G}})$ with n nodes, m edges, and parameters κ, δ where $\log \frac{1}{\delta}$ and $\log\left\|\boldsymbol{r}^{\mathcal{G}}\right\|_\infty$ are both $O(\log^c n)$ for some constant c and $\kappa < m$,* UltraSparsify *computes in $\widetilde{O}(m)$ running time another instance $\mathcal{H} = (V^{\mathcal{H}}, E^{\mathcal{H}}, \boldsymbol{g}^{\mathcal{H}}, \boldsymbol{r}^{\mathcal{H}}, s^{\mathcal{H}} = s^{\mathcal{G}})$ along with flow mapping functions $\mathcal{M}_{\mathcal{H} \to \mathcal{G}}, \mathcal{M}_{\mathcal{G} \to \mathcal{H}}$ such that $V^{\mathcal{H}} = V^{\mathcal{G}}$, and with high probability we have*

1. *E^H consists of a spanning tree in the graph $(V^{\mathcal{G}}, E^{\mathcal{G}})$, up to $m - n + 1$ self-loops and at most $\widetilde{O}(\frac{m}{\kappa})$ other non self-loop edges.*

[2] By non-selfloop degree, we mean that self-loops do not count towards the degree of a vertex.

2. With $\kappa_{\mathcal{G}\to\mathcal{H}} = \widetilde{O}(\kappa m^{3/(p-1)})$ *for any flow* $\boldsymbol{f}^{\mathcal{G}}$ *of* $\mathcal{G}$ *we have*

$$\mathcal{E}_{\mathcal{H}}\left(\frac{\mathcal{M}_{\mathcal{G}\to\mathcal{H}}(\boldsymbol{f}^{\mathcal{G}})}{\kappa_{\mathcal{G}\to\mathcal{H}}}\right) \geq \frac{1}{\kappa_{\mathcal{G}\to\mathcal{H}}}\mathcal{E}_{\mathcal{G}}(\boldsymbol{f}^{\mathcal{G}}) - \delta\left\|\boldsymbol{f}^{\mathcal{G}}\right\|_2\left\|\boldsymbol{g}^{\mathcal{G}}\right\|_2,$$

and with $\kappa_{\mathcal{H}\to\mathcal{G}} = \widetilde{O}(m^{2/(p-1)})$*, for any flow solution* $\boldsymbol{f}^{\mathcal{H}}$ *of* $\mathcal{H}$ *we have*

$$\mathcal{E}_{\mathcal{G}}\left(\frac{\mathcal{M}_{\mathcal{H}\to\mathcal{G}}(\boldsymbol{f}^{\mathcal{H}})}{\kappa_{\mathcal{H}\to\mathcal{G}}}\right) \geq \frac{1}{\kappa_{\mathcal{H}\to\mathcal{G}}}\mathcal{E}_{\mathcal{H}}(\boldsymbol{f}^{\mathcal{H}}) - \delta\left(\left\|\boldsymbol{f}^{\mathcal{H}}\right\|_2\left\|\boldsymbol{g}^{\mathcal{G}}\right\|_2 + \left\|\boldsymbol{f}^{\mathcal{H}}\right\|_2^2\right).$$

The flow mappings $\mathcal{M}_{\mathcal{H}\to\mathcal{G}}, \mathcal{M}_{\mathcal{G}\to\mathcal{H}}$ *preserve residue of flow, and can be applied in* $\widetilde{O}(m)$ *time.*

Algorithm 1 Recursive Preconditioning Algorithm for p-smoothed flow/circulation problem

procedure RecursivePreconditioning($\mathcal{G}, \boldsymbol{b}, \boldsymbol{f}^{(0)}, \kappa, \delta$)
  $m \leftarrow \left|E^{\mathcal{G}}\right|$. If $m \leq \widetilde{O}(\kappa)$, solve $\mathcal{G}$ using the algorithm from [AKPS19]
  $T \leftarrow \widetilde{O}(2^{3p}\kappa m^{\frac{6}{p-1}})$
  for $t = 0$ to T **do**
    Construct the residual smoothed p-norm circulation problem $\mathcal{H}_1$ for $(\mathcal{G}, \boldsymbol{b})$ with the current solution $\boldsymbol{f}^{(t)}$, given by Corollary 3.3.
    $\delta' \leftarrow \min\{1, \left\|\boldsymbol{g}^{\mathcal{H}_1}\right\|^{-\frac{p}{p-1}}\} \cdot \delta/(4Tm)$.
    $\mathcal{H}_2, \mathcal{M}_{\mathcal{H}_2\to\mathcal{H}_1}, \kappa_{\mathcal{H}_2\to\mathcal{H}_1} \leftarrow$ UltraSparsify($\mathcal{H}_1, \kappa, \delta'$) ▷ $\mathcal{H}_2$ is an ultrasparsifier for $\mathcal{H}_1$
    $\mathcal{H}_3, \mathcal{M}_{\mathcal{H}_3\to\mathcal{H}_2} \leftarrow$ Eliminate($\mathcal{H}_2$) ▷ Gaussian elimination to remove degree 1,2 vertices
    $\mathcal{H}_4, \mathcal{H}_{\text{loops}} \leftarrow$ RemoveLoops($\mathcal{H}_3$) ▷ Remove self-loops
    $\boldsymbol{\Delta}^{\mathcal{H}_{\text{loops}}} \leftarrow$ SolveLoops($\mathcal{H}_{\text{loops}}, 1/p$) ▷ Solve the self-loop instance
    $\boldsymbol{\Delta}^{\mathcal{H}_4} \leftarrow$ RecursivePreconditioning($\mathcal{H}_4, \mathbf{0}, \mathbf{0}, \kappa, \delta/T$) ▷ Recurse on smaller instance
    $\boldsymbol{\Delta}^{\mathcal{H}_3} \leftarrow \boldsymbol{\Delta}^{\mathcal{H}_4} + \boldsymbol{\Delta}^{\mathcal{H}_{\text{loops}}}$ ▷ Adding solution for $\mathcal{H}^{\text{loops}}$ to obtain solution for $\mathcal{H}_3$
    $\boldsymbol{\Delta}^{\mathcal{H}_2} \leftarrow \left|V^{\mathcal{H}_2}\right|^{-\frac{1}{p-1}} \cdot \mathcal{M}_{\mathcal{H}_3\to\mathcal{H}_2}(\boldsymbol{\Delta}^{\mathcal{H}_3})$. ▷ Undo elimination to map solution back to $\mathcal{H}_2$
    $\boldsymbol{\Delta}^{\mathcal{H}_1} \leftarrow \kappa^{-1}_{\mathcal{H}_2\to\mathcal{H}_1}\mathcal{M}_{\mathcal{H}_2\to\mathcal{H}_1}(\boldsymbol{\Delta}^{\mathcal{H}_2})$ ▷ Map it back to the residual problem
    $\boldsymbol{f}^{(t+1)} \leftarrow \boldsymbol{f}^{(t)} + 2^{-3p}\boldsymbol{\Delta}^{\mathcal{H}_1}$ ▷ Update the current flow solution
  return $\boldsymbol{f}^{(T)}$

Our key theorem about the performance of the algorithm is then:

Theorem 3.7 (Recursive Preconditioning)**.** *For all* $p \geq 2$*, say we are given a smoothed* p*-norm instance* $\mathcal{G}$*, residues* $\boldsymbol{b}$*, initial solution* $\boldsymbol{f}^{(0)}$*, and* $\delta \leq 1$ *such that* $\log 1/\delta, \log\left\|\boldsymbol{g}^{\mathcal{G}}\right\|, \log\left\|\boldsymbol{r}^{\mathcal{G}}\right\|, \log\left\|\boldsymbol{f}^{(0)}\right\| \leq \widetilde{O}(1)$*. We can pick* $\kappa = \widetilde{\Theta}(m^{\frac{1}{\sqrt{p-1}}})$ *so that the procedure* RecursivePreconditioning($\mathcal{G}, \boldsymbol{b}, \boldsymbol{f}^{(0)}, \kappa, \delta$) *runs in time* $2^{O(p^{3/2})}m^{1+O(\frac{1}{\sqrt{p-1}})}$*, and returns a flow* $\boldsymbol{f}$ *on* $\mathcal{G}$ *such that* $\boldsymbol{f}$ *satisfies residues* $\boldsymbol{b}$*, and*

$$\mathcal{E}^{\star}(\mathcal{G}) - \mathcal{E}^{\mathcal{G}}(\boldsymbol{f}^{(T)}) \leq \frac{1}{2}(\mathcal{E}^{\star}(\mathcal{G}) - \mathcal{E}^{\mathcal{G}}(\boldsymbol{f}^{(0)})) + \delta s^{\mathcal{G}}.$$

Proof. (of Theorem 3.7) By scaling $\boldsymbol{g}^{\mathcal{G}}, \boldsymbol{r}^{\mathcal{G}}$, we can assume that $s^{\mathcal{G}} = 1$ without loss of generality.

Let us consider iteration t of the for loop in RecursivePreconditioning. First, let us prove guarantees on the optimal solutions of all the relevant instances. By the guarantees of Corollary 3.3, we know that $\mathcal{H}_1$ is a smoothed p-norm circulation problem with the same underlying graph $(V^{\mathcal{H}_1}, E^{\mathcal{H}_1}) = (V^{\mathcal{G}}, E^{\mathcal{G}})$, such that $\mathcal{E}^{\star}(\mathcal{H}_1) \geq 2^p(\mathcal{E}^{\star}(\mathcal{G}) - \mathcal{E}^{\mathcal{G}}(\boldsymbol{f}^{(t)}))$.

From Theorem 3.6, we know that UltraSparsify returns a smoothed p-norm circulation instance $\mathcal{H}_2$ on the same set of vertices such that $\mathcal{E}^\star(\mathcal{H}_2) \geq \kappa^{-1}_{\mathcal{H}_1\to\mathcal{H}_2}\mathcal{E}^\star(\mathcal{H}_1) - \delta'\left\|\boldsymbol{g}^{\mathcal{H}_1}\right\|\left\|\boldsymbol{f}^{\star\mathcal{H}_1}\right\|$.

From Theorem 3.4, we know that the instance $\mathcal{H}_3$ returned by Eliminate($\mathcal{H}_2$) satisfies $\mathcal{H}_2 \preceq_1^{\text{cycle}} \mathcal{H}_3$, and hence $\mathcal{E}^\star(\mathcal{H}_2) \leq \mathcal{E}^\star(\mathcal{H}_3)$. From Lemma 3.5, we know that $\mathcal{E}^\star(\mathcal{H}_4) + \mathcal{E}^\star(\mathcal{H}_{\text{loops}}) = \mathcal{E}^\star(\mathcal{H}_3)$. Combining these guarantees, we obtain,

$$\mathcal{E}^\star(\mathcal{H}_3) = \mathcal{E}^\star(\mathcal{H}_4) + \mathcal{E}^\star(\mathcal{H}_{\text{loops}}) \geq 2^p\kappa^{-1}_{\mathcal{H}_1\to\mathcal{H}_2}(\mathcal{E}^\star(\mathcal{G}) - \mathcal{E}^{\mathcal{G}}(\boldsymbol{f}^{(t)})) - \delta\left\|\boldsymbol{g}^{\mathcal{H}_1}\right\|\left\|\boldsymbol{f}^{\star\mathcal{H}_1}\right\|.$$

Now, we analyze the approximation guarantee provided by the solutions to these instances. From Lemma 3.5, SolveLoops($\mathcal{H}_{\text{loops}}$) returns a $\boldsymbol{\Delta}^{\mathcal{H}_{\text{loops}}}$ that satisfies $\mathcal{E}^{\mathcal{H}_{\text{loops}}}(\boldsymbol{\Delta}^{\mathcal{H}_{\text{loops}}}) \geq \frac{1}{2}\alpha^\star(\mathcal{H}_{\text{loops}})$. By induction, RecursivePreconditioning($\mathcal{H}_4, \mathbf{0}, \kappa, \delta T^{-1}$), upon starting with the initial solution $\mathbf{0}$, returns a $\boldsymbol{\Delta}^{\mathcal{H}_4}$ that satisfies, $\mathcal{E}^{\mathcal{H}_4}(\boldsymbol{\Delta}^{\mathcal{H}_4}) \geq \frac{1}{2}\mathcal{E}^\star(\mathcal{H}_4) - \delta T^{-1}$. Combining these guarantees, we have,

$$\mathcal{E}^{\mathcal{H}_3}(\boldsymbol{\Delta}^{\mathcal{H}_3}) = \mathcal{E}^{\mathcal{H}_4}(\boldsymbol{\Delta}^{\mathcal{H}_4}) + \mathcal{E}^{\mathcal{H}_{\text{loops}}}(\boldsymbol{\Delta}^{\mathcal{H}_{\text{loops}}}) \geq \frac{1}{2}\mathcal{E}^\star(\mathcal{H}_3) - \delta T^{-1}.$$

From Theorem 3.4, we also have $\mathcal{H}_3 \preceq^{\text{cycle}}_{\kappa_{\text{elim}}} \mathcal{H}_2$, for $\kappa_{\text{elim}} = \left|V^{\mathcal{H}_2}\right|^{\frac{1}{p-1}}$, and hence

$$\kappa^{-1}_{\text{elim}}\mathcal{E}^{\mathcal{H}_3}(\boldsymbol{\Delta}^{\mathcal{H}_3}) \leq \mathcal{E}^{\mathcal{H}_2}(\kappa^{-1}_{\text{elim}}\mathcal{M}_{\mathcal{H}_3\to\mathcal{H}_2}(\boldsymbol{\Delta}^{\mathcal{H}_3})) = \mathcal{E}^{\mathcal{H}_2}(\boldsymbol{\Delta}^{\mathcal{H}_2}).$$

Finally, from Theorem 3.6, we have,

$$\mathcal{E}^{\mathcal{H}_1}(\boldsymbol{\Delta}^{\mathcal{H}_1}) = \mathcal{E}^{\mathcal{H}_1}(\kappa^{-1}_{\mathcal{H}_2\to\mathcal{H}_1}\mathcal{M}_{\mathcal{H}_2\to\mathcal{H}_1}(\boldsymbol{\Delta}^{\mathcal{H}_2})) \geq \kappa^{-1}_{\mathcal{H}_2\to\mathcal{H}_1}\mathcal{E}^{\mathcal{H}_2}(\boldsymbol{\Delta}^{\mathcal{H}_2}) - \delta'\left\|\boldsymbol{g}^{\mathcal{H}_1}\right\|\left\|\boldsymbol{\Delta}^{\mathcal{H}_2}\right\| - \delta'\left\|\boldsymbol{\Delta}^{\mathcal{H}_2}\right\|^2$$

Combining these guarantees, we obtain,

$$\begin{aligned}
\mathcal{E}^{\mathcal{H}_1}(\boldsymbol{\Delta}^{\mathcal{H}_1}) &\geq \kappa^{-1}_{\mathcal{H}_2\to\mathcal{H}_1}\kappa^{-1}_{\text{elim}}\left(\frac{1}{2}2^p\kappa^{-1}_{\mathcal{H}_1\to\mathcal{H}_2}(\mathcal{E}^\star(\mathcal{G}) - \mathcal{E}^{\mathcal{G}}(\boldsymbol{f}^{(t)})) - \frac{1}{2}\delta'\left\|\boldsymbol{g}^{\mathcal{H}_1}\right\|\left\|\boldsymbol{f}^{\star\mathcal{H}_1}\right\| - \delta T^{-1}\right) \\
&\qquad - \delta'\left\|\boldsymbol{g}^{\mathcal{H}_1}\right\|\left\|\boldsymbol{\Delta}^{\mathcal{H}_2}\right\| - \delta'\left\|\boldsymbol{\Delta}^{\mathcal{H}_2}\right\|^2 \\
&\geq \widetilde{\Omega}(2^p\kappa^{-1}m^{-\frac{6}{p-1}})(\mathcal{E}^\star(\mathcal{G}) - \mathcal{E}^{\mathcal{G}}(\boldsymbol{f}^{(t)})) \\
&\qquad - \delta'\left\|\boldsymbol{g}^{\mathcal{H}_1}\right\|\left\|\boldsymbol{f}^{\star\mathcal{H}_1}\right\| - \delta'\left\|\boldsymbol{g}^{\mathcal{H}_1}\right\|\left\|\boldsymbol{\Delta}^{\mathcal{H}_2}\right\| - \delta'\left\|\boldsymbol{\Delta}^{\mathcal{H}_2}\right\|^2 - \delta T^{-1}. \\
&\geq \widetilde{\Omega}(2^p\kappa^{-1}m^{-\frac{6}{p-1}})(\mathcal{E}^\star(\mathcal{G}) - \mathcal{E}^{\mathcal{G}}(\boldsymbol{f}^{(t)})) - 2\delta T^{-1},
\end{aligned}$$

$\left\|\boldsymbol{f}^{\star\mathcal{H}_1}\right\| \leq \sqrt{m}\left\|\boldsymbol{g}^{\mathcal{H}_1}\right\|^{\frac{1}{p-1}}$, since $\mathcal{E}^{\mathcal{H}_1}(\boldsymbol{f}^{\star\mathcal{H}_1}) \geq 0$ implies that $m^{1-p/2}\left\|\boldsymbol{f}^{\star\mathcal{H}_1}\right\|_2^p \leq \left\|\boldsymbol{f}^{\star\mathcal{H}_1}\right\|_p^p \leq \boldsymbol{g}^{\mathcal{H}_1\top}\boldsymbol{f}^{\star\mathcal{H}_1} \leq \left\|\boldsymbol{g}^{\mathcal{H}_1}\right\|\left\|\boldsymbol{f}^{\star\mathcal{H}_1}\right\|$. Similarly, $\left\|\boldsymbol{\Delta}^{\mathcal{H}_1}\right\| \leq \sqrt{m}\left\|\boldsymbol{g}^{\mathcal{H}_1}\right\|^{\frac{1}{p-1}}$, and $\left\|\boldsymbol{\Delta}^{\mathcal{H}_2}\right\| \leq \sqrt{m}\left\|\boldsymbol{\Delta}^{\mathcal{H}_1}\right\|$, by the reverse tree routing map.

Thus, by Theorem 3.3, $\boldsymbol{f}^{(t+1)}$ satisfies

$$\begin{aligned}
\mathcal{E}^\star(\mathcal{G}) - \mathcal{E}^{\mathcal{G}}(\boldsymbol{f}^{(t+1)}) &\leq \mathcal{E}^\star(\mathcal{G}) - \mathcal{E}^{\mathcal{G}}(\boldsymbol{f}^{(t)}) - 2^{-4p}\mathcal{E}^{\mathcal{H}_1}(\boldsymbol{\Delta}^{\mathcal{H}_1}) \\
&\leq \left(1 - \widetilde{\Omega}(2^{-3p}\kappa^{-1}m^{-\frac{6}{p-1}})\right)(\mathcal{E}^\star(\mathcal{G}) - \mathcal{E}^{\mathcal{G}}(\boldsymbol{f}^{(t)})) + 2^{-4p}\cdot 2\delta T^{-1}.
\end{aligned}$$

Thus, repeating for loop $\widetilde{O}(2^{3p}\kappa m^{\frac{6}{p-1}})$ times gives us a solution $\boldsymbol{f}^{(T)}$ such that

$$\mathcal{E}^\star(\mathcal{G}) - \mathcal{E}^{\mathcal{G}}(\boldsymbol{f}^{(T)}) \leq \frac{1}{2}(\mathcal{E}^\star(\mathcal{G}) - \mathcal{E}^{\mathcal{G}}(\boldsymbol{f}^{(0)})) + \delta.$$

Now, we analyze the running time. In a single iteration, the total cost of all the operations other than the recursive call to RecursivePreconditioning is $\widetilde{O}(m)$. Note that $\mathcal{H}_1$ has m edges. Theorem 3.6 tells us that $\mathcal{H}_2$ consists of a tree ($n-1$ edges), at most $\widetilde{O}(m/\kappa)$ non-selfloop edges, plus many self-loops. After invoking Eliminate, and RemoveLoops, the instance $\mathcal{H}_4$ has no self-loops left, and after dropping vertices with degree 0, only has vertices with degree at least 3. Every edge removed in Eliminate decreases the number of edges and vertices with non-zero non-selfloop degree by 1. Suppose n' is the number of vertices in $\mathcal{H}_4$ with non-zero degree. Then, Eliminate must have removed $n - n'$ edges from $\mathcal{H}_2$. Since $\mathcal{H}_4$ must have at least $\frac{3}{2}n'$ vertices, we have $n - 1 + \widetilde{O}(m/\kappa) - (n - n') \geq \frac{3}{2}n'$. Hence, $n' \leq \widetilde{O}(m/\kappa)$. Thus $\mathcal{H}_4$ is an instance with at most $\widetilde{O}(m/\kappa)$ vertices and edges.

Thus, the total running time recurrence is

$$T(m) \leq \widetilde{O}(2^{3p}\kappa m^{\frac{6}{p-1}})\Big(T(m/\kappa) + \widetilde{O}(m)\Big).$$

Note that κ is fixed throughout the recursion. By picking $\kappa = \widetilde{\Theta}(m^{\frac{1}{\sqrt{p-1}}})$, we can fix the depth of the recursion to be $O(\sqrt{p-1})$. The total cost is dominated by the cost at the bottom level of the recursion, which adds up to a total running time of $2^{O(p^{3/2})}m^{1+O(\frac{1}{\sqrt{p-1}})}$.

The above discussion does not take into account the reduction in δ as we go down the recursion. Observe that δ is lower bounded by

$$\delta(mT\max\|\boldsymbol{g}\|^{\frac{p}{p-1}})^{-O(\sqrt{p-1})} = \delta(m2^p \max\{\{\|\boldsymbol{f}_0\|, \left\|\boldsymbol{g}^{\mathcal{G}}\right\|, \left\|\boldsymbol{r}^{\mathcal{G}}\right\|\}^{\frac{p}{p-1}}\})^{-O(\sqrt{p-1})}.$$

Thus, we always satisfy $\log\delta = \widetilde{O}(1)$. ☐

We can now prove the central collaries regardling smoothed ℓ_p-norm flows and ℓ_p-norm flows.

Theorem 1.1 (Smoothed ℓ_p-norm flows)**.** *For any $p \geq 2$, given weights $\boldsymbol{r} \in \mathbb{R}^E_{\geq 0}$, a "gradient" $\boldsymbol{g} \in \mathbb{R}^E$, a demand vector $\boldsymbol{b} \in \mathbb{R}^V$ (with $\boldsymbol{b}^\top \boldsymbol{1} = 0$), and an initial solution $\boldsymbol{f}^{(0)}$ such that all parameters are bounded by $2^{poly(\log n)}$, we can compute a flow $\widetilde{\boldsymbol{f}}$ satisfying demands $\boldsymbol{b}$,* i.e., *$\boldsymbol{B}^{G\top}\widetilde{\boldsymbol{f}} = \boldsymbol{b}$, such that*

$$val(\widetilde{\boldsymbol{f}}) - OPT \leq \frac{1}{poly(m)}\Big(val(\boldsymbol{f}^{(0)}) - OPT\Big) + \frac{1}{poly(m)}$$

in $2^{O(p^{3/2})}m^{1+O(\frac{1}{\sqrt{p}})}$ time, where m denotes the number of edges in G.

Proof. First note that the Problem (2) is a smoothed p-norm instance after flipping the sign of $\boldsymbol{g}$, and the sign of the objective function. We can solve this smoothed p-norm instance $\mathcal{G}$ by using Theorem 3.7 to compute the desired approximate solution to the residual problems. We start with $\boldsymbol{f}^{(0)}$ as our initial solution. At iteration t, we invoke Theorem 3.7 using $\boldsymbol{f}^{(t-1)}$ as the initial solution,

$$\boldsymbol{f}(t) \leftarrow \textsc{RecursivePreconditioning}\left(\mathcal{G}, \boldsymbol{b}, \boldsymbol{f}^{(t-1)}, \kappa, \frac{1}{\text{poly}(n)}\right),$$

where κ is given by Theorem 3.7.

We know that $\boldsymbol{f}^{(t)}$ satisfies,

$$\mathcal{E}^\star(\mathcal{G}) - \mathcal{E}^{\mathcal{G}}(\boldsymbol{f}^{(t)}) \leq \frac{1}{2}(\mathcal{E}^\star(\mathcal{G}) - \mathcal{E}^{\mathcal{G}}(\boldsymbol{f}^{(t-1)})) + \frac{1}{\text{poly}(n)}.$$

Iterating $O(\log n)$ times, we obtain,

$$\mathcal{E}^\star(\mathcal{G}) - \mathcal{E}^{\mathcal{G}}(\boldsymbol{f}^{(T)}) \leq \frac{1}{\text{poly}(n)}(\mathcal{E}^\star(\mathcal{G}) - \mathcal{E}^{\mathcal{G}}(\boldsymbol{f}^{(0)})) + \frac{1}{\text{poly}(n)}.$$

Finally, noting that we had flipped the sign of the objective function in Problem (2), we obtain our claim. ☐

Theorem 1.2 (ℓ_p-norm flows)**.** *For any $p \geq 2$, given an unweighted graph $G(V,E)$ and demands $\boldsymbol{b}$, using the routine* pFlows$(\mathcal{G}, \boldsymbol{b})$ *(Algorithm 2) we can compute a flow $\widetilde{\boldsymbol{f}}$ satisfying satisfying $\boldsymbol{b}$,* i.e., *$\boldsymbol{B}^{G\top}\widetilde{\boldsymbol{f}} = \boldsymbol{b}$, such that*

$$\left\|\widetilde{\boldsymbol{f}}\right\|_p^p \leq \left(1 + \frac{1}{poly(m)}\right) \min_{\boldsymbol{f}:\boldsymbol{B}^{G\top}\boldsymbol{f}=\boldsymbol{b}} \|\boldsymbol{f}\|_p^p.$$

in $2^{O(p^{3/2})}m^{1+O(\frac{1}{\sqrt{p}})}$, time, where m denotes the number of edges in G.

Proof. The pseudocode for our procedure pFlows$(\mathcal{G}, \boldsymbol{b})$ for this problem is given in Algorithm 2. Our goal is to compute a flow $\widetilde{\boldsymbol{f}}$ satisfying $\boldsymbol{B}^{G\top}\widetilde{\boldsymbol{f}} = \boldsymbol{b}$, such that

$$\left\|\widetilde{\boldsymbol{f}}\right\|_p^p \leq \left(1 + \frac{1}{\text{poly}(m)}\right)\|\boldsymbol{f}^\star\|_p^p,$$

where $\boldsymbol{f}^\star$ is the flow minimizing the ℓ_p-norm with residue $\boldsymbol{b}$. For concreteness, we take this to mean $\left\|\widetilde{\boldsymbol{f}}\right\|_p^p \leq (1 + 3m^{-c})\|\boldsymbol{f}^\star\|_p^p$, for some constant c. We construct a smoothed p-norm instance $\mathcal{G} = (V, E, \mathbf{0}, \mathbf{0}, 1)$. Note that the smoothed p-norm flow problem $(\mathcal{G}, \boldsymbol{b})$ finds a flow satisfying residues $\boldsymbol{b}$, and maximizing $\mathcal{E}^{\mathcal{G}}(\boldsymbol{f}) = -\|\boldsymbol{f}\|_p^p$. We can solve this smoothed p-norm instance by iteratively refining using Corollary 3.3, and using Theorem 3.7 to compute the desired approximate solution to the residual problems.

Formally, we use Laplacian solvers to compute in $\widetilde{O}(m)$ time $\boldsymbol{f}^{(0)}$ as a 2-approximation to $\min_{\boldsymbol{f}:\boldsymbol{B}^\top\boldsymbol{f}=\boldsymbol{b}}\|\boldsymbol{f}\|$. We have,

$$\left\|\boldsymbol{f}^{(0)}\right\|_p \leq \left\|\boldsymbol{f}^{(0)}\right\|_2 \leq 2 \min_{\boldsymbol{B}^{G\top}\boldsymbol{f}=\boldsymbol{b}} \|\boldsymbol{f}\|_2 \leq 2\|\boldsymbol{f}^\star\|_2 \leq 2m^{\frac{1}{2}-\frac{1}{p}}\|\boldsymbol{f}^\star\|_p.$$

At each iteration t, we construct the residual smoothed p-norm circulation problem $\mathcal{H}_t$ for $(\mathcal{G}, \boldsymbol{b})$ with the current solution $\boldsymbol{f}^{(t)}$, given by Corollary 3.3. We then invoke

$$\boldsymbol{\Delta}^{(t)} \leftarrow \textsc{RecursivePreconditioning}(\mathcal{H}_t, \mathbf{0}, \mathbf{0}, \kappa, \frac{1}{2^p m^{p/2} m^c} \cdot \left\|\boldsymbol{f}^{(0)}\right\|^p),$$

where $\kappa = \widetilde{\Theta}(m^{\frac{1}{\sqrt{p-1}}})$ is given by Theorem 3.7. Let $\boldsymbol{\Delta}^{(t)}$ be the flow returned. We know

$$\mathcal{E}^\star(\mathcal{H}_t) - \mathcal{E}^{\mathcal{H}_t}(\boldsymbol{\Delta}^{(t)}) \leq \frac{1}{2}\mathcal{E}^\star(\mathcal{H}_t) + \frac{1}{2^p m^{p/2} m^c}\left\|\boldsymbol{f}^{(0)}\right\|_p^p$$

We let $\boldsymbol{f}^{(t+1)} \leftarrow \boldsymbol{f}^{(t)} + 2^{-3p}\boldsymbol{\Delta}^{(t)}$. Thus, by Corollary 3.3, at every iteration, we have

$$
\begin{aligned}
-\left\|\boldsymbol{f}^{(t+1)}\right\|_p^p = \mathcal{E}^{\mathcal{G}}(\boldsymbol{f}^{(t+1)}) &\geq \mathcal{E}^{\mathcal{G}}(\boldsymbol{f}^{(t)}) + 2^{-4p}\mathcal{E}^{\mathcal{H}_t}(\boldsymbol{\Delta}^{(t)}) \\
&\geq -\left\|\boldsymbol{f}^{(t)}\right\|_p^p + 2^{-4p}\left(\frac{1}{2}\mathcal{E}^{\star}(\mathcal{H}_t) - \frac{1}{2^p m^{p/2} m^c}\left\|\boldsymbol{f}^{(0)}\right\|_p^p\right) \\
&\geq -\left\|\boldsymbol{f}^{(t)}\right\|_p^p + 2^{-4p}\left(\frac{1}{2}2^p(\mathcal{E}^{\star}(\mathcal{G}) - \mathcal{E}^{\mathcal{G}}(\boldsymbol{f}^{(t)})) - \frac{1}{m^c}\|\boldsymbol{f}^{\star}\|_p^p\right) \\
&\geq -\left\|\boldsymbol{f}^{(t)}\right\|_p^p + 2^{-4p}\left(\left\|\boldsymbol{f}^{(t)}\right\|_p^p - \|\boldsymbol{f}^{\star}\|_p^p\right) - \frac{2^{-4p}}{m^c}\|\boldsymbol{f}^{\star}\|_p^p.
\end{aligned}
$$

where we have used, $\left\|\boldsymbol{f}^{(0)}\right\|_p \leq 2m^{\frac{1}{2}-\frac{1}{p}}\|\boldsymbol{f}^{\star}\|_p$,

$$
\left\|\boldsymbol{f}^{(t+1)}\right\|_p^p - \|\boldsymbol{f}^{\star}\|_p^p \leq (1-2^{-4p})\left(\left\|\boldsymbol{f}^{(t)}\right\|_p^p - \|\boldsymbol{f}^{\star}\|_p^p\right) + \frac{2^{-4p}}{m^c}\|\boldsymbol{f}^{\star}\|_p^p.
$$

Thus

$$
\begin{aligned}
\left\|\boldsymbol{f}^{(t)}\right\|_p^p - \|\boldsymbol{f}^{\star}\|_p^p &\leq (1-2^{-4p})\left(\left\|\boldsymbol{f}^{(t)}\right\|_p^p - \|\boldsymbol{f}^{\star}\|_p^p\right) + \frac{2^{-4p}}{m^c}\|\boldsymbol{f}^{\star}\|_p^p \\
&\leq \max\left((1-2^{-(4p+1)})\left(\left\|\boldsymbol{f}^{(t)}\right\|_p^p - \|\boldsymbol{f}^{\star}\|_p^p\right), 3m^{-c}\|\boldsymbol{f}^{\star}\|_p^p\right).
\end{aligned}
$$

Where first step follows by rearranging terms. To establish the second inequality, first consider the case when $\left(\left\|\boldsymbol{f}^{(t)}\right\|_p^p - \|\boldsymbol{f}^{\star}\|_p^p\right) \geq 2m^{-c}\|\boldsymbol{f}^{\star}\|_p^p$ and hence $\left\|\boldsymbol{f}^{(t)}\right\|_p^p - \|\boldsymbol{f}^{\star}\|_p^p \leq (1-2^{-4p}/2)\left(\left\|\boldsymbol{f}^{(t)}\right\|_p^p - \|\boldsymbol{f}^{\star}\|_p^p\right)$, meanwhile, when $\left(\left\|\boldsymbol{f}^{(t)}\right\|_p^p - \|\boldsymbol{f}^{\star}\|_p^p\right) < 2m^{-c}\|\boldsymbol{f}^{\star}\|_p^p$, the inequality is immediate.

Iterating $T = \Theta((c+p)2^{4p}\log m)$ times gives us

$$
\left\|\boldsymbol{f}^{(T)}\right\|_p^p - \|\boldsymbol{f}^{\star}\|_p^p \leq \max\left((1-2^{-(4p+1)})^T\left(\left\|\boldsymbol{f}^{(0)}\right\|_p^p - \|\boldsymbol{f}^{\star}\|_p^p\right), 3m^{-c}\|\boldsymbol{f}^{\star}\|_p^p\right) \leq 3m^{-c}\|\boldsymbol{f}^{\star}\|_p^p.
$$

□

4 Graph Theoretic Preconditioners

In this section, we discuss at a high level of the construction of ultra-sparsifiers for a smooth ℓ_p-norm instance. We start by restating the main theorem of our ultra-sparsifier. After establishing the necessary tools, we prove this theorem at the end of this section.

Theorem 3.6 (Ultra-Sparsification)**.** *Given any instance $\mathcal{G} = (V^{\mathcal{G}}, E^{\mathcal{G}}, \boldsymbol{g}^{\mathcal{G}}, \boldsymbol{r}^{\mathcal{G}}, s^{\mathcal{G}})$ with n nodes, m edges, and parameters κ, δ where $\log\frac{1}{\delta}$ and $\log\left\|\boldsymbol{r}^{\mathcal{G}}\right\|_\infty$ are both $O(\log^c n)$ for some constant c and $\kappa < m$,* UltraSparsify *computes in $\widetilde{O}(m)$ running time another instance $\mathcal{H} = (V^{\mathcal{H}}, E^{\mathcal{H}}, \boldsymbol{g}^{\mathcal{H}}, \boldsymbol{r}^{\mathcal{H}}, s^{\mathcal{H}} = s^{\mathcal{G}})$ along with flow mapping functions $\mathcal{M}_{\mathcal{H}\to\mathcal{G}}, \mathcal{M}_{\mathcal{G}\to\mathcal{H}}$ such that $V^{\mathcal{H}} = V^{\mathcal{G}}$, and with high probability we have*

Algorithm 2 Computing p-norm minimizing flows. Given constant p and c, the routine computes $\widetilde{\boldsymbol{f}}$ with residues $\boldsymbol{b}$ and p-norm that is within a factor $(1+3m^{-c})$ of the minimum p-norm achievable for these residues.

procedure PFLOWS($\mathcal{G}, \boldsymbol{b}$)
  Use Laplacian solvers to compute $\boldsymbol{f}^{(0)}$ as a 2-approximation to $\min_{\boldsymbol{f}:\boldsymbol{B}^\top \boldsymbol{f}=\boldsymbol{b}} \|\boldsymbol{f}\|$.
  $\delta \leftarrow \min\left\{1, \frac{1}{2^p m^{p/2} m^c} \cdot \left\|\boldsymbol{f}^{(0)}\right\|_p^p\right\}$
  $\kappa \leftarrow \widetilde{\Theta}(m^{\frac{1}{\sqrt{p-1}}})$
  $T \leftarrow \Theta((c+p)2^{4p} \log m)$
  for $t = 0$ to $T-1$ **do**
    Construct the residual smoothed p-norm circulation problem $\mathcal{H}_t$ for $(\mathcal{G}, \boldsymbol{b})$ with the current solution $\boldsymbol{f}^{(t)}$, given by Corollary 3.3.
    $\boldsymbol{\Delta}^{(t)} \leftarrow$ RECURSIVEPRECONDITIONING($\mathcal{H}_t, \boldsymbol{0}, \boldsymbol{0}, \kappa, \delta$)
    $\boldsymbol{f}^{(t+1)} \leftarrow \boldsymbol{f}^{(t)} + 2^{-3p} \boldsymbol{\Delta}^{(t)}$
  return $\widetilde{\boldsymbol{f}} \leftarrow \boldsymbol{f}^{(T)}$

1. *E^H consists of a spanning tree in the graph $(V^{\mathcal{G}}, E^{\mathcal{G}})$, up to $m-n+1$ self-loops and at most $\widetilde{O}(\frac{m}{\kappa})$ other non self-loop edges.*

2. *With $\kappa_{\mathcal{G}\to\mathcal{H}} = \widetilde{O}(\kappa m^{3/(p-1)})$ for any flow $\boldsymbol{f}^{\mathcal{G}}$ of $\mathcal{G}$ we have*

$$\mathcal{E}_{\mathcal{H}}\left(\frac{\mathcal{M}_{\mathcal{G}\to\mathcal{H}}(\boldsymbol{f}^{\mathcal{G}})}{\kappa_{\mathcal{G}\to\mathcal{H}}}\right) \geq \frac{1}{\kappa_{\mathcal{G}\to\mathcal{H}}} \mathcal{E}_{\mathcal{G}}(\boldsymbol{f}^{\mathcal{G}}) - \delta \left\|\boldsymbol{f}^{\mathcal{G}}\right\|_2 \left\|\boldsymbol{g}^{\mathcal{G}}\right\|_2,$$

and with $\kappa_{\mathcal{H}\to\mathcal{G}} = \widetilde{O}(m^{2/(p-1)})$, for any flow solution $\boldsymbol{f}^{\mathcal{H}}$ of $\mathcal{H}$ we have

$$\mathcal{E}_{\mathcal{G}}\left(\frac{\mathcal{M}_{\mathcal{H}\to\mathcal{G}}(\boldsymbol{f}^{\mathcal{H}})}{\kappa_{\mathcal{H}\to\mathcal{G}}}\right) \geq \frac{1}{\kappa_{\mathcal{H}\to\mathcal{G}}} \mathcal{E}_{\mathcal{H}}(\boldsymbol{f}^{\mathcal{H}}) - \delta\left(\left\|\boldsymbol{f}^{\mathcal{H}}\right\|_2 \left\|\boldsymbol{g}^{\mathcal{G}}\right\|_2 + \left\|\boldsymbol{f}^{\mathcal{H}}\right\|_2^2\right).$$

The flow mappings $\mathcal{M}_{\mathcal{H}\to\mathcal{G}}, \mathcal{M}_{\mathcal{G}\to\mathcal{H}}$ preserve residue of flow, and can be applied in $\widetilde{O}(m)$ time.

Our high-level approach is the same as Spielman-Teng [ST14], where we utilize a low-stretch spanning tree, and move off-tree edges to a small set of portal nodes. Once most off-tree edges are only between a small set of portal nodes, we sparsify the graph over the portal nodes to reduce the number of edges. As we need to map flow solutions between the original instance and the sparsified instance, our main concern is to carry out these step without incurring too much error on the objective function values. In our case we have ℓ_p^p resistances and gradients on edges in addition to ℓ_2^2 resistances, and thus the main challenge is to simultaneously preserve their respective terms.

4.1 Tree-Portal Routing

Faced with a sparse graph or dense graph, we wish to move most edges onto a few vertices of a tree, so that many of the remaining vertices are low degree and can be eliminated. Our high-level approach is the same as Spielman-Teng [ST14], where we utilize a low-stretch spanning tree, and move off-tree edges to a small set of portal nodes. However, when we reroute flow on the graph where most edges are moved to be among a few vertices using the tree, we need to (approximately) preserve three different properties of the flow on the original graph, namely the inner product between gradients and flows $\sum_e \boldsymbol{g}_e \boldsymbol{f}_e$, the 2-norm energy $\boldsymbol{r}_e \boldsymbol{f}_e^2$, and the ℓ_p-norm energy $\sum_e \boldsymbol{f}_e^p$. It turns out that we can move edges around on our graphs to produce a new graph while exactly preserving the linear term $\sum_e \boldsymbol{g}_e \boldsymbol{f}_e$ for flows mapped between one graph and the other. This means any tree is acceptable from the point of preserving the linear term. To move edges around and bound distortion of solutions w.r.t. the quadratic $\boldsymbol{r}_e \boldsymbol{f}_e^2$ term, we use a low stretch tree w.r.t. the $\boldsymbol{r}$ weights as resistances. This leaves us with little flexibility for the $\sum_e \boldsymbol{f}_e^p$ term. However, for large p, provided *every* p-th order term is weighted the same (i.e. we have $s \sum_e \boldsymbol{f}_e^p$ instead of $\sum_e \boldsymbol{s}_e \boldsymbol{f}_e^p$), it turns out that, moving edges along any tree will result in bounded distortion of the solution, provided we are careful about how we move those edges. Thus, we can move edges around carefully to be among a small subset of the portal nodes while simultaneously controlling all linear, 2-nd order and p-th order terms. But, this only works if all the p-th order terms are weighted the same. This leads us to maintain uniform-weighted p-th order terms as an invariant throughout the algorithm. The iterative refinement steps naturally weigh all p-th order terms the same provided the original function does. However, our sparsification procedures do not immediately achieve this, but we show we can enforce this uniform-weight invariant with only a manageable additional distortion of our solutions. Elimination also does not naturally weigh all p-th order terms the same even in our case when the original function does, but we can bound the distortion incurred by explicitly making the weights uniform. Our tree-based edge re-routing naturally creates maps between solutions on the old and new graphs.

We first formalize what we mean by moving off-tree edges. Suppose we have a spanning tree T of a graph (V, E) and a subset set of nodes $\widehat{V} \subset V$ designated as *portal nodes*, for any off-tree edge $e = \{u, v\} \in E \setminus T$, there is a unique tree path $\mathcal{P}_T(u, v)$ in T from u to v. We define a *tree-portal path* $\mathcal{P}_{T,\widehat{V}}(u, v)$, which is not necessarily a simple path.

Definition 4.1 (Tree-portal path and edge moving)**.** Given spanning tree T and set of portal nodes $\widehat{V}$, let $e = \{u, v\}$ be any edge not in T, and $P_T(u, v)$ the unique tree path in T from u to v. We define e's *tree-portal path* $P_{T,\widehat{V}}(u, v)$ and e's image under *tree-portal edge moving* as follows

1. If $P_T(u, v)$ doesn't go through any portal vertex. In this case, we replace $\{u, v\}$ with a distinct self-loop of v. We let $P_{T,\widehat{V}}(u, v)$ be the path $P_T(u, v)$ followed by the self-loop at v.

2. If $P_T(u, v)$ goes through exactly one portal vertex $\widehat{v}$. In this case, we replace $\{u, v\}$ with a distinct self-loop at $\widehat{v}$. We let $P_{T,\widehat{V}}(u, v)$ be the tree path $P_T(u, \widehat{v})$ followed by the self-loop at $\widehat{v}$ and then the tree path $P_T(\widehat{v}, v)$.

3. If P_{uv} goes through at least two portal vertices. In this case, let $\widehat{u}$ (and $\widehat{v}$) be closest the portal vertex to u (and v) on P_{uv}, we replace $\{u, v\}$ with a distinct edge[3] $\{\widehat{u}, \widehat{v}\}$. We let $P_{T,\widehat{V}}(u, v)$ be the tree path $P_T(u, \widehat{u})$ followed by the new edge from $\widehat{u}$ to $\widehat{v}$ and then the tree path $P_T(\widehat{v}, v)$.

This maps any off-tree edge e to a unique (edge or self-loop) $\widehat{e}$ given any $T, \widehat{V}$. We denote the tree-portal edge moving with the map $\widehat{e} = \textsc{Move}_{T,\widehat{V}}(e)$.

Although we will get self-loops in tree-portal routing, to keep the discussion simple, we ignore the possibility of getting self-loops. This still captures all the main ideas, and the algorithm/analysis

[3]We will keep multi-edges explicitly between portal nodes.

extends to self-loops in a very straightforward but slightly tedious way. We discuss self-loops briefly at the end of the section.

Tree-portal routing is a mapping from flow solutions on the original off-tree edges to a flow solution (with the same residue) using the edges they are mapped to. Any flow along off-tree edge (u, v) in the original graph is rerouted (again from u to v) using the tree-portal path $P_{T,\widehat{V}}(u, v)$ instead. Rerouting the flow of any off-tree edge along its tree-portal path increases the congestion on tree edges, which in turn incurs error in the ℓ_2^2 and ℓ_p^p terms in the objective function. We need to pick the tree and portal nodes carefully to bound the error.

Definition 4.2. Given any graph (V, E), resistance $\boldsymbol{r}$ on edges, a spanning tree T, and a set of portals $\widehat{V} \subset V$, for any $e = \{u, v\} \in E$, let $\widehat{e} = \textsc{Move}_{T,\widehat{V}}(e)$ and $P_{T,\widehat{V}}(u, v)$ be as specified above. The *stretch* of $e = \{u, v\} \in E \setminus T$ in the tree-portal routing is

$$\mathsf{Str}_{T,\widehat{V}}(e) \stackrel{\text{def}}{=} \frac{1}{\boldsymbol{r}_e} \sum_{e' \in P_{T,\widehat{V}}(e) \setminus \{\widehat{e}\}} \boldsymbol{r}_{e'},$$

and the stretch of a tree edge $e \in T$ is $\mathsf{Str}_{T,\widehat{V}}(e) = 1$. Note with our definition $\mathsf{Str}_{T,\emptyset}(e)$ gives the standard stretch.

The starting point is low stretch spanning trees [AN12], which provide good bounds on the total ℓ_2^2 stretch.

Lemma 4.3 (Low-Stretch Trees [AN12])**.** *Given any graph $G = (V, E)$ of m edges and n nodes, as well as resistance $\boldsymbol{r}$, $\mathrm{LSST}(\boldsymbol{r})$ finds a spanning tree in $O(m \log n \log \log n)$ time such that*

$$\sum_{e \in E} \mathsf{Str}_{T,\emptyset}(e) \leq O(m \log n \log \log n).$$

We will construct a low-stretch spanning tree T of $(V^{\mathcal{G}}, E^{\mathcal{G}}, \boldsymbol{r}^{\mathcal{G}})$ using the above result. Still, the error will be too large if we only use tree edges to reroute the flow of all the off-tree edges, since the low average stretch doesn't prevent one tree edge to be on the tree path for many off-tree edges. Thus, we need to add portal nodes so we can shortcut between them to reduce the extra congestion on tree edges.

4.2 Partitioning Trees into Subtrees and Portals

Next, we show how to find a small set of good portal nodes so that rerouting flow on off-tree edges using their tree-portal paths incurs small error in the objective function. Pseudocode of this routine is in Algorithm 3, and its guarantees are stated in Lemma 4.4 below.

Lemma 4.4. *There is a linear-time routine* $\textsc{FindPortals}$ *that given any graph G, a spanning tree T, with $\widehat{m}$ off-tree edges, and a portal count $\widehat{n} \leq \widehat{m}$, returns a subset of $\widehat{V}$ of $\widehat{n}$ vertices so that for all edges $\widehat{e} \in T$, we have*

$$\sum_{e : \widehat{e} \in P_{T,\widehat{V}}(e)} \mathsf{Str}_{T,\widehat{V}}(e) \leq \frac{10}{\widehat{n}} \sum_{e} \mathsf{Str}_{T,\emptyset}(e)$$

$$\left| e : \widehat{e} \in P_{T,\widehat{V}}(e) \right| \leq \frac{10\widehat{m}}{\widehat{n}}.$$

Algorithm 3 Find portal nodes for tree-portal routing

procedure FINDPORTAL(T,E,$\widehat{n}$)

$\forall e \in E : \eta(e) \leftarrow \max\left(\mathsf{Str}_{T,\emptyset}(e), \frac{\sum_{e' \in E} \mathsf{Str}_{T,\emptyset}(e')}{|E|}\right)$

Call *decompose* in [ST14] (page 881 of journal version) with $(T, E, \eta, \widehat{n})$.

The subroutine breaks T into at most $\widehat{n}$ edge-disjoint induced tree pieces to divide up the $\eta(e)$'s roughly evenly so that the sum of $\eta(e)$ for all e attached to each non-singleton piece is not too big.

The subroutine works by recursively cut off sub-trees from T whenever the sum of $\eta(e)$ of all e attached to a sub-tree is above $\frac{2\sum_e \eta(e)}{\widehat{n}}$.

Let $\widehat{V}$ be the set of nodes where the tree pieces intersect.

This lemma will be a fairly straightforward using the tree decomposition subroutine (page 881 of journal version) from Spielman and Teng [ST14], which we include below for completeness.

Definition 4.5 ([ST14] Definition 10.2)**.** Given a tree T that spans a set of vertices V, a T-decomposition is a decomposition of V into sets $W_1, \ldots, W_h$ such that $V = \bigcup W_i$, the graph induced by T on each W_i is a tree, possibly with just one vertex, and for all $i \neq j$, $|W_i \bigcap W_j| \leq 1$.

Given an additional set of edges E on V, a (T, E)-decomposition is a pair $(\{W_1, \ldots, W_h\}, \rho)$ where $\{W_1, \ldots, W_h\}$ is a T-decomposition and ρ is a map that sends each edge of E to a set or pair of sets in $\{W_1, \ldots, W_h\}$ so that for each edge in $(u, v) \in E$,

1. if $\rho(u, v) = \{W_i\}$, then $\{u, v\} \subset W_i$, and
2. if $\rho(u, v) = \{W_i, W_j\}$, then either $u \in W_i$ and $v \in W_j$ or vice versa.

Theorem 4.6 ([ST14] Theorem 10.3)**.** *There exists a linear-time algorithm such that on input a set of edges E on vertex set V, a spanning tree T on V, a function $\eta : E \to \mathbb{R}^+$, and an integer $1 < t \leq \sum_{e \in E} \eta(e)$, outputs a (T, E)-decomposition $(\{W_1, \ldots, W_h\}, \rho)$, such that*

1. $h \leq t$

2. for all W_i such that $|W_i| > 1$,

$$\sum_{e \in E : W_i \in \rho(e)} \eta(e) \leq \frac{4}{t} \sum_{e \in E} \eta(e)$$

We can use the above theorem to show Lemma 4.4.

Proof of Lemma 4.4. We will apply Theorem 4.6 with $t = \widehat{n}$, and the function η will be

$$\eta(e) = \max\left(\mathsf{Str}_{T,\emptyset}(e), \frac{\sum_{e' \in E} \mathsf{Str}_{T,\emptyset}(e')}{\widehat{m}}\right)$$

Note by construction $\sum_e \eta(e) \leq 2 \sum_{e' \in E} \mathsf{Str}_{T,\emptyset}(e')$. We get $\{W_1, \ldots, W_{\widehat{n}}\}$ back, and let T_i be the tree induced by T on W_i. Note the T_i's will be edge disjoint, and cover all tree edges of T. Our set of portals will be the set of nodes that are in more than one of the W_i's, i.e. the nodes where different T_i's overlap. Note the number of portals is at most the number of T_i's by an inductive argument from any T_i that is a sub-tree in T. Such T_i have exactly one portal, and we can remove T_i from T and continue the argument until all that remain in T is one sub-tree.

Consider any tree edge $\widehat{e} \in T$, suppose it in T_i for some i. $\widehat{e}$ can only be on the tree-portal routing for some edge $\{u, v\}$ when $W_i \in \rho(u, v)$. Note as T_i contains at least one tree edge, we know $|W_i| > 1$, the second guarantee in Theorem 4.6 gives

$$\sum_{e:W_i \in \rho(e)} \max\left(\mathsf{Str}_{T,\emptyset}(e), \frac{\sum_{e' \in E} \mathsf{Str}_{T,\emptyset}(e')}{\widehat{m}}\right) \leq \frac{4}{t}\left(2 \sum_{e' \in E} \mathsf{Str}_{T,\emptyset}(e')\right)$$

which directly gives the bounds we want in the lemma. □

4.3 Graph Sparsification

Once we are able to move most of the edges onto a small subset of vertices, we wish to sparsify the resulting dense graph over those vertices. This sparsification has to simultaneously preserve properties of 1-st, 2-nd and p-th order terms, as well as the interactions between them, which turns out to be challenging. We resort to expander decomposition which allows us to partition the vertex set s.t. the edges internal to each subset form an expander and not too many edges cross the partitions. Just having an expander graph is not enough to allow us to sample the graph due to the need of preserving the linear terms. Thus, we also require that on each expander the orthogonal projection of the gradient to the cycle space of the sub-graph has its maximum squared entry not much larger than the the average squared entry. We refer to this as a uniform (projected) gradient. We discuss how to obtain an expander decomposition that guarantees the projected gradients are uniform in the expanders later in this overview. Given the uniform projected gradient condition, we show that we can uniformly sample edges of these expanders to create sparsified versions of them. We construct maps between the flows on an original expander and its sampled version that work for *any* flow, not only a circulation. These maps preserve the linear term $\sum_e \boldsymbol{g}_e \boldsymbol{f}_e$ exactly, while bounding the cost of the 2-norm and ℓ_p-norm terms by relating them to the cost of the optimal routing of a flow with the same demands and same gradient inner product, and showing that optimal solutions are similar on the original expander and its sampled version. This strategy resembles the flow maps developed in [KLOS14], and like their maps, we route demands using electrical flows on individual expanders, but additionally we need create a flow in the cycle space that depends on projection of the gradient onto the cycle space.

Tree-portal routing will give us an instance where all the off-tree edges are between portal nodes. We can look at the sub-graph restricted to the portal nodes and the off-tree edges between them. This graph has many fewer nodes comparing to the original graph but roughly the same number of edges, and thus is much denser. We can then sparsify this graph to reduce the number of off-tree edges similar to the construction of spectral sparsifiers. The main technical difficulty is that in the sparsified graph, we still want the ℓ_p^p terms in our objective function to have a same scalar s for every edge, but similar to the case of how resistances are scaled in spectral sparsification, to preserve the total value of the ℓ_p^p terms, we would naturally want to scale s according to the probability we sample an edge e. Thus, to get a same scalar s for all sampled edges, we are limited to uniform sampling. We know uniform sampling works in expanders (c.f. [ST14, SS11] and [KLOS14, She13] for ℓ_2 and ℓ_∞ respectively), so the natural approach is to first decompose the graph into expanders, and sampling uniformly inside each expander. However, because of the presence of a gradient, we need to be a bit more careful than even expanderdecomposition-based sparsification steps. Thus, we work with *uniform expanders*.

Definition 4.7. A graph [4] G is a α-uniform ϕ-expander (or *uniform expander* when parameters not spelled out explicitly) if

1. $\boldsymbol{r}$ on all edges are the same.
2. $\boldsymbol{s}$ on all edges are the same.
3. G has *conductance*[5] at least ϕ.
4. The projection of $\boldsymbol{g}$ onto the cycle space of G, $\widehat{\boldsymbol{g}}^G = (I - \boldsymbol{B}\boldsymbol{L}^\dagger\boldsymbol{B}^\top)\boldsymbol{g}$, is α-*uniform* (see next definition), where $\boldsymbol{B}$ is the edge-vertex incidence matrix of G, and $\boldsymbol{L} = \boldsymbol{B}^\top\boldsymbol{B}$ is the Laplacian.

Definition 4.8. A vector $\boldsymbol{y} \in \mathbb{R}^m$ is said to be α-*uniform* if

$$\|\boldsymbol{y}\|_\infty^2 \leq \frac{\alpha}{m}\|\boldsymbol{y}\|_2^2.$$

We abuse the notation to also let the all zero vector $\mathbf{0}$ be 1-uniform.

In Section 5 we show how to decompose the graph consisting of portals and the off-tree edges between them into vertex disjoint uniform expanders such that more than half of the edges are inside the expanders.[6]

Theorem 4.9 (Decomposition into Uniform Expanders)**.** *Given any graph/gradient/resistance instance $\mathcal{G}$ with n vertices, m edges, unit resistances, and gradient $\boldsymbol{g}^{\mathcal{G}}$, along with a parameter δ,* DECOMPOSE$(\mathcal{G}, \delta)$ *returns vertex disjoint subgraphs $\mathcal{G}_1, \mathcal{G}_2, \ldots$ in $O(m \log^7 n \log^2(n/\delta))$ time such that at least $m/2$ edges are contained in these subgraphs, and each $\mathcal{G}_i$ satisfies (for some absolute constant $c_{partition}$):*

1. *The graph $(V^{\mathcal{G}_i}, E^{\mathcal{G}_i})$ has conductance at least*

$$\phi = \frac{1}{c_{partition} \cdot \log^3 n \cdot \log(n/\delta)},$$

and degrees at least $\phi \cdot \frac{m}{3n}$, where $c_{partition}$ is an absolute constant.

2. *The projection of its gradient $\boldsymbol{g}^{\mathcal{G}_i}$ into the cycle space of $\mathcal{G}_i$, $\widehat{\boldsymbol{g}}^{\mathcal{G}_i}$ satisfies one of:*

   (a) $\widehat{\boldsymbol{g}}^{\mathcal{G}_i}$ is $O(\log^8 n \log^3(n/\delta))$-uniform,

$$\left(\widehat{\boldsymbol{g}}_e^{\mathcal{G}_i}\right)^2 \leq \frac{O\left(\log^{14} n \log^5(n/\delta)\right)}{m_i}\left\|\widehat{\boldsymbol{g}}^{\mathcal{G}_i}\right\|_2^2 \qquad \forall e \in E(\mathcal{G}_i).$$

   Here m_i is the number of edges in $\mathcal{G}^{\mathcal{G}_i}$.

[4] We use an instance and its underlying graph interchangeably in our discussion.

[5] $\boldsymbol{r}$ are uniform, so conductance is defined as in unweighted graphs. We use the standard definition of conductance. For graph $G = (V, E)$, the conductance of any $\emptyset \neq S \subsetneq V$ is $\phi(S) = \frac{\delta(S)}{\min(vol(S), vol(V \setminus S))}$ where $\delta(S)$ is the number of edges on the cut $(S, V \setminus S)$ and $vol(S)$ is the sum of the degree of nodes in S. The conductance of a graph is $\phi_G = \min_{S \neq \emptyset, V} \phi(S)$.

[6] Some of the expanders we find actually won't satisfy the projected gradient being α-uniform constraint (case 3(b) in Theorem 4.9). For those expanders, the projection of the gradient in the cycle space is tiny so we make it 0. This leads to the additive error in Theorem 3.6.

(b) *The ℓ_2^2 norm of $\widehat{\boldsymbol{g}}^{\mathcal{G}_i}$ is smaller by a factor of δ than the unprojected gradient:*

$$\left\| \widehat{\boldsymbol{g}}^{\mathcal{G}_i} \right\|_2^2 \leq \delta \cdot \left\| \boldsymbol{g}^{\mathcal{G}} \right\|_2^2.$$

Moreover, the min degree of any node in the expanders is up to a polylog factor close to the average degree. For the off-tree edges not included in these uniform expanders, we work on the pre-image[7] of them in the next iteration. That is, for any edge $\widehat{e}$ inside one of the expanders, we remove its pre-image from the instance $\mathcal{G}$, and work on the remaining off-tree edges in $\mathcal{G}$ in the next iteration. This iterative process terminates when the number of remaining off-tree edges is small enough (i.e. $\widetilde{O}(|E^{\mathcal{G}}|/\kappa)$). This takes $O(\log |E^{\mathcal{G}}|)$ iterations as a constant fraction of off-tree edges are moved to be inside the expanders each iteration.

Sparsify Uniform Expanders If we append a column containing the gradient of edges to the edge-vertex incidence matrix $\boldsymbol{B}$, the conditions of a α-uniform ϕ-expander is equivalent to each row of $\boldsymbol{B}$ having leverage score at most $\frac{n\alpha\phi^{-1}}{m}$ where n, m are number of nodes and edges. An underlying connection with the ℓ_p-norm row sampling result by Cohen and Peng [CP15] is that this is also a setting under which ℓ_q-norm functionals are preserved under uniform sampling. We refrain from developing a more complete picture of such machinery here, and will utilize ideas closer to routing on expanders [KM09, KLOS14] to show a cruder approximation in Section D.

Theorem 4.10 (Sampling Uniform Expanders)**.** *Given an α-uniform ϕ-expander $\mathcal{G} = (V^{\mathcal{G}} E^{\mathcal{G}}, r^{\mathcal{G}}, s^{\mathcal{G}}, \boldsymbol{g}^{\mathcal{G}})$ with m edges and vertex degrees at least $d_{\min}$, for any sampling probability τ satisfying*

$$\tau \geq c_{sample} \cdot \log n \cdot \left(\frac{\alpha}{m} + \frac{1}{\phi^2 d_{\min}} \right),$$

where c_{sample} is some absolute constant, SampleAndFixGradient$(\mathcal{G}, \tau)$ *w.h.p. returns a partial instance $\mathcal{H} = (H, r^{\mathcal{H}}, s^{\mathcal{H}}, \boldsymbol{g}^{\mathcal{H}})$ and maps $\mathcal{M}_{\mathcal{G}\to\mathcal{H}}$ and $\mathcal{M}_{\mathcal{H}\to\mathcal{G}}$. The graph H has the same vertex set as G, and H has at most $2\tau m$ edges. Furthermore, $r^{\mathcal{H}} = \tau \cdot r^{\mathcal{G}}$ and $s^{\mathcal{H}} = \tau^p \cdot s^{\mathcal{G}}$. The maps $\mathcal{M}_{\mathcal{G}\to\mathcal{H}}$ and $\mathcal{M}_{\mathcal{H}\to\mathcal{G}}$ certify*

$$\mathcal{H} \preceq_\kappa \mathcal{G} \text{ and } \mathcal{G} \preceq_\kappa \mathcal{H},$$

where $\kappa = m^{1/(p-1)} \phi^{-9} \log^3 n$.

4.4 Ultra-sparsification Algorithm and Error Analysis

Now we put all the pieces together. We need to show that adding together our individual sparsifiers results in a sparsifier of the overall graph. This is fairly immediate given the strong guarantees we established on the individual graphs. We also need to be able to repeatedly decompose and sparsify enough times that the overall graph becomes sparse. To address this issue, we use ideas from [KMP11] that suggest scaling up the tree from the tree routing section limits the error incurred during sampling. Here it again becomes important that because we rely on [SW18], we know exactly which edges belong to a sparsifier. This guarantee limits the interaction between sparsification of different expanders.

[7] By pre-image of $\widehat{e}$ we mean the original off-tree edge e that gets moved to $\widehat{e}$ in the tree-portal routing, i.e. $e = \text{Move}_{T,\widehat{V}}^{-1}(\widehat{e})$.

After constructing a low-stretch spanning tree T, we round each $\boldsymbol{r}_e^{\mathcal{G}}$ of off-tree edges $e \in E^{\mathcal{G}} \setminus T$ down to the nearest power of 2 (can be less than 1) if $\boldsymbol{r}_e^{\mathcal{G}} \geq \delta$, and round to 0 otherwise. This gives of $\log \frac{\|\boldsymbol{r}^{\mathcal{G}}\|_\infty}{\delta}$ bucket of edges with uniform resistances, and we work with one bucket of edges at a time, since the edges in a uniform expander need to have uniform $\boldsymbol{r}_e$. If $\mathcal{G}'$ is the instance of $\mathcal{G}$ after rounding the resistance of off-tree edges, it is easy to see the following error guarantee.

Lemma 4.11. *$\mathcal{G} \preceq_1 \mathcal{G}'$ with the identity mapping, , and for any flow solution $\boldsymbol{f}^{\mathcal{G}'}$ of $\mathcal{G}'$, again using the identity mapping, we have*

$$\mathcal{E}_{\mathcal{G}}(\frac{1}{2}\boldsymbol{f}^{\mathcal{G}'}) \geq \frac{1}{2}\mathcal{E}_{\mathcal{G}'}(\boldsymbol{f}^{\mathcal{G}'}) - \delta\left\|\boldsymbol{f}^{\mathcal{G}'}\right\|_2^2.$$

To avoid using too many symbols, we reuse $\mathcal{G}$ to refer to the original instance after the resistance rounding (i.e. the $\mathcal{G}'$ above). Denote E^r the subset of edges in $E^{\mathcal{G}} \setminus T$ containing edges with $\boldsymbol{r}_e = r$ for some particular r, note there are at most $\log \frac{\|\boldsymbol{r}^{\mathcal{G}}\|_\infty}{\delta}$ possible value of r. We work iteratively on the set E^r, starting with $E_0^r = E^r$. In the i-th iteration, we use FindPortal$(T, E_i^r, m/\kappa)$ (Lemma 4.4) to find a set of m/κ portal vertices for the edges remaining in E_i^r, note the low-stretch spanning tree is fixed through the process, but each iteration we find a new set of portals using FinalPortal as introduced in Section 4.2.

We then move edges in E_i^r using the tree-portal routing. We let $\widehat{G}_i^r$ to be the graph of the m/κ portal nodes and the off-tree edges between them. Note the number of edges in $\widehat{G}_i^r$ is $|E_i^r|$ and the number of nodes is m/κ.

So far we haven't specified the $\boldsymbol{r}, s, \boldsymbol{g}$ values on the edges in $\widehat{G}_i^r$, and these values will depend on the tree-portal routing as well as the average degree in $\widehat{G}_i^r$. For now we focus on discuss the edge set in our final sparsified instance, and assume we have $\boldsymbol{r}, s, \boldsymbol{g}$ values for $\widehat{G}_i^r$. We will come back to specify these quantities later.

We use Decompose (Theorem 4.9) on the graph $\widehat{G}_i^r$ to compute a collection of vertex disjoint sub-graphs $\{\widehat{G}_{i,1}^r, \widehat{G}_{i,2}^r, \ldots\}$, and at least half of the edges in $\widehat{G}_i^r$ are inside these sub-graphs. We let $\widehat{E}_i^r$ to be the edges contained in these sub-graphs, and $\widetilde{E}_i^r$ be the set of pre-images of edges in $\widehat{E}_i^r$ (in terms of tree-portal off-tree edge moving). We remove $\widetilde{E}_i^r$ from E_i^r and proceed to iteration $i+1$. If at the beginning of some iteration i, the size of E_i^r is at most $\widetilde{O}(m/\kappa)$, we leave them as off-tree edges, and denote E_{last}^r as the set containing them. Note for any r, the iterative process must finish in $O(\log \kappa)$ iterations as we start with $|E^r| < m$ edges. We do this for all r.

So far any edge in the original instance $\mathcal{G}$ we get either (1) a tree edge in T, or (2) an off-tree edge in a $\widehat{G}_{i,j}^r$ for some resistance value r, iteration i, and j-th expander computed in that iteration, or (3) an off-tree edge remaining in $\mathbf{E}_{last}^r$ for some resistance value r. There are $n-1$ edges in group (1), and $\widetilde{O}(m/\kappa)$ edges in group (3), so we can keep all these edges in the ultra-sparsifier $\mathcal{H}$. For the off-tree edges in group (2), we uniformly sample the edges in each $\widehat{G}_{i,j}^r$ to get a sparsified graph $\overline{H}_{i,j}^r$. Technically our sampling result only applies to an α-uniform ϕ-expander $\widehat{G}_{i,j}^r$ (i.e. case $3(a)$ in Theorem 4.9). If the $\widehat{G}_{i,j}^r$ we get back from Decompose is in case $3(b)$ of, we perturb the gradient on edges so that the projection of the gradient to the cycle space of the expander is 0, i.e. project the gradient to the space orthogonal to the cycle space. Then we have $\widehat{G}_{i,j}^r$ after perturbation is a 1-uniform ϕ-expander.

The edges in our final ultra-sparsifier $\mathcal{H}$ will be the tree edges in T, the off-tree edges in the $\mathbf{E}_{last}^r$'s over all resistance bucket value r, and the off-tree edges in the $\overline{H}_{i,j}^r$'s over all resistance r, iteration i and expander j. We argued about the size of all but the edges in the $\overline{H}^r_{i,j}$'s, which we will do now.

Sampling Probability We first specify the probability we sample each edge in $\widehat{G}^r_{i,j}$ to get $\overline{H}^r_{i,j}$ which we denote by $\tau_{r,i}$ (same across all the expanders, i.e. j's, for any resistance r and iteration i). By Theorem 4.10 we need the probability to be at least $c_{sample} \log n(d_{\min}^{-1}\phi^{-2} + \alpha m^{-1})$. Here c_{sample} is a fixed constant across all r, i, j's, and the guarantees on $\widehat{G}^r_{i,j}$ from Theorem 4.9 allow us to use some fixed polylog$n$ as ϕ^{-2} and α across all r, i, j's. The only parameter that varies across different r, i's is $d_{\min}$, a lower bound on the minimum vertex degree in $\widehat{G}^r_{i,j}$, which by Theorem 4.9 is within a (fixed) polylog factor of the average degree in $\widehat{G}^r_i$. As there are $m_{r,i}$ edges and m/κ nodes in $\widehat{G}^r_i$, the average degree is $m_{r,i}\kappa/m$. Thus, we can write $\tau_{r,i} = \frac{c_1 m \log^{c_2} n}{\kappa m_{r,i}}$ for some global constants c_1, c_2, and since both $m_{r,i}$ and κ is at most m, $\tau_{r,i}$ satisfies the requirement on τ in Theorem 4.10. With this particular choice of $\tau_{r,i}$ we can use SampleAndFixGradient to sample $\widehat{G}^r_{i,j}$ and the the guarantees from Theorem 4.10. Now we can prove the statement about the number of off-tree edges in $\mathcal{H}$.

Lemma 4.12. *The total number of edges over all $\overline{H}^r_{i,j}$'s is $\widetilde{O}(\frac{m}{\kappa})$ with high probability.*

Proof. Pick any r, i, recall when we call Decompose in that iteration, we have $\widehat{G}^r_i$ with uniform $\boldsymbol{r}$, $m_{r,i} = |E^r_i|$ edges and $n_i = m/\kappa$ nodes. From the previous discussion of the sampling probability, we know it is sufficient to call SampleAndFixGradient on $\widehat{G}^r_{i,j}$ with probability

$$\tau_{r,i} = \frac{c_1 m \log^{c_2} n}{\kappa m_{r,i}}$$

for some constants c_1, c_2. By Theorem 4.10, the number of edges in $\overline{H}^r_{i,j}$'s over all j is at most $\widetilde{\Theta}(\frac{m}{\kappa})$ with high probability since over all j the $\widehat{G}^r_{i,j}$'s contain $\Theta(m_{r,i})$ edges.

Since for each r the number of iterations is at most $i \le \log \kappa$, and there are $\log \frac{\|\boldsymbol{r}^{\mathcal{G}}\|_\infty}{\delta}$ possible r values, the final bound in the lemma follows from summing over all r, i. Note we can hide all log factors as $\log n$ factors by our assumption in Theorem 3.6 that $\log\|\boldsymbol{r}^{\mathcal{G}}\|_\infty$ and $\log \frac{1}{\delta}$ are both polylog in n. □

Now we discuss the $\boldsymbol{g}, \boldsymbol{r}, s$ values we put on the edges in all the steps. Note we need the final instance $\mathcal{H}$ to have a uniform scalar $s^{\mathcal{H}}$ for every $|f_e|^p$ term, so we can recursively optimize the instance. However, in the intermediate steps, we will divide the instance into sub-instances induced by the different subsets of edges, e.g. $\widehat{G}^r_{i,j}$'s, and later combine sub-instances induced by the sampled sub-graphs $\overline{H}^r_{i,j}$'s to get $\mathcal{H}$. Each of these sub-instances will have its own scalar, e.g. $s^G_{r,i}, s^H_{r,i}$, but in general they won't necessarily have the same value across different sub-instances. Notation-wise, in the following discussion, we assume each edge has its own scalar $\boldsymbol{s}_e$ associated with the term $|f_e|^p$ in the intermediate instances. Eventually, the different scaling we do to s in the intermediate steps will cancel so that in $\mathcal{H}$ we have the scalar $s^{\mathcal{H}}$. The input $\mathcal{G}$ has a uniform scalar $s^{\mathcal{G}}$, and we will make $s^{\mathcal{H}} = s^{\mathcal{G}}$.

Now we specify the $\boldsymbol{g}$, $\boldsymbol{r}$ and s values of the edges in the final instance $\mathcal{H}$ as well as in some of the key intermediate sub-instances we consider.

Algorithm 4 Producing Ultra-Sparsifier $\mathcal{H}$ with unit $s^{\mathcal{H}} = s^{\mathcal{G}}$

procedure UltraSparsify($\mathcal{G}$, κ,δ)
  $T \leftarrow \mathrm{LSST}(\boldsymbol{r}^{\mathcal{G}})$. (low-stretch spanning tree)
  Initiate $\mathcal{H}$ with T, and the identity flow mapping.
  Round $\boldsymbol{r}^{\mathcal{G}}$ down to nearest power of 2, or 0 if less than δ
  $\widehat{n} \leftarrow m/\kappa$ (number of portal nodes per batch)
  for Each bucket of resistance value r **do**
    Let $i \leftarrow 0$, $E^r \leftarrow \{e | e \in E^{\mathcal{G}} \setminus T, \boldsymbol{r}^{\mathcal{G}}_e = r\}$
    while E^r has more than $\widetilde{O}(m/\kappa)$ off-tree edges **do**
      Let $m_{r,i}$ be the number of edges in E^r.
      Find $\widehat{n}$ portal nodes to short-cut tree routing:

$$\widehat{V} \leftarrow \textsc{FindPortal}(T, E^r, \widehat{n}).$$

      Route edges in E^r along T, using portal nodes to short-cut tree-portal routing:
      $\widehat{G}^r_i \leftarrow \textsc{TreePortalRoute}(E^r, T, \widehat{V})$,,
      Decompose the graph after tree-portal routing into uniform expanders:

$$\left\{\widehat{G}^r_{i,1}, \widehat{G}^r_{i,2}, \ldots\right\} \leftarrow \textsc{Decompose}\left(\widehat{G}^r_i, \delta/m^5\right).$$

      Remove the pre-image of edges in $\widehat{G}^r_{i,1}, \widehat{G}^r_{i,2}, \ldots$ from E^r.
      Set $\tau_{r,i} \leftarrow \frac{c_1 m \log^{c_2} n}{m_{r,i}\kappa}$ (for sampling $\widehat{G}^r_{i,j}$'s in SampleAndFixGradient)
      for each $\widehat{G}^r_{i,j}$ **do**
        Rescale the gradients and ℓ_p^p scalar as

$$\boldsymbol{r}^{\widehat{G}^r_{i,j}} = r\kappa \log^2 n \tag{10}$$

$$s^{\widehat{G}^r_{i,j}} = \tau_{r,i}^{-p} \cdot s^{\mathcal{G}} \tag{11}$$

        Let $\overline{H}^r_{i,j} \leftarrow \textsc{SampleAndFixGradient}\left(\widehat{G}^r_{i,j}, \tau_{r,i}\right)$.
        Add $\overline{H}^r_{i,j}$ to H, and incorporate the flow mappings between $\widehat{G}^r_{i,j}$ and $\overline{H}^r_{i,j}$
            (composed with the tree-portal routing between $\widehat{G}^r_{i,j}$ and
            its pre-image) to the mapping between $\mathcal{G}$ and $\mathcal{H}$.
      $i \leftarrow i + 1$
    Add all remaining edges of E^r to H with the identity flow mapping on them
  return $\mathcal{H}$, $\mathcal{M}_{\mathcal{H}\to\mathcal{G}}$, and $\kappa_{\mathcal{H}\to\mathcal{G}} = \widetilde{O}(\kappa m^{3/(p-1)})$.

Algorithm 5 Tree-Portal Routing of Edges

procedure TreePortalRoute(E,T,$\widehat{V}$)
  Initialize $\widehat{E} \leftarrow \emptyset$
  for each $e = \{u, v\} \in E$ **do**
    Let $\widehat{e} \leftarrow \text{Move}_{T,\widehat{V}}(e)$, and $P_{T,\widehat{V}}(u, v)$ be its tree-portal path.
    (See Definition 4.1)
    Let $\boldsymbol{r}_{\widehat{e}}$,$\boldsymbol{s}_{\widehat{e}}$ be the same as $\boldsymbol{r}_e$ and $\boldsymbol{s}_e$.
    Set $\boldsymbol{g}_{\widehat{e}}$ so that sending 1 unit of flow from u to v along $P_{T,\widehat{V}}(u, v)$ has the same flow dot gradient as $\boldsymbol{g}_e$, i.e. the flow dot gradient of sending directly along e. Note all edges on $P_{T,\widehat{V}}(u, v)$ other than $\widehat{e}$ have known gradients.
    Add $\widehat{e}$ to $\widehat{E}$ with $\boldsymbol{g}_{\widehat{e}}$, $\boldsymbol{r}_{\widehat{e}}$, $\boldsymbol{s}_{\widehat{e}}$ as specified. Note $\widehat{E}$ may contain multi-edges.
  return $\widehat{E}$

1. $e \in T$: The gradient, resistance and $\boldsymbol{s}_e$ on these edges in $\mathcal{H}$ remain the same as in $\mathcal{G}$, that is $\boldsymbol{g}^{\mathcal{H}}_e = \boldsymbol{g}^{\mathcal{G}}_e$, $\boldsymbol{r}^{\mathcal{H}}_e = \boldsymbol{r}^{\mathcal{G}}_e$, and $\boldsymbol{s}_e = s^{\mathcal{G}}$.

2. $e \in E^r_{last}$: These off-tree edges remain at the end for each bucket E^r. We keep their gradient, resistance, and $\boldsymbol{s}_e = s^{\mathcal{G}}$ as in the original instance.

3. $\widehat{e} \in \widehat{G}^r_{i,j}$, the j-th expander computed in iteration i for resistance r: In the intermediate sub-instance induced by $\widehat{G}^r_{i,j}$, we have $\boldsymbol{r}_{\widehat{e}} = r\kappa \log^2 n$, $\boldsymbol{s}_{\widehat{e}} = \tau^{-p}_{r,i} s^{\mathcal{G}}$. For the gradient on $\widehat{e}$, recall $\widehat{e}$ is the image of some off-tree edge e under the mapping $\text{Move}_{T,\widehat{V}^r_i}$ where $\widehat{V}^r_i$ is the set of portals in the i-th iteration for resistance r. Under the tree-portal routing, any flow along $e = (u, v)$ will be rerouted along the tree-portal path $P_{T,\widehat{V}^r_i}(u, v)$. We want the linear term (i.e. gradient times flow) in the objective function to remain the same under this rerouting, so routing 1 unit of flow from u to v along $P_{T,\widehat{V}^r_i}(u, v)$ should give the same dot product with the gradients as routing 1 unit of flow from u to v along e in the original instance (i.e. $\boldsymbol{g}^{\mathcal{G}}_e$). As the only off-tree edge on the tree-portal path is $\widehat{e}$, and we are keeping the original gradients on all the tree edges, this uniquely determines $\boldsymbol{g}_{\widehat{e}}$.

4. $\overline{e} \in \overline{H}^r_{i,j}$ for some r, i, j: As specified in Theorem 4.10, if the edge $\overline{e}$ is sampled (with uniform probability $\tau_{r,i}$), and $\boldsymbol{r}_{\overline{e}}$,$\boldsymbol{s}_{\overline{e}}$ are their corresponding values in $\widehat{G}^r_{i,j}$ scaled up by $\tau_{r,i}$ and $\tau^p_{r,i}$ respectively. In particular we get back $\boldsymbol{s}_{\overline{e}} = s^{\mathcal{G}}$ as the $\tau^p_{r,i}$ scaling cancels the $\tau^{-p}_{r,i}$ scaling in $\widehat{G}^r_{i,j}$.

Note all the edges in our $\mathcal{H}$ (i.e. group $1, 2, 4$ above) end up with the same scalar $s^{\mathcal{H}} = s^{\mathcal{G}}$.

Now we bound the approximation error. For simplicity, we carry out the analysis ignoring the additive errors in the bound, and defer the discussion of them to the end. In particular, additive errors come in at two cases. The first is when we round an original resistance to 0 when it is less than δ, and the second is in Decompose, we may get an expander whose projected gradient is not $\widetilde{O}(1)$-uniform but has tiny norm (i.e. case $3(b)$), and we zero out its projection to the cycle space before sampling. For now we assume we don't have these cases.

We summarize the notations in our algorithm and analysis in Table 1. We explicitly point out whenever we change the gradient, resistance or s value on an edge. We will use instances and their

Table 1: Glossary of Notations in Algorithm and Analysis.

Notations in Ultrasparsify	
$\mathcal{G}$	Input instance with $\left(V^{\mathcal{G}}, E^{\mathcal{G}}, \boldsymbol{g}^{\mathcal{G}}, \boldsymbol{r}^{\mathcal{G}}, s^{\mathcal{G}}\right)$.
T	Low stretch spanning tree of $\mathcal{G}$ (stretch with respect to $\boldsymbol{r}^{\mathcal{G}}$).
E^r	All in $E^{\mathcal{G}} \setminus T$ whose resistance after rounding is r.
E_i^r	The remaining edges in E^r at the i-th iteration of tree-portal routing E^r.
$\widehat{G}_i^r$	The image of edges in E_i^r by the mapping $\textsc{Move}_{T,\widehat{V}}$, i.e. moving off-tree edges along tree-portal path. The gradients of edges in $\widehat{G}_i^r$ are set to preserve the linear flow dot gradient term under tree-portal routing.
$m_{r,i}$	The number of edges in E_i^r (also the size of $\widehat{G}_i^r$).
$\widehat{G}_{i,j}^r$	The j-th expander we get from decomposing $\widehat{G}_i^r$. Edges keep their gradients from $\widehat{G}_i^r$, and $\boldsymbol{r}, s$ are scaled.
$\widehat{E}_i^r$	The union of edges contained in the expanders $\widehat{G}_{i,j}^r$ (i.e. over all j's).
$\widetilde{E}_i^r$	The pre-image of edges in $\widehat{E}_i^r$.
$\mathbf{E}_{last}^r$	The set of edges remaining in E^r after the last iteration for r.
$\tau_{r,i}$	The probability we use in SampleAndFixGradient to uniformly sample $\widehat{G}_{i,j}^r$.
$\overline{H}_{i,j}^r$	The sparsified graph of $\widehat{G}_{i,j}^r$ computed by SampleAndFixGradient. $\boldsymbol{g}, \boldsymbol{r}, s$ on edges are computed by the subroutine.
Additional notations in the analysis	
$\overline{G}$	The instance with the same edge set as $\mathcal{G}$. Note† $E^{\mathcal{G}} = T + \sum_r E_{last}^r + \sum_{r,i} \widetilde{E}_i^r$. Edges in $\overline{G}$ has the same $\boldsymbol{g}, \boldsymbol{r}, s$ as in $\mathcal{G}$ except for those in $\sum_{r,i} \widetilde{E}_i^r$. For any resistance r and iteration i, $e \in \widetilde{E}_i^r$ has the same gradient as in $\mathcal{G}$, but $\boldsymbol{r}_e = r\kappa \log^2 n$, $s_e = \tau_{r,i}^{-p} s^{\mathcal{G}}$ are scaled.
$\overline{G}_i^r$	The instance $\overline{G}$ restricted to the set of edges in $\widetilde{E}_i^r$.
$\overline{G}_{rest}$	The instance $\overline{G}$ restricted to the set of edges in $\sum_r E_{last}^r$.

† We use addition on sets as union but signify that the sets are disjoint.

underlying graphs interchangeably, and when we refer to a subgraph as an instance, it will be clear what are the $\boldsymbol{g}, \boldsymbol{r}, s$ values for the instance.

First we let $\overline{G}$ be the instance on the same nodes and edges as $\mathcal{G}$, but for any $e \in \widetilde{E}_i^r$ (i.e. e will be mapped to some $\widehat{e}$ in $\widehat{G}_{i,j}^r$), we rescale the resistance and s to be $\boldsymbol{r}_e = r\kappa \log^2 n$, and $\boldsymbol{s}_e = \tau_{r,i}^{-p} s^{\mathcal{G}}$. Note the gradient of e in $\overline{G}$ stays the same as in $\mathcal{G}$. We first bound the approximation error between $\mathcal{G}$ and this rescaled instance $\overline{G}$.

Lemma 4.13. *$\mathcal{G} \preceq_{\widetilde{O}(m^{1/(p-1)}\kappa)} \overline{G} \preceq_1 \mathcal{G}$ with the identity mapping in both directions.*

Proof. For any edge e, we have $\boldsymbol{g}_e^{\overline{G}} = \boldsymbol{g}_e^{\mathcal{G}}$. As to the ℓ_p^p scalar, we have either $\boldsymbol{s}_e^{\overline{G}} = \boldsymbol{s}_e^{\mathcal{G}}$, or if e is eventually moved to some $\widehat{G}_{i,j}^r$ then

$$\boldsymbol{s}_e^{\overline{G}} = \tau_{r,i}^{-p} s^{\mathcal{G}} = \left(\frac{m_{r,i}\kappa}{c_1 m \log^{c_2} n}\right)^p s^{\mathcal{G}}$$

as $m_{i,r} \geq \widetilde{O}m/\kappa$ or otherwise we would have stopped for resistance value r, we can assume $m_{r,i}\kappa \geq c_1 m \log^{c_2} n$ so

$$\boldsymbol{s}_e^{\mathcal{G}} \leq \boldsymbol{s}_e^{\overline{G}} \leq (\kappa^{p/(p-1)})^{p-1} \boldsymbol{s}_e^{\mathcal{G}} \leq (m^{1/(p-1)}\kappa)^{p-1} \boldsymbol{s}_e^{\mathcal{G}}$$

where the second inequality is by $m_{r,i} \leq m$, and the third inequality is by $\kappa < m$. Similar calculation gives $\boldsymbol{r}_e^{\mathcal{G}} \leq \boldsymbol{r}_e^{\overline{G}} \leq \kappa \log^2 n \cdot \boldsymbol{r}_e^{\mathcal{G}}$. Our result directly follow by Lemma 2.13. ☐

Now we break $\overline{G}$ into sub-instances induced on the disjoint edge sets. Let $\overline{E}_i^r$ be the instance of $\overline{G}$ restricted to edges in $\widetilde{E}_i^r$, T the instance restricted to the tree edges, and $\overline{G}_{rest}$ the instance restricted to edges in any of the E_{last}^r's. When use addition as union on sets when the sets are disjoint. The objective of the sum of two instances is simply the sum of the individual instances objectives.

Lemma 4.14. *For any resistance value r, round i, we have*

$$T + \overline{E}_i^r \preceq_{\widetilde{O}(m^{1/(p-1)})} T + \widehat{E}_i^r \preceq_{\widetilde{O}(m^{1/(p-1)})} T + \overline{E}_i^r$$

where the flow mapping is the tree-portal routing and its reverse.

Proof. Fix any resistance value r and iteration i, the set of remaining off-tree edges of resistance r in iteration i is E_i^r, and these edges have a total stretch at most $O(m \log n \log\log n)$ with T by Lemma 4.3, and $E_i^r = m_{r,i}$. As we use FindPortal to get a set of m/κ portal nodes $\widehat{V}$ in that iteration, by Lemma 4.4, for any edge e' on T, we have in $T + E_i^r$

$$W_{e'} \stackrel{\text{def}}{=} \sum_{e \in E_i^r : e' \in P_{T,\widehat{V}}(e)} \mathsf{Str}_{T,\widehat{V}}(e) \leq \frac{10}{\widehat{n}} \sum_{e \in E_i^r} \mathsf{Str}_{T,\widehat{V}}(e) \leq 10\kappa \log n \log\log n \leq \kappa \log^2 n$$

and

$$K_{e'} \stackrel{\text{def}}{=} \left| e \in E_i^r : e' \in P_{T,\widehat{V}}(e) \right| \leq \frac{10\widehat{m}}{\widehat{n}} \leq \frac{10\kappa m_{r,i}}{m}$$

We first look at the direction from from $T + \overline{E}_i^r$ to $T + \widehat{E}_i^r$. Let $\boldsymbol{f}$ be the flow in $T + \overline{E}_i^r$, and $\widehat{\boldsymbol{f}}$ be the tree-portal routing of $\boldsymbol{f}$. In the tree-portal routing, flow on tree edges is mapped to the same flow, while any flow along an off-tree edge $\overline{e} = (u, v) \in \overline{E}_i^r$ is rerouted along the tree-portal path

$P_{T,\widehat{V}}(u,v)$. This rerouting clearly preserves the residue between $\boldsymbol{f}, \widehat{\boldsymbol{f}}$, and if $\widehat{e} \in \widehat{E}_i^r$ is the image of (u,v), its gradient $\boldsymbol{g}_{\widehat{e}}$ in $\widehat{E}_i^r$ is by construction set to be the value which preserves the linear term in the objective function for $\boldsymbol{f}$ and $\widehat{\boldsymbol{f}}$. The cost of ℓ_2^2 and ℓ_p^p terms for $f_{\overline{e}}$ is the same as the corresponding costs for $\widehat{\boldsymbol{f}}_{\widehat{e}}$, since $\widehat{e}$ is only used for the rerouting of $\overline{e}$ (so $|\widehat{\boldsymbol{f}}_{\widehat{e}}| = |f_{\overline{e}}|$), and they have the same $\boldsymbol{r}, \boldsymbol{s}$ values. Thus, the contribution to the ℓ_2^2, ℓ_p^p terms in objective function from the off-tree edges are the same for $\boldsymbol{f}$ and $\widehat{\boldsymbol{f}}$. The only extra cost comes from the ℓ_2^2 and ℓ_p^p terms of tree edges for $\widehat{\boldsymbol{f}}$ since we put additional flow through them. First consider the sum of the ℓ_p^p terms over all tree edges for $\widehat{\boldsymbol{f}}$ in $T + \widehat{E}_i^r$. Recall we don't scale the s value for tree edges, so the scalar is still $s^{\mathcal{G}}$ on tree edges, while for off-tree edges in $\overline{E}_i^r$, the value s is scaled to be $\left(\frac{\kappa m_{r,i}}{c_1 m \log^{c_2} n}\right)^p s^{\mathcal{G}}$

$$
\begin{aligned}
\sum_{e' \in T} s^{\mathcal{G}} \left|\widehat{\boldsymbol{f}}_{e'}\right|^p &= \sum_{e' \in T} s^{\mathcal{G}} \left| \sum_{\overline{e}: e' \in P_{T,\widehat{V}}(\overline{e})} \boldsymbol{f}_{\overline{e}} \right|^p \\
&= \sum_{e' \in T} s^{\mathcal{G}} K_{e'}^p \left| \sum_{\overline{e}: e' \in P_{T,\widehat{V}}(\overline{e})} \frac{1}{K_{e'}} \boldsymbol{f}_{\overline{e}} \right|^p \\
&\le \sum_{e' \in T} s^{\mathcal{G}} K_{e'}^p \sum_{\overline{e}: e' \in P_{T,\widehat{V}}(\overline{e})} \frac{1}{K_{e'}} |\boldsymbol{f}_{\overline{e}}|^p && \text{(Using Jensen's inequality)} \\
&= \sum_{e' \in T} s^{\mathcal{G}} K_{e'}^{p-1} \sum_{\overline{e}: e' \in P_{T,\widehat{V}}(\overline{e})} |\boldsymbol{f}_{\overline{e}}|^p \\
&\le \sum_{\overline{e}} |\boldsymbol{f}_{\overline{e}}|^p \sum_{e' \in P_{T,\widehat{V}}(\overline{e})} s^{\mathcal{G}} K_{e'}^{p-1} \\
&\le \sum_{\overline{e}} |\boldsymbol{f}_{\overline{e}}|^p m \cdot s^{\mathcal{G}} K_{e'}^{p-1} && \text{(Tree- portal path's length} < m) \\
&\le \sum_{\overline{e}} (10 c_1 \log^{c_2} n)^{p-1} m \cdot \boldsymbol{s}_{\overline{e}} |\boldsymbol{f}_{\overline{e}}|^p
\end{aligned}
$$

So the ℓ_p^p term goes up by at most a factor $(10c_1 \log^{c_2} n)^{p-1} m$. Similar calculation shows that the ℓ_2^2 term goes up by at most a constant factor by the tree-portal routing. Thus, we get $T + \overline{E}_i^r \preceq_{\widetilde{O}(m^{1/(p-1)})} T + \widehat{E}_i^r$. The other direction is symmetric using the reverse tree-portal routing, and the calculation stays the same since the tree-portal routing in reverse incurs the same load/congestion on tree edges. ☐

If we put the $\overline{G}_i^r$ over all resistance r's and round i's together, we get

Lemma 4.15.

$$
T + \sum_{r,i} \overline{G}_i^r \preceq_{\widetilde{O}(m^{1/(p-1)})} T + \sum_{r,i} \widehat{E}_i^r \preceq_{\widetilde{O}(m^{1/(p-1)})} T + \sum_{r,i} \overline{G}_i^r
$$

The sum is over all possible resistance value r's, and over all iterations i for r.

Proof. By Lemma 4.14 and Lemma 2.12 we have

$$
\bigcup_{r,i} T + \overline{G}_i^r \preceq_{\widetilde{O}(m^{1/(p-1)})} \bigcup_{r,i} T + \widehat{E}_i^r \preceq_{\widetilde{O}(m^{1/(p-1)})} \bigcup_{r,i} T + \overline{G}_i^r
$$

Note the $\overline{G}_i^r$'s (and the $\widehat{E}_i^r$'s) are disjoint for different resistance values or different iterations, thus these edges contribution to the objective function value simply adds up. For the tree edges, since there are at most $\log^2 n$ different pairs of resistance and iteration pairs, we have

$$T \preceq_1 \bigcup_{r,i} T + \preceq_{\log^2 n} T$$

by considering the mapping that split flow on one tree edge to $\log^2 n$ copies of it and the reverse mapping of merging. Note $|a_1|^x + \ldots + |a_1|^x \leq (|a_1| + \ldots + |a_k|)^x \leq k(|a_1|^x + \ldots + |a_1|^x)$. This gives the final result we want. □

Note that $\overline{G}$ is the disjoint union of $T + \sum_{r,i} \overline{G}_i^r + \overline{G}_{rest}$, while $\mathcal{H}$ is the disjoint union of $T + \sum_{r,i} \overline{H}_i^r + \overline{G}_{rest}$. Thus, we can show the following

Lemma 4.16. $\overline{G} \preceq_{\widetilde{O}(m^{2/(p-1)})} \mathcal{H} \preceq_{\widetilde{O}(m^{2/(p-1)})} \overline{G}$.

Proof. Recall for each resistance value r, in the i-th round, $\widehat{G}_{i,j}^r$ is the j-th uniform expander we find, and $\overline{H}_{i,j}^r$ is the sparsified graph of $\widehat{G}_{i,j}^r$.

$$\begin{aligned}
\overline{G} &= T + \overline{G}_{rest} + \sum_{r,i} \overline{G}_i^r && \text{(valid as the sets are disjoint)} \\
&\preceq_{\widetilde{O}(m^{1/(p-1)})} T + \overline{G}_{rest} + \sum_{r,i} \widehat{E}_i^r && \text{(Lemma 4.15)} \\
&= T + \overline{G}_{rest} + \sum_{r,i,j} \widehat{G}_{i,j}^r && \text{($\widehat{E}_i^r$ is the disjoint union of $\widehat{G}_{i,j}^r$ over all j)} \\
&\preceq_{\widetilde{O}(m^{1/(p-1)})} T + \overline{G}_{rest} + \sum_{r,i,j} \widehat{H}_{i,j}^r && \text{(By Theorem 4.10, and sets being disjoint)} \\
&= \mathcal{H}
\end{aligned}$$

$\overline{G} \preceq_{\widetilde{O}(m^{2/(p-1)})} \mathcal{H}$ follows by taking the composition of all the intermediate steps, and multiplying the approximation error by Lemma 2.10. The other direction is similar. □

Now we can prove the main ultra-sparsification theorem.

Proof of Theorem 3.6. Other than the additive error terms and the self-loops, everything in the theorem statement follow directly from Lemma 4.12 (the number of off-tree edges), and composition of Lemma 4.13 with Lemma 4.16 (the approximation error). We explicitly spell out the flow mappings between $\mathcal{G}$ and $\mathcal{H}$. We start with the $\mathcal{G}$ to $\mathcal{H}$ direction. We break the flow in $\mathcal{G}$ as the sum of flow on disjoint edge subsets T,$\overline{G}_{rest}$, and $\widetilde{E}_i^r$, specify the mapping from each piece to $\mathcal{H}$, and later take the sum of the mappings. The mapping from T and $\overline{G}_{rest}$ to $\mathcal{H}$ is just the identity. For flow on $\widetilde{E}_i^r$, we get a flow on $T + \widehat{E}_i^r$ by tree-portal routing. As $\widehat{E}_i^r$ is the sum of $\widehat{G}_{i,j}^r$'s, for the flow mapped to $\widehat{G}_{i,j}^r$, we map it to a flow on $\overline{H}_{i,j}^r$ using the flow mapping in SampleAndFixGradient. We add these mapping over all j's to get a mapping from the flow on $T + \widehat{E}_i^r$ to a flow on $\mathcal{H}$, and take the composition with the tree-portal routing to get a mapping from $\widetilde{E}_i^r$ to $\mathcal{H}$. Summing over all r, i (together with the identity on T and $\overline{G}_{rest}$ gives the mapping from $\mathcal{G}$ to $\mathcal{H}$. The mapping from $\mathcal{H}$ to $\mathcal{G}$ is symmetric, and in the part from $\widehat{E}_i^r$ to $T + \widetilde{E}_i^r$ we use the reverse of tree-portal routing.

All the subroutines take nearly linear time, and we have at most $\log n$ different r, and for each r there are at most $\log m$ iterations, so the overall running time is $\widetilde{O}(m)$. The flow mappings can also be applied in $\widetilde{O}(m)$ time, and they are linear maps.

Now we look at the additive error terms. In particular, additive errors come in at two places. The first is when we round an original resistance to 0 when it is less than δ, and we have Lemma 4.11 to bound the error (at that step). The second place is in Decompose (Algorithm 6), we may get an expander $\widehat{G}_{i,j}^r$ whose projected gradient is not α-uniform but has tiny norm (i.e. case 3(b)), and we zero out its projection to the cycle space before sampling to make it 1-uniform. If we have a flow f on such an $\widehat{G}_{i,j}^r$, the additive error is in the linear term, and is equal to the dot product of f with the removed gradient. We let $\boldsymbol{g}_i^r$, $\boldsymbol{g}_{i,j}^r$ be the gradient on edges in $\widehat{G}_i^r$,$\widehat{G}_{i,j}^r$ respectively, and $\widehat{\boldsymbol{g}}_i^r$, $\widehat{\boldsymbol{g}}_{i,j}^r$ as the projection of $\boldsymbol{g}_i^r$ (and $\boldsymbol{g}_{i,j}^r$) to the cycle space of $\widehat{G}_i^r$ (and $\widehat{G}_{i,j}^r$). We remove $\widehat{\boldsymbol{g}}_{i,j}^r$ from the gradient $\boldsymbol{g}_{i,j}^r$ when $\widehat{\boldsymbol{g}}_{i,j}^r \leq \delta' \widehat{\boldsymbol{g}}_i^r$ for some parameter δ', so the additive error we introduce is $f^T \widehat{\boldsymbol{g}}_{i,j}^r$, which is at most $\|f\|_2 \left\|\widehat{\boldsymbol{g}}_{i,j}^r\right\|_2$, which is in turn at most $\delta' \|f\|_2 \|\boldsymbol{g}_i^r\|_2$ as $\widehat{\boldsymbol{g}}_i^r$ is a projection of $\boldsymbol{g}_i^r$. Now we look at how this additive error propagates in terms of the overall approximation error between $\mathcal{G}$ and $\mathcal{H}$. We will get an additional factor m when we combine the additive errors over all the individual expanders where we carry out this perturbation. Note we are not really introducing more error here, but simply because $\sqrt{m}\|\sum_i f_i\| \geq \sum_i \|f_i\| \geq \|\sum_i f_i\|$ when f_i's have disjoint support and total size m. The additive error is also amplified through the intermediate steps, but since the multiplicative approximation errors are $m^O 1/p$, we lose at most another polynomial factor. Additional polynomial factor comes in because the norm of the gradient vector after tree-routing can be off by a polynomial factor comparing to the norm of the original gradient. However, overall the blowup is at most polynomial, and we use a polynomially smaller δ' in Decompose to accommodate these factors to get the additive error in our final result. The same argument applies to the additive error introduced by resistance rounding (e.g. round to 0 when the gradient is at most δ / m^c for some large enough c). ☐

We brief go over the case when tree-portal routing gives self-loops. We treat self-loops the same way as the edges that are in the uniform expanders except they don't go through the expander decomposition and sampling steps. Once we get a self-loop $\widehat{e}$ from tree-portal routing of some edge $e \in E^{\mathcal{G}}$, we add $\widehat{e}$ to $\mathcal{H}$, where the gradient on $\widehat{e}$ is set (the same way as non self-loops) to preserve the flow dot gradient term under tree-portal routing. We remove its pre-image e from E_i^r, but if in some iteration, more than half of the edges in E_i^r are mapped to self-loops by tree-portal routing, we skip the decomposition and sampling steps also for other edges, as we don't have a dense enough graph between the portal nodes to sparsify. We still have the size of E_i^r drop by at least 1/2 across each iteration as before. The final caveat is that since self-loops don't go through SampleAndFixGradient, and thus their s values are not scaled to be the same as the rest of the edges in $\mathcal{H}$. This is not an issue because we will remove them from the instance and optimize them individually (see Lemma 3.5), so they won't exist in the instance that we recursively solve, so uniform s scalar is not required for them.

5 Decomposing into Uniform Expanders

In this section we prove our decomposition result necessary for finding large portions of edges that can be sampled. This and the subsequent sampling step in Appendix D are critical for reducing the number of edges between portal vertices, after they were routed there in Line 12 of UltraSparsify (Algorithm 4). The main algorithmic guarantees can be summarized as below in Theorem 4.9.

Theorem 4.9 (Decomposition into Uniform Expanders)**.** *Given any graph/gradient/resistance instance $\mathcal{G}$ with n vertices, m edges, unit resistances, and gradient $\boldsymbol{g}^{\mathcal{G}}$, along with a parameter δ,* Decompose$(\mathcal{G}, \delta)$ *returns vertex disjoint subgraphs $\mathcal{G}_1, \mathcal{G}_2, \ldots$ in $O(m \log^7 n \log^2(n/\delta))$ time such that at least $m/2$ edges are contained in these subgraphs, and each $\mathcal{G}_i$ satisfies (for some absolute constant $c_{partition}$):*

1. *The graph $(V^{\mathcal{G}_i}, E^{\mathcal{G}_i})$ has conductance at least*

$$\phi = \frac{1}{c_{partition} \cdot \log^3 n \cdot \log(n/\delta)},$$

and degrees at least $\phi \cdot \frac{m}{3n}$, where $c_{partition}$ is an absolute constant.

2. *The projection of its gradient $\boldsymbol{g}^{\mathcal{G}_i}$ into the cycle space of $\mathcal{G}_i$, $\widehat{\boldsymbol{g}}^{\mathcal{G}_i}$ satisfies one of:*

   (a) $\widehat{\boldsymbol{g}}^{\mathcal{G}_i}$ is $O(\log^8 n \log^3(n/\delta))$-uniform,

$$\left(\widehat{\boldsymbol{g}}_e^{\mathcal{G}_i}\right)^2 \leq \frac{O\left(\log^{14} n \log^5(n/\delta)\right)}{m_i} \left\|\widehat{\boldsymbol{g}}^{\mathcal{G}_i}\right\|_2^2 \qquad \forall e \in E(\mathcal{G}_i).$$

   Here m_i is the number of edges in $\mathcal{G}^{\mathcal{G}_i}$.

   (b) The ℓ_2^2 norm of $\widehat{\boldsymbol{g}}^{\mathcal{G}_i}$ is smaller by a factor of δ than the unprojected gradient:

$$\left\|\widehat{\boldsymbol{g}}^{\mathcal{G}_i}\right\|_2^2 \leq \delta \cdot \left\|\boldsymbol{g}^{\mathcal{G}}\right\|_2^2.$$

We will obtain the expansion properties via expander decompositions. Specifically we will invoke the following result from [SW18] as a black box.

Lemma 5.1. *There is a routine* ExpanderDecompose *that when given any graph G and any degrees $\boldsymbol{d}$ such that $\boldsymbol{d}_u \geq deg_G(u)$ for all u, along with a parameter $0 < \phi < 1$,* ExpanderDecompose$(G, \boldsymbol{d}, \phi)$ *returns a partition of the vertices of G into $V_1, V_2, \ldots$ in $O(m\phi^{-1} \log^4 n)$ time such that $G[V_i]$ has conductance at least ϕ w.r.t. $\boldsymbol{d}_u$, and the number of edges between the V_is is at most $O(\sum_u \boldsymbol{d}_u \phi \log^3 n)$.*

Note that we explicitly introduce the $\boldsymbol{d}$ vector containing the degrees of the initial graph because we will repeatedly invoke this partition routine. This is due to our other half of the routine, which is to repeatedly project $\boldsymbol{g}$ among the remaining edges, and removing the ones that contribute to too much of its ℓ_2^2-norm in order to ensure uniformity as given in Case 2a of Theorem 4.9. To see that this process makes progress, we need the key observation from Lemma 2.15 that projections can only decrease the ℓ_2^2 norm of $\widehat{\boldsymbol{g}}$, the projection of the gradient.

Algorithm 6 Decomposition into Uniform Expanders

procedure DECOMPOSE($\mathcal{G}$, δ)
  Set $\phi \leftarrow c_{partition} \log^3 n \log(1/\delta)$ for some absolute constant $c_{partition}$.
  Iteratively remove all vertices with degree less than $\frac{m}{10n}$ to form $\mathcal{G}_{large}$.
  Compute $\overline{\boldsymbol{d}}$, the degrees of $\mathcal{G}_{large}$
  Return RECURSIVEDECOMPOSE($\mathcal{G}_{large}, \phi, 1, log(n/\delta)$).

Algorithm 7 Recursive Helper for Decomposition

Compute the projection of $\boldsymbol{g}^{\mathcal{G}}$ into its cycle space, $\widehat{\boldsymbol{g}}^{\mathcal{G}}$.
procedure DECOMPOSERECURSIVE($\mathcal{G}$, $\overline{\boldsymbol{d}}$, ϕ, i, L)
  Form $\mathcal{G}_{trimmed}$ by removing all edges $e \in E^{\mathcal{G}}$ such that $(\widehat{\boldsymbol{g}}_e^{\mathcal{G}})^2 \geq \frac{10L}{m^{\mathcal{G}}} \cdot \left\|\widehat{\boldsymbol{g}}^{\mathcal{G}}\right\|_2^2$.
  $(G_1, G_2, \ldots, G_t) \leftarrow$ EXPANDERDECOMPOSE($(V^{\mathcal{G}_{trimmed}}, E^{\mathcal{G}_{trimmed}}), \overline{\boldsymbol{d}}_{V^{\mathcal{G}_{trimmed}}}, \phi$).
  Initialize collection of results, $\mathcal{P}^{\mathcal{G}} \leftarrow \emptyset$.
  for $i = 1 \ldots t$ **do**
    Form $\mathcal{G}_i$ from the edges in $\mathcal{G}$ corresponding to G_i
    Compute $\widehat{\boldsymbol{g}}^{\mathcal{G}_i}$, the projection of $\boldsymbol{g}(\mathcal{G}_i)$ onto its cycle space.
    if $i = L$ or $(\|\widehat{\boldsymbol{g}}^{\mathcal{G}_i}\|_2^2 \geq \frac{1}{2}\|\widehat{\boldsymbol{g}}^{\mathcal{G}}\|_2^2$ and $m^{\mathcal{G}_i} \geq m^{\mathcal{G}}/2)$ **then**
      Add $\mathcal{G}_i$ to the results, $\mathcal{P}^{\mathcal{G}} \leftarrow \mathcal{P}^{\mathcal{G}} + \mathcal{G}_i$.
    else
      Recurse on $\mathcal{P}^{\mathcal{G}}$: $\mathcal{P}^{\mathcal{G}} \leftarrow \mathcal{P}^{\mathcal{G}} +$ DECOMPOSERECURSIVE($\mathcal{G}_i, \overline{\boldsymbol{d}}, \phi, i+1, L$).
  Return $\mathcal{P}^{\mathcal{G}}$.

This leads to an approach where we alternate between dropping the edges with high energy, and repartitioning the remaining edges into expanders. Pseudocode of this routine is in Algorithm 6, which calls a recursive routine, DECOMPOSERECURSIVE shown in Algorithm 7 with a suitable value of ϕ and number of layers. Note that we also need to trim the initial graph so that we only work with large degree vertices.

We will also need the following result (Lemma 28 of [KLOS14], see also [KM11]).

Lemma 5.2. *Suppose G is a unit weight graph with conductance ϕ. Then the projection operations into cycle and potential flow spaces both have ℓ_∞ norms bounded by $O(\phi^{-2}\log n)$:*

$$\left\|\boldsymbol{B}^{\mathcal{G}}\left(\boldsymbol{B}^{\mathcal{G}\top}\boldsymbol{B}^{\mathcal{G}}\right)^{\dagger}\boldsymbol{B}^{\mathcal{G}\top}\right\|_{\infty} \leq O(\phi^{-2}\log n)$$

and

$$\left\|I - \boldsymbol{B}^{\mathcal{G}}\left(\boldsymbol{B}^{\mathcal{G}\top}\boldsymbol{B}^{\mathcal{G}}\right)^{\dagger}\boldsymbol{B}^{\mathcal{G}\top}\right\|_{\infty} \leq O(\phi^{-2}\log n).$$

Proof. (of Theorem 4.9)

We start by bounding the qualities of the $\mathcal{G}$ pieces returned. As we only return pieces that are the outputs of EXPANDERDECOMPOSE, all of them have conductance at least ϕ by Lemma 5.1. Also, since we only keep the non-trivial pieces containing edges, taking the singleton cuts gives that the degrees in these pieces are at least

$$\phi \cdot \boldsymbol{d}_u \geq \frac{m}{2n} \cdot \frac{1}{c_{partition}\log^3 n \cdot \log(n/\delta)} = \frac{m}{10n\log^3 n\log(n/\delta)}.$$

Now consider the quality of each $\widehat{\boldsymbol{g}}^{\mathcal{G}_i}$: if it was returned due to $i = L$, then the energy of the projected gradient must have been halved at least $L - \log n$ times, or by a factor of $2^{L-\log n} = 2^{\log(1/\delta)} = 1/\delta$. Thus we would have

$$\left\|\widehat{\boldsymbol{g}}^{\mathcal{G}_i}\right\|_2^2 \leq \delta \left\|\widehat{\boldsymbol{g}}^{\mathcal{G}}\right\|_2^2 \leq \delta \left\|\boldsymbol{g}^{\mathcal{G}}\right\|_2^2.$$

Otherwise, we must have terminated because both the energy and edge count did not decrease too much. An edge e was kept in the trimmed set only if

$$\left(\widehat{\boldsymbol{g}}_e^{\mathcal{G}}\right)^2 \leq \frac{10L}{m^{\mathcal{G}}} \left\|\widehat{\boldsymbol{g}}^{\mathcal{G}}\right\|_2^2 \leq \frac{20L}{m^{\mathcal{G}_i}} \left\|\widehat{\boldsymbol{g}}^{\mathcal{G}}\right\|_2^2.$$

Combining this with the termination requirement of $\left\|\widehat{\boldsymbol{g}}^{\mathcal{G}_i}\right\|_2^2 \geq \frac{1}{2}\left\|\widehat{\boldsymbol{g}}^{\mathcal{G}}\right\|_2^2$ gives that the ℓ_∞ norm of the pre-projection gradient on $\mathcal{G}_i$, $\widehat{\boldsymbol{g}}^{\mathcal{G}}_{E^{\mathcal{G}_i}}$ satisfies

$$\left\|\widehat{\boldsymbol{g}}^{\mathcal{G}}_{E^{\mathcal{G}_i}}\right\|_\infty^2 \leq \frac{40L}{m^{\mathcal{G}_i}} \left\|\widehat{\boldsymbol{g}}^{\mathcal{G}_i}\right\|_2^2.$$

On the other hand, because $\mathcal{G}_i$ has expansion ϕ, doing an orthogonal cycle projection on it can only increase the ℓ_∞-norm of a vector by a factor of $O(\phi^{-2}\log n)$ by Lemma 5.2. Thus we have

$$\begin{aligned}\left\|\widehat{\boldsymbol{g}}^{\mathcal{G}_i}\right\|_\infty^2 &= \left\|\left(\boldsymbol{I} - \boldsymbol{B}^{\mathcal{G}}\left(\boldsymbol{B}^{\mathcal{G}\top}\boldsymbol{B}^{\mathcal{G}}\right)^{\dagger}\boldsymbol{B}^{\mathcal{G}\top}\right)\widehat{\boldsymbol{g}}^{\mathcal{G}}_{E^{\mathcal{G}_i}}\right\|_\infty^2 \leq \left\|\boldsymbol{I} - \boldsymbol{B}^{\mathcal{G}}\left(\boldsymbol{B}^{\mathcal{G}\top}\boldsymbol{B}^{\mathcal{G}}\right)^{\dagger}\boldsymbol{B}^{\mathcal{G}\top}\right\|_\infty^2 \left\|\widehat{\boldsymbol{g}}^{\mathcal{G}}_{E^{\mathcal{G}_i}}\right\|_\infty^2 \\ &\leq O\left(\phi^{-4}\log^2 n\right) \cdot \left\|\widehat{\boldsymbol{g}}^{\mathcal{G}}_{E^{\mathcal{G}_i}}\right\|_\infty^2 \leq O\left(\phi^{-4}\log^2 n\right) \cdot O\left(\frac{L}{m^{\mathcal{G}_i}}\right)\left\|\widehat{\boldsymbol{g}}^{\mathcal{G}_i}\right\|_2^2 = \frac{O\left(\log^{14} n \log^5(n/\delta)\right)}{m^{\mathcal{G}_i}}\left\|\widehat{\boldsymbol{g}}^{\mathcal{G}_i}\right\|_2^2,\end{aligned}$$

which is the desired (post-projection) uniformity bound.

We now bound the number of edges removed during all the recursive calls. The bound on L means this recursion has at most $O(\log(n/\delta))$ levels. Lemma 5.1 gives that the number of edges between the expander pieces is

$$O\left(\sum_u \boldsymbol{d}_u \phi \log^3 n\right) \cdot \log(n/\delta) = O\left(m\phi\log^3 n \log(n/\delta)\right),$$

so the setting of $\phi = \frac{1}{c_{partition}\log^3 n \log(n/\delta))}$ gives at most $m/10$ edges between the pieces for an appropriate choice of $c_{partition}$.

Furthermore, as each edge's contribution to $\widehat{\boldsymbol{g}}$ is non-negative, the number of edges whose relative contribution exceed $\frac{10L}{m}$ is at most $\frac{m}{10L}$. Summing this over all levels gives at most $m/10$ edges removed from the trimming step on Line 3 of DecomposeRecursive in Algorithm 7.

Finally, the running time is dominated by the expander decomposition calls. As there are $O(\log(n/\delta))$ levels of recursion and each level deals with edge-disjoint subsets, we obtain the total running time by substituting the value of ϕ into the runtime of expander decompositions as given in Lemma 5.1. □

Acknowledgements

This project would not have been possible without Dan Spielman's optimism about the existence of analogs of numerical methods for ℓ_p-norms, which he has expressed to us on multiple occasions over the past six years. We also thank Ainesh Bakshi, Jelani Nelson, Aaron Schild, and Junxing Wang for comments and suggestions on earlier drafts and presentations of these ideas.

As with many recent works in optimization algorithms on graphs, this project has its large share of influence by the late Michael B. Cohen. In fact, Michael's first papers on recursive preconditioning [CKM+14] and ℓ_p-norm preserving sampling of matrices [CP15] directly influenced the constructions of preconditioners (Section 4.4) and uniform expanders (Section 5 and Appendix D) respectively. While our overall algorithm falls short of what Michael would consider 'snazzy', it's also striking how many aspects of it he predicted, including: the use of expander decompositions; the $p \to \infty$ case being different than the $p \to 1$ case; and the large initial dependence on p that's also eventually fixable (see Section 1.4).

Richard regrets not being able to convince Michael to systematically investigate preconditioning for ℓ_p-norm flows. He is deeply grateful to Aleksander Mądry, Jon Kelner, Ian Munro, Tom Cohen, Marie Cohen, Sebastian Bubeck, and Ilya Razenshteyn for many helpful conversations following Michael's passing.

References

[AKPS19] 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, SODA 2019*, 2019.

[AL11] Morteza Alamgir and Ulrike V Luxburg. Phase transition in the family of p-resistances. In *Advances in Neural Information Processing Systems*, pages 379–387, 2011.

[ALdOW17] Zeyuan Allen-Zhu, Yuanzhi Li, Rafael Mendes de Oliveira, and Avi Wigderson. Much faster algorithms for matrix scaling. In *Symposium on Foundations of Computer Science (FOCS)*, pages 890–901, 2017. Available at: https://arxiv.org/abs/1704.02315.

[AN12] Ittai Abraham and Ofer Neiman. Using petal-decompositions to build a low stretch spanning tree. In *STOC*, 2012.

[BCLL18] 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.

[BK96] 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.

[BKKL17] Ruben Becker, Andreas Karrenbauer, Sebastian Krinninger, and Christoph Lenzen. Near-optimal approximate shortest paths and transshipment in distributed and streaming models. In *31st International Symposium on Distributed Computing, DISC 2017, October 16-20, 2017, Vienna, Austria*, pages 7:1–7:16, 2017. Available at: https://arxiv.org/abs/1607.05127.

[CKM+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. Available at http://arxiv.org/abs/1010.2921.

[CKM+14] Michael B. Cohen, Rasmus Kyng, Gary L. Miller, Jakub W. Pachocki, Richard Peng, Anup Rao, and Shen Chen Xu. Solving SDD linear systems in nearly $m \log^{1/2} n$ time. In *STOC*, pages 343–352, 2014.

[CMMP13] 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. Available at http://arxiv.org/abs/1110.1358.

[CMTV17] Michael B. Cohen, Aleksander Madry, Dimitris Tsipras, and Adrian Vladu. Matrix scaling and balancing via box constrained newton's method and interior point methods. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 902–913, 2017. Available at: https://arxiv.org/abs/1704.02310.

[CP15] Michael B. Cohen and Richard Peng. ℓ_p row sampling by Lewis weights. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, STOC '15, pages 183–192, New York, NY, USA, 2015. ACM. Available at http://arxiv.org/abs/1412.0588.

[DS08] 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 http://arxiv.org/abs/0803.0988.

[EACR+16] Ahmed El Alaoui, Xiang Cheng, Aaditya Ramdas, Martin J Wainwright, and Michael I Jordan. Asymptotic behavior of ℓ_p-based Laplacian regularization in semi-supervised learning. In *Conference on Learning Theory*, pages 879–906, 2016.

[Edm65] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of mathematics*, 17(3):449–467, 1965. Available at: https://cms.math.ca/10.4153/CJM-1965-045-4.

[EK72] Jack Edmonds and Richard M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM*, 19(2):248–264, 1972.

[ET75] Shimon Even and Robert Endre Tarjan. Network flow and testing graph connectivity. *SIAM J. Comput.*, 4(4):507–518, 1975.

[GKK+15] Mohsen Ghaffari, Andreas Karrenbauer, Fabian Kuhn, Christoph Lenzen, and Boaz Patt-Shamir. Near-optimal distributed maximum flow: Extended abstract. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, PODC 2015, Donostia-San Sebastián, Spain, July 21 - 23, 2015*, pages 81–90, 2015. Available at: https://arxiv.org/abs/1508.04747.

[GN79] Zvi Galil and Amnon Naamad. Network flow and generalized path compression. In *Proceedings of the 11h Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1979, Atlanta, Georgia, USA*, pages 13–26, 1979.

[GR98] Andrew V. Goldberg and Satish Rao. Beyond the flow decomposition barrier. *J. ACM*, 45(5):783–797, 1998.

[GT88] Andrew V. Goldberg and Robert Endre Tarjan. A new approach to the maximum-flow problem. *J. ACM*, 35(4):921–940, 1988.

[GT14] Andrew V. Goldberg and Robert Endre Tarjan. Efficient maximum flow algorithms. *Commun. ACM*, 57(8):82–89, 2014.

[HK73] John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, 2(4):225–231, 1973.

[HO13] Dorit S. Hochbaum and James B. Orlin. Simplifications and speedups of the pseudoflow algorithm. *Networks*, 61(1):40–57, 2013.

[Hoc08] Dorit S Hochbaum. The pseudoflow algorithm: A new algorithm for the maximum-flow problem. *Operations research*, 56(4):992–1009, 2008.

[Kar73] Alexander V. Karzanov. O nakhozhdenii maksimalńogo potoka v setyakh spetsialńogo vida i nekotorykh prilozheniyakh. *Matematicheskie Voprosy Upravleniya Proizvodstvom*, 5:81–94, 1973. In Russian, title translation: on finding maximum flows in networks with special structure and some applications.

[KBR07] Vladimir Kolmogorov, Yuri Boykov, and Carsten Rother. Applications of parametric maxflow in computer vision. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[KLOS14] 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. Available at http://arxiv.org/abs/1304.2338.

[KM09] J.A. Kelner and A. Madry. Faster generation of random spanning trees. In *FOCS*, 2009.

[KM11] Jonathan A. Kelner and Petar Maymounkov. Electric routing and concurrent flow cutting. *Theor. Comput. Sci.*, 412(32):4123–4135, 2011. Available at: https://arxiv.org/abs/0909.2859.

[KMP11] 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. Available at http://arxiv.org/abs/1102.4842.

[KMP12] Ioannis Koutis, Gary L. Miller, and Richard Peng. A fast solver for a class of linear systems. *Communications of the ACM*, 55(10):99–107, October 2012. Available at https://cacm.acm.org/magazines/2012/10/155538-a-fast-solver-for-a-class-of-linear-systems/fulltext.

[KMP14] I. Koutis, G. Miller, and R. Peng. Approaching optimality for solving sdd linear systems. *SIAM Journal on Computing*, 43(1):337–354, 2014. Available at http://arxiv.org/abs/1003.2958.

[KOSZ13] J. A. Kelner, L. Orecchia, A. Sidford, and Z. A. Zhu. A simple, combinatorial algorithm for solving sdd systems in nearly-linear time. In *STOC*, 2013.

[KRSS15] R. Kyng, A. B. Rao, S. Sachdeva, and D. A Spielman. Algorithms for lipschitz learning on graphs. In *COLT*, 2015.

[KS96] David R. Karger and Clifford Stein. A new approach to the minimum cut problem. *J. ACM*, 43(4):601–640, 1996.

[KS16] Rasmus Kyng and Sushant Sachdeva. Approximate gaussian elimination for laplacians - fast, sparse, and simple. In *FOCS*, pages 573–582. IEEE Computer Society, 2016. Available at http://arxiv.org/abs/1605.02353.

[LPS15] Y. T. Lee, R. Peng, and D. A. Spielman. Sparsified cholesky solvers for SDD linear systems. *CoRR*, abs/1506.08204, 2015.

[LS13] Yin Tat Lee and Aaron Sidford. Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems. In *Proceedings of the 2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, FOCS '13, pages 147–156, Washington, DC, USA, 2013. IEEE Computer Society.

[LS14] 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.

[LSBG13] Bingdong Li, Jeff Springer, George Bebis, and Mehmet Hadi Gunes. A survey of network flow applications. *Journal of Network and Computer Applications*, 36(2):567–581, 2013.

[Mad10] 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. Available at http://arxiv.org/abs/1008.1975.

[Mad11] Aleksander Madry. *From graphs to matrices, and back: new techniques for graph algorithms*. PhD thesis, Massachusetts Institute of Technology, 2011.

[Mad13] A. Madry. Navigating central path with electrical flows: From flows to matchings, and back. In *FOCS*, 2013.

[Mad16] Aleksander Madry. Computing maximum flow with augmenting electrical flows. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 593–602, 2016. Available at: https://arxiv.org/abs/1608.06016.

[NN94] Y. Nesterov and A. Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming.* Society for Industrial and Applied Mathematics, 1994.

[Orl13] James 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*, pages 765–774, 2013.

[Pen16] Richard Peng. Approximate undirected maximum flows in $O(m \text{ polylog}(n))$ time. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1862–1867. SIAM, 2016. Available at http://arxiv.org/abs/1411.7631.

[PS14] 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. Available at http://arxiv.org/abs/1311.3286.

[PZZ13] Bo Peng, Lei Zhang, and David Zhang. A survey of graph theoretical approaches to image segmentation. *Pattern Recognition*, 46(3):1020–1038, 2013.

[ROF92] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268, 1992.

[Sac19] Sushant Sachdeva. Private Communication, 2019.

[Sch02] Alexander Schrijver. On the history of the transportation and maximum flow problems. *Math. Program.*, 91(3):437–445, 2002.

[She13] 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. Available at http://arxiv.org/abs/1304.2077.

[She17a] Jonah Sherman. Area-convexity, l_∞ regularization, and undirected multicommodity flow. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 452–460, 2017.

[She17b] 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. Available at: https://arxiv.org/abs/1606.07425.

[Spi18] D. A. Spielman. Conductance, the Normalized Laplacian, and Cheegers Inequality. http://www.cs.yale.edu/homes/spielman/561/lect11-18.pdf, 2018.

[SS11] D. Spielman and N. Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011. Available at http://arxiv.org/abs/0803.0929.

[ST83] Daniel D Sleator and Robert Endre Tarjan. A data structure for dynamic trees. *Journal of computer and system sciences*, 26(3):362–391, 1983. Announced at STOC'81.

[ST85] Daniel Dominic Sleator and Robert Endre Tarjan. Self-adjusting binary search trees. *J. ACM*, 32(3):652–686, 1985.

[ST14] D. Spielman and S. Teng. Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. *SIAM Journal on Matrix Analysis and Applications*, 35(3):835–885, 2014. Available at http://arxiv.org/abs/cs/0607105.

[SW18] Thatchaphol Saranurak and Di Wang. Expander decomposition and pruning: Faster, stronger, and simpler, 2018. To appear at SODA 2019. https://dw236.github.io/papers/main_decomp.pdf.

[Tro12] Joel A. Tropp. User-friendly tail bounds for sums of random matrices. *Found. Comput. Math.*, 12(4):389–434, August 2012. Available at http://arxiv.org/abs/1004.4389.

[ZWC10] Mingqiang Zhu, Stephen J Wright, and Tony F Chan. Duality-based algorithms for total-variation-regularized image restoration. *Computational Optimization and Applications*, 47(3):377–400, 2010.

A Deferred Proofs from Prelims, Section 2.2

Lemma 2.9 (Reflexivity)**.** *For every smoothed p-norm instance $\mathcal{G}$, and every $\kappa \geq 1$, $\mathcal{G} \preceq_\kappa \mathcal{G}$ and $\mathcal{G} \preceq_\kappa^{cycle} \mathcal{G}$ with the identity map.*

Proof. Consider the map $\mathcal{M}_{\mathcal{G}\to\mathcal{G}}$ such that for every flow $\boldsymbol{f}^{\mathcal{G}}$ on $\mathcal{G}$, we have $\mathcal{M}_{\mathcal{G}\to\mathcal{G}}(\boldsymbol{f}^{\mathcal{G}}) = \boldsymbol{f}^{\mathcal{G}}$. Thus,

$$\begin{aligned}
\mathcal{E}^{\mathcal{G}}\left(\kappa^{-1}\mathcal{M}_{\mathcal{G}\to\mathcal{G}}(\boldsymbol{f}^{\mathcal{G}})\right) &= \mathcal{E}^{\mathcal{G}}\left(\kappa^{-1}\boldsymbol{f}^{\mathcal{G}}\right) \\
&= \left(\boldsymbol{g}^{\mathcal{G}}\right)^\top\left(\kappa^{-1}\boldsymbol{f}^{\mathcal{G}}\right) - h_p(\boldsymbol{r}, \kappa^{-1}\boldsymbol{f}^{\mathcal{G}}) \\
&\geq \kappa^{-1}\left(\boldsymbol{g}^{\mathcal{G}}\right)^\top\boldsymbol{f}^{\mathcal{G}} - \kappa^{-2}h_p(\boldsymbol{r}, \boldsymbol{f}^{\mathcal{G}}) && \text{(Using Lemma B.1)} \\
&\geq \kappa^{-1}\left(\boldsymbol{g}^{\mathcal{G}}\right)^\top\boldsymbol{f}^{\mathcal{G}} - \kappa^{-1}h_p(\boldsymbol{r}, \boldsymbol{f}^{\mathcal{G}}) = \kappa^{-1}\mathcal{E}^{\mathcal{G}}(\boldsymbol{f}^{\mathcal{G}}).
\end{aligned}$$

Moreover $(\boldsymbol{B}^{\mathcal{G}})^\top\mathcal{M}_{\mathcal{G}\to\mathcal{G}}(\boldsymbol{f}^{\mathcal{G}}) = \boldsymbol{B}^{\mathcal{G}}\boldsymbol{f}^{\mathcal{G}}$. Thus, the claims follow. □

Lemma 2.10 (Composition)**.** *Given two smoothed p-norm instances, $\mathcal{G}_1, \mathcal{G}_2$, such that $\mathcal{G}_1 \preceq_{\kappa_1} \mathcal{G}_2$ with the map $\mathcal{M}_{\mathcal{G}_1\to\mathcal{G}_2}$ and $\mathcal{G}_2 \preceq_{\kappa_2} \mathcal{G}_3$ with the map $\mathcal{M}_{\mathcal{G}_2\to\mathcal{G}_3}$, then $\mathcal{G}_1 \preceq_{\kappa_1\kappa_2} \mathcal{G}_3$ with the map $\mathcal{M}_{\mathcal{G}_1\to\mathcal{G}_3} = \mathcal{M}_{\mathcal{G}_2\to\mathcal{G}_3} \circ \mathcal{M}_{\mathcal{G}_1\to\mathcal{G}_2}$.*

Similarly, for any $\mathcal{G}_1, \mathcal{G}_2$, if $\mathcal{G}_1 \preceq_{\kappa_1}^{cycle} \mathcal{G}_2$ with the map $\mathcal{M}_{\mathcal{G}_1\to\mathcal{G}_2}$ and $\mathcal{G}_2 \preceq_{\kappa_2}^{cycle} \mathcal{G}_3$ with the map $\mathcal{M}_{\mathcal{G}_2\to\mathcal{G}_3}$, then $\mathcal{G}_1 \preceq_{\kappa_1\kappa_2}^{cycle} \mathcal{G}_3$ with the map $\mathcal{M}_{\mathcal{G}_1\to\mathcal{G}_3} = \mathcal{M}_{\mathcal{G}_2\to\mathcal{G}_3} \circ \mathcal{M}_{\mathcal{G}_1\to\mathcal{G}_2}$.

Proof. It is easy to observe that the given mapping is linear. Given a flow $\boldsymbol{f}^{\mathcal{G}_1}$ on $\mathcal{G}_1$, we have,

$$\begin{aligned}(\boldsymbol{B}^{\mathcal{G}_3})^\top\left(\mathcal{M}_{\mathcal{G}_2\to\mathcal{G}_3}\circ\mathcal{M}_{\mathcal{G}_1\to\mathcal{G}_2}(\boldsymbol{f}^{\mathcal{G}_1})\right) &= (\boldsymbol{B}^{\mathcal{G}_3})^\top\left(\mathcal{M}_{\mathcal{G}_2\to\mathcal{G}_1}\left(\mathcal{M}_{\mathcal{G}_3\to\mathcal{G}_2}(\boldsymbol{f}^{\mathcal{G}_1})\right)\right)\\ &= (\boldsymbol{B}^{\mathcal{G}_2})^\top\left(\mathcal{M}_{\mathcal{G}_3\to\mathcal{G}_2}(\boldsymbol{f}^{\mathcal{G}_1})\right) = (\boldsymbol{B}^{\mathcal{G}_1})^\top(\boldsymbol{f}^{\mathcal{G}_1}).\end{aligned}$$

Moreover,

$$\begin{aligned}\mathcal{E}_{\mathcal{G}_3}\left((\kappa_1\kappa_2)^{-1}\mathcal{M}_{\mathcal{G}_2\to\mathcal{G}_3}\left(\mathcal{M}_{\mathcal{G}_1\to\mathcal{G}_2}(\boldsymbol{f}^{\mathcal{G}_1})\right)\right) &\geq \mathcal{E}_{\mathcal{G}_3}\left(\kappa_2{}^{-1}\mathcal{M}_{\mathcal{G}_2\to\mathcal{G}_3}\left(\kappa_1{}^{-1}\mathcal{M}_{\mathcal{G}_1\to\mathcal{G}_2}(\boldsymbol{f}^{\mathcal{G}_1})\right)\right) && \text{(Using linearity)}\\ &\geq \kappa_2{}^{-1}\mathcal{E}_{\mathcal{G}_2}\left(\kappa_1{}^{-1}\mathcal{M}_{\mathcal{G}_1\to\mathcal{G}_2}(\boldsymbol{f}^{\mathcal{G}_1})\right) && \text{(Using } \mathcal{G}_2\preceq_{\kappa_2}\mathcal{G}_3\text{)}\\ &\geq (\kappa_2\kappa_1)^{-1}\mathcal{E}_{\mathcal{G}_1}\left(\boldsymbol{f}^{\mathcal{G}_1}\right) && \text{(Using } \mathcal{G}_1\preceq_{\kappa_1}\mathcal{G}_2\text{)}\end{aligned}$$

The same proof works for $\preceq^{\text{cycle}}$. ◻

Lemma 2.12 ($\preceq_\kappa$ under union)**.** *Consider four smoothed p-norm instances, $\mathcal{G}_1, \mathcal{G}_2, \mathcal{H}_1, \mathcal{H}_2$, on the same set of vertices,* i.e. $V^{\mathcal{G}_1} = V^{\mathcal{G}_2} = V^{\mathcal{H}_1} = V^{\mathcal{H}_2}$, *such that for $i = 1, 2$, $\mathcal{H}_i \preceq_\kappa \mathcal{G}_i$ with the map $\mathcal{M}_{\mathcal{H}_i\to\mathcal{G}_i}$. Let $\mathcal{G} \stackrel{\text{def}}{=} \mathcal{G}_1 \cup \mathcal{G}_2$, and $\mathcal{H} \stackrel{\text{def}}{=} \mathcal{H}_1 \cup \mathcal{H}_2$. Then, $\mathcal{H} \preceq_\kappa \mathcal{G}$ with the map*

$$\mathcal{M}_{\mathcal{H}\to\mathcal{G}}\left(\boldsymbol{f}^{\mathcal{H}} = (\boldsymbol{f}^{\mathcal{H}_1}, \boldsymbol{f}^{\mathcal{H}_2})\right) \stackrel{\text{def}}{=} \left(\mathcal{M}_{\mathcal{H}_1\to\mathcal{G}_1}\left(\boldsymbol{f}^{\mathcal{H}_1}\right), \mathcal{M}_{\mathcal{H}_2\to\mathcal{G}_2}\left(\boldsymbol{f}^{\mathcal{H}_2}\right)\right),$$

where $(\boldsymbol{f}^{H_1}, \boldsymbol{f}^{H_2})$ is the decomposition of $\boldsymbol{f}^H$ onto the supports of H_1 and H_2.

Proof. Let $\boldsymbol{f}^{\mathcal{H}}$ be a flow on $\mathcal{H}$. We write $\boldsymbol{f}^{\mathcal{H}} = (\boldsymbol{f}^{\mathcal{H}_1}, \boldsymbol{f}^{\mathcal{H}_2})$. Let $\boldsymbol{f}^{\mathcal{G}} \stackrel{\text{def}}{=} \mathcal{M}_{\mathcal{H}\to\mathcal{G}}(\boldsymbol{f}^{\mathcal{H}})$. If $\boldsymbol{f}^{\mathcal{G}_i}$ denotes $\mathcal{M}_{\mathcal{H}_i\to\mathcal{G}_i}(\boldsymbol{f}^{\mathcal{H}_i})$ for $i = 1, 2$, then we know that $\boldsymbol{f}^{\mathcal{G}} = (\boldsymbol{f}^{\mathcal{G}_1}, \boldsymbol{f}^{\mathcal{G}_2})$. Thus, the objectives satisfy

$$\begin{aligned}\mathcal{E}^{\mathcal{G}}(\kappa^{-1}\boldsymbol{f}^{\mathcal{G}}) &= \mathcal{E}^{\mathcal{G}_1}(\kappa^{-1}\boldsymbol{f}^{\mathcal{G}_1}) + \mathcal{E}^{\mathcal{G}_2}(\kappa^{-1}\boldsymbol{f}^{\mathcal{G}_2})\\ &\geq \kappa^{-1}\mathcal{E}^{\mathcal{H}_1}(\boldsymbol{f}^{\mathcal{H}_1}) + \kappa^{-1}\mathcal{E}^{\mathcal{H}_2}(\boldsymbol{f}^{\mathcal{H}_2}) = \kappa^{-1}\mathcal{E}^{\mathcal{H}}(\boldsymbol{f}^{\mathcal{H}})\end{aligned}$$

For the residues, we have,

$$\begin{aligned}(\boldsymbol{B}^{\mathcal{G}})^\top(\boldsymbol{f}^{\mathcal{G}}) &= (\boldsymbol{B}^{\mathcal{G}_1})^\top(\boldsymbol{f}^{\mathcal{G}_1}) + (\boldsymbol{B}^{\mathcal{G}_2})^\top(\boldsymbol{f}^{\mathcal{G}_2})\\ &= (\boldsymbol{B}^{\mathcal{H}_1})^\top(\boldsymbol{f}^{\mathcal{H}_1}) + (\boldsymbol{B}^{\mathcal{H}_2})^\top(\boldsymbol{f}^{\mathcal{H}_2}) = (\boldsymbol{B}^{\mathcal{H}})^\top(\boldsymbol{f}^{\mathcal{H}}).\end{aligned}$$

Thus, $\mathcal{H} \preceq_\kappa \mathcal{G}$. ◻

Lemma 2.13. *For all $\kappa \geq 1$, and for all pairs of smoothed p-norm instances, $\mathcal{G}, \mathcal{H}$, on the same underlying graphs,* i.e., $(V^{\mathcal{G}}, E^{\mathcal{G}}) = (V^{\mathcal{H}}, E^{\mathcal{H}})$, *such that,*

1. *the gradients are identical, $\boldsymbol{g}^{\mathcal{G}} = \boldsymbol{g}^{\mathcal{H}}$,*
2. *the ℓ_2^2 resistances are off by at most κ,* i.e., $\boldsymbol{r}_e^{\mathcal{G}} \leq \kappa\boldsymbol{r}_e^{\mathcal{H}}$ *for all edges e, and*
3. *the p-norm scaling is off by at most κ^{p-1},* i.e., $s^{\mathcal{G}} \leq \kappa^{p-1}s^{\mathcal{H}}$,

then $\mathcal{H} \preceq_\kappa \mathcal{G}$ with the identity map.

Proof. Consider the map $\mathcal{M}_{\mathcal{H}\to\mathcal{G}}(\boldsymbol{f}) = \boldsymbol{f}$. Thus, since the underlying graphs are the same, we immediately have $(\boldsymbol{B}^{\mathcal{G}})^\top\boldsymbol{f} = (\boldsymbol{B}^{\mathcal{H}})^\top\boldsymbol{f}$. For the objective, we have

$$\begin{aligned}\mathcal{E}^{\mathcal{G}}(\kappa^{-1}\boldsymbol{f}) &= \sum_e\left(\kappa^{-1}\boldsymbol{g}_e^{\mathcal{G}}\boldsymbol{f}_e - \kappa^{-2}\boldsymbol{r}_e^{\mathcal{G}}\boldsymbol{f}_e^2 - \kappa^{-p}s^{\mathcal{G}}|\boldsymbol{f}_e|^p\right)\\ &\geq \kappa^{-1}\sum_e\left(\boldsymbol{g}_e^{\mathcal{H}}\boldsymbol{f}_e - \boldsymbol{r}_e^{\mathcal{H}}\boldsymbol{f}_e^2 - s^{\mathcal{H}}|\boldsymbol{f}_e|^p\right) = \kappa^{-1}\mathcal{E}^{\mathcal{H}}(\boldsymbol{f}).\end{aligned}$$

◻

B Deferred Proofs for Numerical Methods from Section 3

The following simple lemma characterizes the change in smoothed ℓ_p-norms under rescaling of the input vector.

Lemma B.1. *For all $\boldsymbol{x} \in \mathbb{R}^m, \boldsymbol{r} \in \mathbb{R}^m_{\geq 0}, s \in \mathbb{R}_{\geq 0}$, and $\lambda \in \mathbb{R}$, we have,*

$$\min\{|\lambda|^2, |\lambda|^p\} h_p(\boldsymbol{r}, s, \boldsymbol{x}) \leq h_p(\boldsymbol{r}, s, \lambda \boldsymbol{x}) \leq \max\{|\lambda|^2, |\lambda|^p\} h_p(\boldsymbol{r}, s, \boldsymbol{x}).$$

Proof. It suffices to prove the claim for $x \in \mathbb{R}, r \in \mathbb{R}_{\geq 0}, s \in \mathbb{R}_{\geq 0}$

$$\begin{aligned} h_p(r, s, \lambda x) &= r(\lambda x)^2 + s|\lambda x|^p \\ &= |\lambda|^2 \cdot rx^2 + |\lambda|^p \cdot s|x|^p \end{aligned}$$

Since all terms are non-negative, we get,

$$\begin{aligned} &h_p(r, s, \lambda x) \geq \min\{|\lambda|^2, |\lambda|^p\} \cdot (rx^2 + s|x|^p), \\ \text{and} \quad &h_p(r, s, \lambda x) \leq \max\{|\lambda|^2, |\lambda|^p\} \cdot (rx^2 + s|x|^p). \end{aligned}$$

□

Lemma 3.1 ([AKPS19])**.** *For all $\boldsymbol{r}, \boldsymbol{x}, \boldsymbol{\delta} \in \mathbb{R}^m$, with $\boldsymbol{r} \in \mathbb{R}^m_{\geq 0}$, and $s \geq 0$, we have*

$$2^{-p} \cdot h_p(\boldsymbol{r} + |\boldsymbol{x}|^{p-2}, s, \boldsymbol{\delta}) \leq h_p(\boldsymbol{r}, s, \boldsymbol{x} + \boldsymbol{\delta}) - h_p(\boldsymbol{r}, s, \boldsymbol{x}) - \boldsymbol{\delta}^\top \nabla_{\boldsymbol{x}} h_p(\boldsymbol{r}, s, \boldsymbol{x}) \leq 2^{2p} \cdot h_p(\boldsymbol{r} + |\boldsymbol{x}|^{p-2}, s, \boldsymbol{\delta}).$$

Proof. Note that all the terms are a sum over the coordinates. Thus, it suffices to prove the inequality for $x, \delta \in \mathbb{R}$, and $r, s \in \mathbb{R}_{\geq 0}$. We have,

$$\begin{aligned} h_p(r, s, x + \delta) - h_p(r, s, x) - \delta \frac{\partial}{\partial x} h_p(r, s, x) &= r(x + \delta)^2 + s|x + \delta|^p - rx^2 - s|x|^p - \delta(2rx + ps|x|^{p-2}x) \\ &= r\delta^2 + s|x + \delta|^p - s|x|^p - ps\delta|x|^{p-2}x \\ &= r\delta^2 + s|x|^p\left(\left|1 + \delta'\right|^p - 1 - p\delta'\right), \end{aligned}$$

where $\delta' = \delta/x$.

Lemma B.2, proved later, proves that for all δ', and $p \geq 2$, we have,

$$\left|1 + \delta'\right|^p - 1 - p\delta' \leq p2^{p-1}\left(\delta'^2 + |\delta'|^p\right).$$

Thus,

$$\begin{aligned} h_p(r, s, x + \delta) - h_p(r, s, x) - \delta \frac{\partial}{\partial x} h_p(r, s, x) &\leq r\delta^2 + s|x|^p p2^{p-1}\left(\delta'^2 + |\delta'|^p\right) \\ &= r\delta^2 + sp2^{p-1}|x|^{p-2}\delta^2 + sp2^{p-1}|\delta|^p \\ &\leq p2^{p-1}((r + s|x|^{p-2})\delta^2 + s|\delta|^p) \\ &= p2^{p-1} h_p(r + s|x|^{p-2}, s, \delta) \\ &\leq 2^{2p} h_p(r + s|x|^{p-2}, s, \delta). \end{aligned}$$

Lemma B.3, proved later, shows that for all δ', and $p \geq 2$, we have,

$$\left|1+\delta'\right|^p - 1 - p\delta' \geq 2^{-p}\big(\delta'^2 + |\delta'|^p\big).$$

$$\begin{aligned}
h_p(r,s,x+\delta) - h_p(r,s,x) - \delta\frac{\partial}{\partial x}h_p(r,s,x) &\geq r\delta^2 + s|x|^p 2^{-p}\big(\delta'^2 + |\delta'|^p\big) \\
&= r\delta^2 + 2^{-p}s|x|^{p-2}\delta^2 + 2^{-p}s|\delta|^p \\
&\geq 2^{-p}((r + s|x|^{p-2})\delta^2 + s|\delta|^p) \\
&= 2^{-p}h_p(r + s|x|^{p-2}, s, \delta).
\end{aligned}$$

□

Lemma B.2. *For all $\delta \in \mathbb{R}$, $p \geq 1$, we have,*

$$|1+\delta|^p - 1 - p\delta \leq p2^{p-1}\big(\delta^2 + |\delta|^p\big).$$

Proof. The proof has to consider several cases.

$\boldsymbol{\delta \geq 1}$. Using mean-value theorem, we know there is some $z \in [0,\delta]$ such that

$$\begin{aligned}
|1+\delta|^p - 1 - p\delta &= (1+\delta)^p - 1 - p\delta \\
&= p\delta\big((1+z)^{p-1} - 1\big) \\
&\leq p\delta(1+\delta)^{p-1} \\
&\leq p\delta(2\delta)^{p-1}.
\end{aligned}$$

$\boldsymbol{0 \leq \delta \leq 1}$. Using mean-value theorem, we know there is some $z \in [0,\delta]$ such that

$$\begin{aligned}
|1+\delta|^p - 1 - p\delta &= (1+\delta)^p - 1 - p\delta \\
&= p\delta\big((1+z)^{p-1} - 1\big) \\
&\leq p\delta\big((1+\delta)^{p-1} - 1\big).
\end{aligned}$$

If $p \leq 2$, we have $(1+\delta)^{p-1}$ is a concave function, and hence $(1+\delta)^{p-1} \leq 1 + (p-1)\delta$. If $p \geq 2$, we have $(1+\delta)^{p-1}$ is a convex function, and hence for $\delta \in [0,1]$, we have $(1+\delta)^{p-1} \leq 1 + (2^{p-1}-1)\delta$. Thus,

$$|1+\delta|^p - 1 - p\delta \leq p\max\{(p-1)\delta^2, (2^{p-1}-1)\delta^2\}.$$

$\boldsymbol{-1 \leq \delta \leq 0}$. Using mean-value theorem, we know there is some $z \in [-|\delta|, 0]$ such that

$$\begin{aligned}
|1+\delta|^p - 1 - p\delta &= (1+\delta)^p - 1 - p\delta \\
&= p\delta\big((1+z)^{p-1} - 1\big) \\
&\leq p|\delta|\big(1 - (1+\delta)^{p-1}\big).
\end{aligned}$$

If $p \leq 2$, we have $(1+\delta)^{p-1}$ is a concave function, and hence for $\delta \in [-1,1]$, we have $(1+\delta)^{p-1} \geq 1+\delta$. If $p \geq 2$, we have $(1+\delta)^{p-1}$ is a convex function, and hence $(1+\delta)^{p-1} \geq 1 + (p-1)\delta$. Thus,

$$|1+\delta|^p - 1 - p\delta \leq p|\delta|\max\{|\delta|, (p-1)|\delta|\}.$$

$\boldsymbol{\delta \leq -1}$. We have,

$$\begin{aligned} |1+\delta|^p - 1 - p\delta &= (|\delta|-1)^p - 1 + p|\delta| \\ &\leq |\delta|^p + p|\delta| \\ &\leq |\delta|^p + p|\delta|^2, \end{aligned}$$

since $|\delta| \geq 1$. ☐

Lemma B.3. *For all $\delta \in \mathbb{R}, p \geq 2$, we have*

$$|1+\delta|^p - 1 - p\delta \geq 2^{-p}(\delta^2 + |\delta|^p).$$

Proof. Let $h(\delta)$ denote the function

$$h(\delta) = |1+\delta|^p - 1 - p\delta - 2^{-p}(\delta^2 + |\delta|^p).$$

Thus, $h(0) = 0$. As for the previous proof, we consider several cases:

$\boldsymbol{\delta \geq 0}$. We have $h(\delta) = (1+\delta)^p - 1 - p\delta - 2^{-p}(\delta^2 + \delta^p)$. Thus,

$$\begin{aligned} h'(\delta) &= p(1+\delta)^{p-1} - p - 2^{-p+1}\delta - p2^{-p}\delta^{p-1} \\ h''(\delta) &= p(p-1)(1+\delta)^{p-2} - 2^{-p+1} - p(p-1)2^{-p}\delta^{p-2} \end{aligned}$$

Observe that since $p \geq 2$, we have $(1+\delta)^{p-2} \geq \max\{1, \delta^{p-2}\} \geq 2^{-1}(1+\delta^{p-2})$, and $p(p-1) \geq 2$. Thus,

$$h''(\delta) \geq 2^{-1}p(p-1) + 2^{-1}p(p-1)\delta^{p-2} - 2^{-p+1} - p(p-1)2^{-p}\delta^{p-2} \geq 0.$$

Since $h(0) = h'(0) = 0$, and $h''(\delta) \geq 0$, for all $\delta \geq 0$, we must have $h(\delta) \geq 0$ for all $\delta \geq 0$.

$\boldsymbol{-1 \leq \delta \leq 0}$. We have,

$$\begin{aligned} h(\delta) &= (1+\delta)^p - 1 - p\delta - 2^{-p}\delta^2 - 2^{-p}|\delta|^p \\ h'(\delta) &= p(1+\delta)^{p-1} - p - 2^{-p+1}\delta + p2^{-p}|\delta|^{p-1}. \end{aligned}$$

Since $0 \leq 1+\delta \leq 1$, and $p-1 \geq 1$, we have $(1+\delta)^{p-1} \leq 1+\delta$, and $|\delta|^{p-1} \leq |\delta|^1 = -\delta$. Thus,

$$\begin{aligned} h'(\delta) &\leq p(1+\delta) - p - 2^{-p+1}\delta + p2^{-p}|\delta| \\ &= -p|\delta| + (2+p)2^{-p}|\delta| \\ &\leq -p|\delta| + 2^{-2}(p+2)|\delta| \leq 0. \end{aligned}$$

$\boldsymbol{\delta \leq -1}$. We have,

$$\begin{aligned} h(\delta) &= (-1-\delta)^p - 1 - p\delta - 2^{-p}\delta^2 - 2^{-p}|\delta|^p \\ h'(\delta) &= -p(-1-\delta)^{p-1} - p - 2^{-p+1}\delta + p2^{-p}|\delta|^{p-1}. \end{aligned}$$

Since $|\delta| \geq 1$, and $p-1 \geq 1$, we have $-\delta = |\delta| \leq |\delta|^{p-1}$. Thus,

$$\begin{aligned} h'(\delta) &\leq -p(-1-\delta)^{p-1} - p + 2^{-p}(2+p)|\delta|^{p-1} \\ &\leq -p\Big((-1-\delta)^{p-1} + 1 - 2^{-p+1}|\delta|^{p-1}\Big). \end{aligned}$$

Now, observe that since $0 \leq -1-\delta$, and $p-1 \geq 1$,

$$|\delta|^{p-1} = (-\delta)^{p-1} = (1 + (-1-\delta))^{p-1} \leq 2^{p-1}(1^{p-1} + (-1-\delta)^{p-1}),$$

Thus,

$$(-1-\delta)^{p-1} + 1 - 2^{-p+1}|\delta|^{p-1} \geq 0,$$

and hence $h'(\delta) \leq 0$ for $\delta \leq -1$.

For the last two cases, since $h(0) = 0$, and $h'(\delta) \leq 0$, for all $\delta \leq 0$. Thus, we must have $h(\delta) \geq 0$, for $\delta \leq 0$. ☐

Theorem 3.2 ([AKPS19])**.** *Given the following optimization problem,*

$$\begin{array}{rl} \max_{\boldsymbol{x}} & \mathcal{E}_1(\boldsymbol{x}) \stackrel{\text{def}}{=} \boldsymbol{g}^\top \boldsymbol{x} - h_p(\boldsymbol{r}, s, \boldsymbol{x}) \\ s.t. & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \end{array} \tag{P1}$$

and an initial feasible solution $\boldsymbol{x}_0$, we can construct the following residual problem:

$$\begin{array}{rl} \max_{\boldsymbol{\delta}} & \mathcal{E}_2(\boldsymbol{\delta}) \stackrel{\text{def}}{=} (\boldsymbol{g}')^\top \boldsymbol{\delta} - h_p(\boldsymbol{r}', s, \boldsymbol{\delta}) \\ s.t. & \boldsymbol{A}\boldsymbol{\delta} = \boldsymbol{0}, \end{array} \tag{R1}$$

where $\boldsymbol{g}' = 2^p(\boldsymbol{g} - \nabla_{\boldsymbol{x}} h(\boldsymbol{r}, s, \boldsymbol{x})|_{\boldsymbol{x}=\boldsymbol{x}_0})$, and $\boldsymbol{r}' = \boldsymbol{r} + s|\boldsymbol{x}_0|^{p-2}$.

There exists a feasible solution $\widetilde{\boldsymbol{\delta}}$ to the residual problem R1 that achieves an objective of $\mathcal{E}_2\left(\widetilde{\boldsymbol{\delta}}\right) \geq 2^p(\mathcal{E}_1(\boldsymbol{x}^\star) - \mathcal{E}_1(\boldsymbol{x}_0))$, where $\boldsymbol{x}^\star$ is an optimal solution to problem P1.

Moreover, given any feasible solution $\boldsymbol{\delta}$ to Program R1, the vector $\boldsymbol{x}_1 \stackrel{\text{def}}{=} \boldsymbol{x}_0 + 2^{-3p}\boldsymbol{\delta}$ is a feasible solution to the Program P1 and obtains the objective

$$\mathcal{E}_1(\boldsymbol{x}_1) \geq \mathcal{E}_1(\boldsymbol{x}_0) + 2^{-4p}\mathcal{E}_2(\boldsymbol{\delta}).$$

Proof. Let $\boldsymbol{x}^\star$ to be an optimal solution to Problem P1. Consider $\widetilde{\boldsymbol{\delta}} = \boldsymbol{x}^\star - \boldsymbol{x}_0$. Thus,

$$\boldsymbol{A}\widetilde{\boldsymbol{\delta}} = \boldsymbol{A}\boldsymbol{x}^\star - \boldsymbol{A}\boldsymbol{x}_0 = \boldsymbol{b} - \boldsymbol{b} = 0.$$

Thus, $\widetilde{\boldsymbol{\delta}}$ is a feasible solution to Problem R1. Moreover, it satisfies,

$$\begin{aligned} \mathcal{E}_2(\widetilde{\boldsymbol{\delta}}) &= (\widetilde{\boldsymbol{\delta}})^\top 2^p(\boldsymbol{g} - \nabla_{\boldsymbol{x}} h(\boldsymbol{r}, s, \boldsymbol{x})|_{\boldsymbol{x}=\boldsymbol{x}_0}) - h_p(\boldsymbol{r} + s|\boldsymbol{x}_0|^{p-2}, s, \widetilde{\boldsymbol{\delta}}) \\ &= 2^p \boldsymbol{g}^\top \widetilde{\boldsymbol{\delta}} - 2^p\Big(2^{-p} h_p(\boldsymbol{r} + s|\boldsymbol{x}_0|^{p-2}, s, \widetilde{\boldsymbol{\delta}}) + (\widetilde{\boldsymbol{\delta}})^\top \nabla_{\boldsymbol{x}} h(\boldsymbol{r}, s, \boldsymbol{x})|_{\boldsymbol{x}=\boldsymbol{x}_0}\Big) \\ &\geq 2^p \boldsymbol{g}^\top \widetilde{\boldsymbol{\delta}} - 2^p\Big(h_p(\boldsymbol{r}, s, \boldsymbol{x}_0 + \widetilde{\boldsymbol{\delta}}) - h_p(\boldsymbol{r}, s, \boldsymbol{x}_0)\Big) && \text{(Using Lemma 3.1)} \\ &= 2^p \boldsymbol{g}^\top (\boldsymbol{x}^\star - \boldsymbol{x}_0) - 2^p(h_p(\boldsymbol{r}, s, \boldsymbol{x}^\star) - h_p(\boldsymbol{r}, s, \boldsymbol{x}_0)) \\ &= 2^p(\mathcal{E}_1(\boldsymbol{x}^\star) - \mathcal{E}_1(\boldsymbol{x}_0)). \end{aligned}$$

Now, given a feasible solution δ to Problem R1, we must have $\boldsymbol{A\delta} = \mathbf{0}$. Thus, $\boldsymbol{Ax}_1 = \boldsymbol{Ax}_0 + 2^{-3p}\boldsymbol{A\delta} = b$, and $\boldsymbol{x}_1$ is a feasible solution to Problem P1. Moreover,

$$\begin{aligned}
\mathcal{E}_1(\boldsymbol{x}_1) &= \boldsymbol{g}^\top(\boldsymbol{x}_0 + 2^{-3p}\boldsymbol{\delta}) - h_p(\boldsymbol{r}, s, \boldsymbol{x}_0 + 2^{-3p}\boldsymbol{\delta}) \\
&\geq \boldsymbol{g}^\top \boldsymbol{x}_0 + 2^{-3p}\boldsymbol{g}^\top\boldsymbol{\delta} - h_p(\boldsymbol{r}, s, \boldsymbol{x}_0) - 2^{-3p}\boldsymbol{\delta}^\top \nabla_{\boldsymbol{x}} h_p(\boldsymbol{r}, s, \boldsymbol{x})|_{\boldsymbol{x}=\boldsymbol{x}_0} - 2^{2p} h_p(\boldsymbol{r} + s|\boldsymbol{x}_0|^{p-2}, s, 2^{-3p}\boldsymbol{\delta}) \\
&\qquad \text{(Using Lemma 3.1)} \\
&\geq \mathcal{E}_1(\boldsymbol{x}_0) + 2^{-4p}\boldsymbol{\delta}^\top \boldsymbol{g}' - 2^{2p} \cdot 2^{-6p} h_p(\boldsymbol{r} + s|\boldsymbol{x}_0|^{p-2}, s, \boldsymbol{\delta}) \qquad \text{(Using Lemma B.1)} \\
&= \mathcal{E}_1(\boldsymbol{x}_0) + 2^{-4p}\mathcal{E}_2(\boldsymbol{\delta}).
\end{aligned}$$

□

C Elimination of Low-Degree Vertices, and Loops

In this section, we that the instance $\mathcal{H}$ returned by UltraSparsify can be reduced to a smaller graph by repeatedly eliminating vertices of degree at most 2. This step is analogous to the partial Cholesky factorization in the Laplacian solver of Spielman and Teng [ST14]. A slight technical issue is that if we run into a cycle where at most 1 vertex on the cycle has edge(s) to the rest of the graph, the elimination of the degree 2 nodes on the cycle essentially becomes an optimization problem on only the cycle edges that can be solved independently from the rest of the graph.

Algorithm 8 Elimination of Degree 1 and 2 vertices and Self-loops

procedure Eliminate($\mathcal{H}$)
  Initiate $\mathcal{H}' \leftarrow \mathcal{H}$
  repeat
    For every edge with non-selfloop degree 1, remove the only non-selfloop edge incident on it
  until No vertex has non-selfloop degree 1
  for every maximal path with all internal nodes having non-selfloop degree 2 **do**
    Replace such a path with a single edge in $\mathcal{H}'$ with the end points as the end points of the path, and,

- resistance is the sum of the resistances of the edges on the path
- gradient is the sum of the gradients of the edges on the path
- s the same as before
- Flow on the new edge is mapped to a flow along the original path (or cycle) in $\mathcal{H}$.

  Move all self-loops from $\mathcal{H}'$ to $\mathcal{H}_{loop}$
  return $\mathcal{H}', \mathcal{H}_{loop}, \mathcal{M}_{(\mathcal{H}'+\mathcal{H}_{loop})\to\mathcal{H}}$

Theorem 3.4 (Eliminating vertices with degree 1 and 2)**.** *Given a smoothed p-norm instance $\mathcal{G}$, the algorithm* Eliminate($\mathcal{G}$) *returns another smoothed p-norm instance $\mathcal{G}'$, along with the map $\mathcal{M}_{\mathcal{G}'\to\mathcal{G}}$ in $O(|V^{\mathcal{G}}| + |E^{\mathcal{G}}|)$ time, such that the graph $G' = (V^{\mathcal{G}'}, E^{\mathcal{G}'})$ is obtained from the graph*

$G = (V^{\mathcal{G}}, E^{\mathcal{G}})$ by first repeatedly removing vertices with non-selfloop degree[8] *1 in G, and then replacing every path $u \rightsquigarrow v$ in G where all internal path vertices have non-selfloop degree exactly 2 in G, with a new edge (u, v).*

Moreover,

$$\mathcal{G}' \preceq^{cycle}_{\frac{1}{n^{\frac{1}{p-1}}}} \mathcal{G} \preceq^{cycle}_{1} \mathcal{G}',$$

where $n = \left|V^{\mathcal{G}}\right|$, and the map $\mathcal{M}_{\mathcal{G}' \to \mathcal{G}}$ can be applied in $O(\left|V^{\mathcal{G}}\right| + \left|E^{\mathcal{G}}\right|)$ time.

Proof. We first observe that a self-loop $e \in E^{\mathcal{G}}$ on a vertex $v \in V^{\mathcal{G}}$ does not contribute to the residue at any vertex, including v. Thus, the circulation constraint on a flow $\boldsymbol{f}^{\mathcal{G}}$ does not impose any constraint on $\boldsymbol{f}^{\mathcal{G}}_e$. Moreover, since the objective $\alpha^{\mathcal{G}}$ can be written as a sum over the edges, for every self-loop e, the variable $\boldsymbol{f}^{\mathcal{G}}_e$ is independent of all other variables. Thus, we can ignore the self-loops in remainder of the proof.

We first prove that we can repeatedly eliminate vertices of non-selfloop degree 1 in G while preserving $\mathcal{E}$ exactly for a circulation. Consider one such vertex $v \in V^{\mathcal{G}}$, and let $e = (v, u) \in E^{\mathcal{G}}$ be the only non-selfloop edge incident on v (the argument for the reverse direction is identical). Given any circulation $\boldsymbol{f}^{\mathcal{G}}$, since the only non-selfloop edge incident on v is e, we must have $\boldsymbol{f}^{\mathcal{G}}_e = 0$. Thus, we can drop e entirely from the instance. Formally, we define

$$V^{\mathcal{G}'} = V^{\mathcal{G}}, \quad E^{\mathcal{G}'} = E^{\mathcal{G}} \setminus \{e\}, \quad \boldsymbol{g}^{\mathcal{G}'} = \boldsymbol{g}^{\mathcal{G}}|_{E^{\mathcal{G}'}}, \quad \boldsymbol{r}^{\mathcal{G}'} = \boldsymbol{r}^{\mathcal{G}}|_{E^{\mathcal{G}'}}, \quad \text{and} \quad s^{\mathcal{G}'} = s^{\mathcal{G}}.$$

We let the mapping $\mathcal{M}_{\mathcal{G} \to \mathcal{G}'}$ to be just the projection on to $E^{\mathcal{G}'}$. Thus, $\mathcal{M}_{\mathcal{G} \to \mathcal{G}'}(\boldsymbol{f}^{\mathcal{G}}) = \boldsymbol{f}^{\mathcal{G}}|_{E^{\mathcal{G}'}}$. Since $\boldsymbol{f}^{\mathcal{G}}_e = 0$, we immediately get $\mathcal{E}^{\mathcal{G}'}(\mathcal{M}_{\mathcal{G} \to \mathcal{G}'}(\boldsymbol{f}^{\mathcal{G}})) = \mathcal{E}^{\mathcal{G}}(\boldsymbol{f}^{\mathcal{G}})$. Thus, $\mathcal{G} \preceq^{\text{cycle}}_{1} \mathcal{G}'$.

Now, consider the mapping $\mathcal{M}_{\mathcal{G}' \to \mathcal{G}}$ that pads a circulation $\boldsymbol{f}^{\mathcal{G}'}$ on $\mathcal{G}'$ with 0 on e, *i.e.*,

$$\left(\mathcal{M}_{\mathcal{G}' \to \mathcal{G}}(\boldsymbol{f}^{\mathcal{G}'})\right)_{e'} = \begin{cases} 0 & \text{if } e' = e, \\ \boldsymbol{f}^{\mathcal{G}'}_{e'} & \text{otherwise.} \end{cases}$$

Again, it is immediate that $\mathcal{E}^{\mathcal{G}}(\mathcal{M}_{\mathcal{G}' \to \mathcal{G}}(\boldsymbol{f}^{\mathcal{G}'})) = \mathcal{E}^{\mathcal{G}'}(\boldsymbol{f}^{\mathcal{G}'})$. Thus, $\mathcal{G}' \preceq^{\text{cycle}}_{1} \mathcal{G}$.

We can repeatedly apply the above transformation to eliminate all vertices of non-selfloop degree 1 in G. For convenience, we let $\mathcal{G}'$ denote the final instance obtained. Thus, we have, $\mathcal{G}' \preceq^{\text{cycle}}_{1} \mathcal{G} \preceq^{\text{cycle}}_{1} \mathcal{G}'$.

Now, we will replace maximal paths with all internal vertices of non-selfloop degree 2 with single edges. Consider such a path P. Formally, P is a path of length l in G, say $P = (v_0, v_1, \ldots, v_{l-1}, v_l)$, with all of $v_1, \ldots, v_{l-1}$ having degree exactly 2, and v_0, v_l have non-selfloop degree at least 3. For convenience, we assume that all edges (v_{i-1}, v_i) are oriented in the same direction. Observe that for a circulation $\boldsymbol{f}^{\mathcal{G}'}$, the flow on all the edges (v_{i-1}, v_i) must be the same, *i.e.*, $\boldsymbol{f}^{\mathcal{G}'}_{(v_{i-1}, v_i)}$ must all be equal. Thus, we can replace P with a single edge e_P while preserving the amount of flow and the direction.

Formally, let $\mathcal{P} = \{P_1, \ldots, P_t\}$ denote the set of all maximal paths in $G' = (V^{\mathcal{G}'}, E^{\mathcal{G}'})$ such that all their internal vertices have non-selfloop degree exactly 2 in G'. We replace each of these paths

[8]By non-selfloop degree, we mean that self-loops do not count towards the degree of a vertex.

with a new edge connecting its endpoints. Let,

$$V^{\mathcal{G}''} = V^{\mathcal{G}'},$$

$$E^{\mathcal{G}''} = E^{\mathcal{G}'} \cup_{P\in\mathcal{P}} \{e_p = (v_0, v_l) | P = (v_0, \ldots, v_l)\} \setminus \cup_{P\in\mathcal{P}} \{e = (v_{i-1}, v_i) \in P | P = (v_0, \ldots, v_l)\},$$

$$\boldsymbol{g}_e^{\mathcal{G}''} = \begin{cases} \boldsymbol{g}_e^{\mathcal{G}'} & \text{if } e \in E^{\mathcal{G}'} \cap E^{\mathcal{G}''}, \\ \sum_{e'\in P} \boldsymbol{g}_{e'}^{\mathcal{G}'} & \text{if } e = e_P \text{ for } P \in \mathcal{P}, \end{cases}$$

$$\boldsymbol{r}_e^{\mathcal{G}''} = \begin{cases} \boldsymbol{r}_e^{\mathcal{G}'} & \text{if } e \in E^{\mathcal{G}'} \cap E^{\mathcal{G}''}, \\ \sum_{e'\in P} \boldsymbol{r}_{e'}^{\mathcal{G}'} & \text{if } e = e_P \text{ for } P \in \mathcal{P}, \end{cases}$$

$$s^{\mathcal{G}''} = s^{\mathcal{G}}.$$

We define the mapping $\mathcal{M}_{\mathcal{G}'\to\mathcal{G}''}$ as follows

$$\left(\mathcal{M}_{\mathcal{G}'\to\mathcal{G}''}(\boldsymbol{f}^{\mathcal{G}'})\right)_e = \begin{cases} \boldsymbol{f}_e^{\mathcal{G}'} & \text{if } e \in E^{\mathcal{G}'} \cap E^{\mathcal{G}''}, \\ \boldsymbol{f}_{(v_0,v_1)}^{\mathcal{G}'} & \text{if } e = e_P, \text{ where } P = (v_0, \ldots, v_l), P \in \mathcal{P}. \end{cases}$$

We define $\mathcal{M}_{\mathcal{G}''\to\mathcal{G}'}$ to be the inverse map of $\mathcal{M}_{\mathcal{G}'\to\mathcal{G}''}$.

$$\left(\mathcal{M}_{\mathcal{G}''\to\mathcal{G}'}(\boldsymbol{f}^{\mathcal{G}''})\right)_e = \begin{cases} \boldsymbol{f}_e^{\mathcal{G}''} & \text{if } e \in E^{\mathcal{G}'} \cap E^{\mathcal{G}''}, \\ \boldsymbol{f}_P^{\mathcal{G}'} & \text{if } e = (v_{i-1}, v_i), i \in [l], \text{ where } P = (v_0, \ldots, v_l) \in \mathcal{P}. \end{cases}$$

It follows from the definitions that for every circulation $\boldsymbol{f}^{\mathcal{G}'}$, letting $\boldsymbol{f}^{\mathcal{G}''}$ denote $\mathcal{M}_{\mathcal{G}'\to\mathcal{G}''}(\boldsymbol{f}^{\mathcal{G}'})$, we have, $\mathcal{M}_{\mathcal{G}''\to\mathcal{G}'}(\boldsymbol{f}^{\mathcal{G}''}) = \boldsymbol{f}^{\mathcal{G}'}$. Moreover,

$$\left(\boldsymbol{g}^{\mathcal{G}'}\right)^\top \boldsymbol{f}^{\mathcal{G}'} = \left(\boldsymbol{g}^{\mathcal{G}'}\right)^\top \boldsymbol{f}^{\mathcal{G}''}$$

$$\sum_{e\in E^{\mathcal{G}'}} \boldsymbol{r}_e^{\mathcal{G}'} \left(\boldsymbol{f}_e^{\mathcal{G}'}\right)^2 = \sum_{e\in E^{\mathcal{G}''}} \boldsymbol{r}_e^{\mathcal{G}''} \left(\boldsymbol{f}_e^{\mathcal{G}''}\right)^2$$

$$s^{\mathcal{G}'} \sum_{e\in E^{\mathcal{G}'}} \left|\boldsymbol{f}_e^{\mathcal{G}'}\right|^p \geq s^{\mathcal{G}''} \sum_{e\in E^{\mathcal{G}''}} \left|\boldsymbol{f}_e^{\mathcal{G}''}\right|^p \geq \frac{1}{n} s^{\mathcal{G}'} \sum_{e\in E^{\mathcal{G}'}} \left|\boldsymbol{f}_e^{\mathcal{G}'}\right|^p,$$

where the last inequality follows since for every path of length l, the contribution to the ℓ_p^p changes by a factor of l^{-1}, and since the paths must be vertex-disjoint, $l \leq |V^{\mathcal{G}}| \leq n$. The above inequalities imply,

$$\mathcal{E}^{\mathcal{G}'}(\boldsymbol{f}^{\mathcal{G}'}) \leq \mathcal{E}^{\mathcal{G}''}(\boldsymbol{f}^{\mathcal{G}''}),$$

and hence $\mathcal{G}' \preceq_1^{\text{cycle}} \mathcal{G}''$. Combined with $\mathcal{G} \preceq_1^{\text{cycle}} \mathcal{G}'$, we get $\mathcal{G} \preceq_1^{\text{cycle}} \mathcal{G}''$. Moreover, we have, for $\kappa = n^{\frac{1}{p-1}}$,

$$\begin{aligned} \mathcal{E}^{\mathcal{G}'}\left(\kappa^{-1}\boldsymbol{f}^{\mathcal{G}'}\right) &= \left(\boldsymbol{g}^{\mathcal{G}'}\right)^\top \kappa^{-1}\boldsymbol{f}^{\mathcal{G}'} - \kappa^{-2} \sum_{e\in E^{\mathcal{G}'}} \boldsymbol{r}_e^{\mathcal{G}'} \left(\boldsymbol{f}_e^{\mathcal{G}'}\right)^2 - \kappa^{-p} s^{\mathcal{G}'} \sum_{e\in E^{\mathcal{G}'}} \left|\boldsymbol{f}_e^{\mathcal{G}'}\right|^p \\ &= \kappa^{-1} \left(\left(\boldsymbol{g}^{\mathcal{G}'}\right)^\top \boldsymbol{f}^{\mathcal{G}'} - \kappa^{-1} \sum_{e\in E^{\mathcal{G}'}} \boldsymbol{r}_e^{\mathcal{G}'} \left(\boldsymbol{f}_e^{\mathcal{G}'}\right)^2 - \frac{1}{n} s^{\mathcal{G}'} \sum_{e\in E^{\mathcal{G}'}} \left|\boldsymbol{f}_e^{\mathcal{G}'}\right|^p \right) \\ &\geq \kappa^{-1} \left(\left(\boldsymbol{g}^{\mathcal{G}''}\right)^\top \boldsymbol{f}^{\mathcal{G}''} - \sum_{e\in E^{\mathcal{G}''}} \boldsymbol{r}_e^{\mathcal{G}''} \left(\boldsymbol{f}_e^{\mathcal{G}''}\right)^2 - s^{\mathcal{G}''} \sum_{e\in E^{\mathcal{G}''}} \left|\boldsymbol{f}_e^{\mathcal{G}''}\right|^p \right) = \kappa^{-1} \mathcal{E}^{\mathcal{G}''}(\boldsymbol{f}^{\mathcal{G}''}). \end{aligned}$$

Thus, $\mathcal{G}'' \preceq_{\kappa}^{\text{cycle}} \mathcal{G}'$. Combining with $\mathcal{G}' \preceq_{1}^{\text{cycle}} \mathcal{G}$, we obtain $\mathcal{G}'' \preceq_{\kappa}^{\text{cycle}} \mathcal{G}$. The final instance returned is $\mathcal{G}''$, giving us our theorem. ☐

Lemma 3.5 (Eliminating Self-loops)**.** *There is an algorithm* RemoveLoops *such that, given a smoothed p-norm instance $\mathcal{G}$ with self-loops in $E^{\mathcal{G}}$, in $O(|V^{\mathcal{G}}| + |E^{\mathcal{G}}|)$ time, it returns instances $\mathcal{G}_1, \mathcal{G}_2$, such that $\mathcal{G} = \mathcal{G}_1 \cup \mathcal{G}_2$, where $\mathcal{G}_1$ is obtained from $\mathcal{G}$ by eliminating all self-loops from $E^{\mathcal{G}}$, and $\mathcal{G}_2$ is an instance consisting of just the self-loops from $\mathcal{G}$. Thus, any flow $\boldsymbol{f}^{\mathcal{G}_2}$ on $\mathcal{G}_2$ is a circulation.*

Moreover, there is an algorithm SolveLoops *that, given $\mathcal{G}_2$, for any $\delta \leq 1/p$, in time $O(|E^{\mathcal{G}_2}| \log 1/\delta)$, finds a circulation $\widetilde{\boldsymbol{f}}^{\mathcal{G}_2}$ on $\mathcal{G}_2$ such that*

$$\mathcal{E}^{\mathcal{G}_2}(\widetilde{\boldsymbol{f}}^{\mathcal{G}_2}) \geq (1-\delta) \max_{\boldsymbol{f}^{\mathcal{G}}:(\boldsymbol{B})^{\mathcal{G}}\boldsymbol{f}^{\mathcal{G}}=\boldsymbol{0}} \mathcal{E}^{\mathcal{G}_2}(\boldsymbol{f}^{\mathcal{G}_2}).$$

Proof. Let E' denote the set of all self-loops in $E^{\mathcal{G}}$. Then, we define $\mathcal{G}_1$ to be the instance obtained by removing all edges in E'. Formally,

$$\mathcal{G}_1 \stackrel{\text{def}}{=} (V^{\mathcal{G}}, E^{\mathcal{G}} \setminus E', \boldsymbol{g}^{\mathcal{G}}|_{E^{\mathcal{G}}\setminus E'}, \boldsymbol{r}^{\mathcal{G}}|_{E^{\mathcal{G}}\setminus E'}, s^{\mathcal{G}}).$$

We define $\mathcal{G}_2$ be the instance $\mathcal{G}$ restricted to E'. Thus,

$$\mathcal{G}_2 \stackrel{\text{def}}{=} (V^{\mathcal{G}}, E', \boldsymbol{g}^{\mathcal{G}}|_{E'}, \boldsymbol{r}^{\mathcal{G}}|_{E'}, s^{\mathcal{G}}).$$

It is immediate that $\mathcal{G} = \mathcal{G}_1 \cup \mathcal{G}_2$. Since $\mathcal{G}_2$ only has self-loops, we have that for every $\boldsymbol{f}^{\mathcal{G}_2}$, we have $(\boldsymbol{B}^{\mathcal{G}_2})^\top \boldsymbol{f}^{\mathcal{G}_2} = \boldsymbol{0}$. Thus, the constraint $(\boldsymbol{B}^{\mathcal{G}_2})^\top \boldsymbol{f}^{\mathcal{G}_2} = \boldsymbol{0}$ is vacuous.

Now, observe that in the absence of linear constraints on $\boldsymbol{f}^{\mathcal{G}_2}$, the variables $\boldsymbol{f}_e^{\mathcal{G}_2}$ are independent for all $e \in E^{\mathcal{G}_2}$. Moreover, we have

$$\mathcal{E}^{\mathcal{G}_2}(\boldsymbol{f}^{\mathcal{G}_2}) = \sum_{e \in E^{\mathcal{G}_2}} \mathcal{E}_e^{\mathcal{G}_2}(\boldsymbol{f}_e^{\mathcal{G}_2}).$$

Thus, we can solve for each $\boldsymbol{f}_e^{\mathcal{G}_2}$ independently. Now, consider a fixed $e \in E^{\mathcal{G}_2}$. We write f_e for $\boldsymbol{f}_e^{\mathcal{G}_2}$. We wish to solve

$$\max_{f_e} \mathcal{E}_e^{\mathcal{G}_2}(f_e) = \max_{f_e} \boldsymbol{g}_e^{\mathcal{G}_2} f_e - \boldsymbol{r}_e^{\mathcal{G}_2} f_e^2 - s^{\mathcal{G}_2}|f_e|^p.$$

Note that the objective function is concave. The gradient of $\mathcal{E}_e^{\mathcal{G}_2}(f_e)$ with respect to f_e is

$$\left(\mathcal{E}_e^{\mathcal{G}_2}\right)'(f_e) = \frac{\mathsf{d}}{\mathsf{d}f_e}\mathcal{E}_e^{\mathcal{G}_2}(f_e) = \boldsymbol{g}_e^{\mathcal{G}_2} - (2\boldsymbol{r}_e^{\mathcal{G}_2} + ps^{\mathcal{G}_2}|f_e|^{p-2})f_e.$$

First observe that if $f_e^\star$ is the optimal solution, it must have the same sign as $\boldsymbol{g}_e^{\mathcal{G}_2}$. Without loss of generality, we assume that $\boldsymbol{g}_e^{\mathcal{G}_2} \geq 0$. Observe that for $f_e \geq \frac{\boldsymbol{g}_e^{\mathcal{G}_2}}{2\boldsymbol{r}_e^{\mathcal{G}_2}}$, we have $\frac{\mathsf{d}}{\mathsf{d}f_e}\mathcal{E}_e^{\mathcal{G}_2} \leq 0$. Thus, $f_e^\star \leq \frac{\boldsymbol{g}_e^{\mathcal{G}_2}}{2\boldsymbol{r}_e^{\mathcal{G}_2}}$. Similarly, we have, $f_e^\star \leq \left(\frac{\boldsymbol{g}_e^{\mathcal{G}_2}}{ps^{\mathcal{G}_2}}\right)^{\frac{1}{p-1}}$, where $f_e^\star$ is the optimal solution. Thus, if we define z as

$$z \stackrel{\text{def}}{=} \min\left\{\frac{\boldsymbol{g}_e^{\mathcal{G}_2}}{2\boldsymbol{r}_e^{\mathcal{G}_2}}, \left(\frac{\boldsymbol{g}_e^{\mathcal{G}_2}}{ps^{\mathcal{G}_2}}\right)^{\frac{1}{p-1}}\right\},$$

then $f_e^\star \leq z$.

Moreover, for $f_e \le \frac{z}{2}$, we have,

$$\left(\mathcal{E}_e^{\mathcal{G}_2}\right)'(f_e) \ge \boldsymbol{g}_e^{\mathcal{G}_2} - \frac{2z\boldsymbol{r}_e^{\mathcal{G}_2}}{2} - \frac{ps^{\mathcal{G}_2}z^{p-1}}{2^{p-1}} \ge \boldsymbol{g}_e^{\mathcal{G}_2} - \frac{\boldsymbol{g}_e^{\mathcal{G}_2}}{2} - \frac{\boldsymbol{g}_e^{\mathcal{G}_2}}{2^{p-1}} \ge 0.$$

Thus, $f_e^\star \ge \frac{z}{2}$, and hence z gives a 2-approximation to $f^\star$ that can be computed in $O(1)$ time. Now, applying binary search allows us to find $f_e \in [(1 - \delta/p)f_e^\star, (1 + \delta/p)f_e^\star]$ in $O(\log 1/\delta)$ time. Now, we show that such an estimate is good enough. Consider the point $\frac{3}{4}z$. We have

$$\max_{f_e} \mathcal{E}_e^{\mathcal{G}_2}(f_e) \ge \mathcal{E}_e^{\mathcal{G}_2}(\frac{3}{4}z) \ge \frac{3}{4}\boldsymbol{g}_e^{\mathcal{G}} z(1 - \frac{1}{2}\frac{3}{4} - \frac{1}{p}\frac{3^{p-1}}{4^{p-1}}) \ge \frac{1}{4}z\boldsymbol{g}_e^{\mathcal{G}}.$$

Now,

$$\begin{aligned}
\mathcal{E}_e^{\mathcal{G}_2}(f_e) - \mathcal{E}_e^{\mathcal{G}_2}(f_e^\star) &\le \delta f_e^\star \max\left\{\left|\left(\mathcal{E}_e^{\mathcal{G}_2}\right)'((1-\delta)f_e^\star)\right|, \left|\left(\mathcal{E}_e^{\mathcal{G}_2}\right)'((1+\delta)f_e^\star)\right|\right\} \\
&\qquad \text{(Using mean-value theorem and concavity)} \\
&\le \delta f_e^\star \max\left\{\boldsymbol{g}_e^{\mathcal{G}}, -\boldsymbol{g}_e^{\mathcal{G}} + (1+\delta)2\boldsymbol{r}_e^{\mathcal{G}} f_e^\star + (1+\delta)^{p-1} ps^{\mathcal{G}} |f_e^\star|^{p-1}\right\} \\
&\le \delta f_e^\star \max\left\{\boldsymbol{g}_e^{\mathcal{G}}, -\boldsymbol{g}_e^{\mathcal{G}} + (1+\delta)\boldsymbol{g}_e^{\mathcal{G}} + (1+\delta)^{p-1}\boldsymbol{g}_e^{\mathcal{G}}\right\} \\
&\le 4\delta f_e^\star \boldsymbol{g}_e^{\mathcal{G}} \qquad \text{(Using } \delta \le 1/p\text{)} \\
&\le 4\delta z \boldsymbol{g}_e^{\mathcal{G}} \le 16\delta \max_{f_e} \mathcal{E}_e^{\mathcal{G}_2}(f_e)
\end{aligned}$$

Rewriting, we get $\mathcal{E}_e^{\mathcal{G}_2}(f_e) \ge (1 - 16\delta)\max_{f_e} \mathcal{E}_e^{\mathcal{G}_2}(f_e)$. Rescaling δ, we obtain our claim.

We can compute such an estimate for all the edges in $O(\left|E^{\mathcal{G}_2}\right| \log 1/\delta)$ time. □

D Sparsifying Uniform Expanders

We now verify that sparsifying that sampling α-uniform expanders preserve the objectives of the optimizations. Pseudocode of our routine and the flow maps constructed by it are in Algorithm 9. We remark that the maps are identical to the ones used for flow sparsifiers by Kelner et al. [KLOS14].

Algorithm 9 Producing Sparsifier

procedure SampleAndFixGradient($\mathcal{G} = (G, r^{\mathcal{G}}, s^{\mathcal{G}}, \boldsymbol{g}^{\mathcal{G}}), \tau$)
- Initialize $\mathcal{H}$ with $V^{\mathcal{H}} = V^{\mathcal{G}}$.
- Sample each edge of $E^{\mathcal{G}}$ independently w. probability τ to form $E^{\mathcal{H}}$.
- Let $r^{\mathcal{H}} \leftarrow \tau \cdot r^{\mathcal{G}}$ and $s^{\mathcal{H}} = \tau^p \cdot s^{\mathcal{G}}$
- Compute the decomposition $\boldsymbol{g}^{\mathcal{G}} = \widehat{\boldsymbol{g}}^{\mathcal{G}} + \boldsymbol{B}^{\mathcal{G}}\boldsymbol{\psi}$, s.t. $\widehat{\boldsymbol{g}}^{\mathcal{G}}$ is the cycle-space projection of $\boldsymbol{g}^{\mathcal{G}}$.
- Let $\widetilde{\boldsymbol{g}}^{\mathcal{H}} \leftarrow (\widehat{\boldsymbol{g}}^{\mathcal{G}})_{|F}$, i.e $\widetilde{\boldsymbol{g}}^{\mathcal{H}}$ is the restriction of $\widehat{\boldsymbol{g}}^{\mathcal{G}}$ to F.
- Let $\widehat{\boldsymbol{g}}^{\mathcal{H}} \leftarrow \left(I - \boldsymbol{B}^{\mathcal{H}}\left(\boldsymbol{B}^{H\top}\boldsymbol{B}^{\mathcal{H}}\right)^{\dagger}\boldsymbol{B}^{H\top}\right)\widetilde{\boldsymbol{g}}^{\mathcal{H}}$ i.e. the cycle-space projection of $\widetilde{\boldsymbol{g}}^{\mathcal{H}}$.
- Let $\boldsymbol{g}^{\mathcal{H}} \leftarrow \widehat{\boldsymbol{g}}^{\mathcal{H}} + \boldsymbol{B}^{\mathcal{H}}\boldsymbol{\psi}$
- Let $\mathcal{M}_{\mathcal{G}\to\mathcal{H}}$ be the map $\boldsymbol{f} \to \boldsymbol{B}^{\mathcal{H}}\left(\boldsymbol{B}^{H\top}\boldsymbol{B}^{\mathcal{H}}\right)^{\dagger}\boldsymbol{B}^{\mathcal{G}\top}\boldsymbol{f} + \frac{1}{\|\widehat{\boldsymbol{g}}^{\mathcal{H}}\|_2^2}\widehat{\boldsymbol{g}}^{\mathcal{H}}\widehat{\boldsymbol{g}}^{\mathcal{G}\top}\boldsymbol{f}$
- Let $\mathcal{M}_{\mathcal{H}\to\mathcal{G}}$ be the map $\boldsymbol{f} \to \boldsymbol{B}^{\mathcal{G}}\left(\boldsymbol{B}^{\mathcal{G}\top}\boldsymbol{B}^{\mathcal{G}}\right)^{\dagger}\boldsymbol{B}^{H\top}\boldsymbol{f} + \frac{1}{\|\widehat{\boldsymbol{g}}^{\mathcal{G}}\|_2^2}\widehat{\boldsymbol{g}}^{\mathcal{G}}\widehat{\boldsymbol{g}}^{H\top}\boldsymbol{f}$
- **return** $\mathcal{H} = (V^{\mathcal{H}}, E^{\mathcal{H}}, r^{\mathcal{H}}, s^{\mathcal{H}}, \boldsymbol{g}^{\mathcal{H}}), \mathcal{M}_{\mathcal{G}\to\mathcal{H}}, \mathcal{M}_{\mathcal{H}\to\mathcal{G}}$

Note that the decomposition in Line (5) can be found by first computing $\boldsymbol{\psi} = \left(\boldsymbol{B}^{\mathcal{G}\top}\boldsymbol{B}^{\mathcal{G}}\right)^{\dagger}\boldsymbol{B}^{\mathcal{G}\top}\boldsymbol{g}^{\mathcal{G}}$. The only randomness in the Algorithm 9 is in the sampling in Line (3). In Lines (5), and (7)-(10), when the pseudo-inverse of a Laplacian is applied, we can rely deterministically on a high-accuracy approximation based on the fact that if the earlier sampling succeeded, both matrices are Laplacians of expanders, and hence well-conditioned. Alternatively, we can call a high-accuracy Laplacian solver. This encurs another small failure probability. In either case, we can ensure that an implicit representation of the operator is only computed once, and succeeds with high probability.

Theorem 4.10 (Sampling Uniform Expanders)**.** *Given an α-uniform ϕ-expander $\mathcal{G} = (V^{\mathcal{G}}E^{\mathcal{G}}, r^{\mathcal{G}}, s^{\mathcal{G}}, \boldsymbol{g}^{\mathcal{G}})$ with m edges and vertex degrees at least $d_{\min}$, for any sampling probability τ satisfying*

$$\tau \geq c_{sample} \cdot \log n \cdot \left(\frac{\alpha}{m} + \frac{1}{\phi^2 d_{\min}}\right),$$

where c_{sample} is some absolute constant, SampleAndFixGradient$(\mathcal{G}, \tau)$ *w.h.p. returns a partial instance $\mathcal{H} = (H, r^{\mathcal{H}}, s^{\mathcal{H}}, \boldsymbol{g}^{\mathcal{H}})$ and maps $\mathcal{M}_{\mathcal{G}\to\mathcal{H}}$ and $\mathcal{M}_{\mathcal{H}\to\mathcal{G}}$. The graph H has the same vertex set as G, and H has at most $2\tau m$ edges. Furthermore, $r^{\mathcal{H}} = \tau \cdot r^{\mathcal{G}}$ and $s^{\mathcal{H}} = \tau^p \cdot s^{\mathcal{G}}$. The maps $\mathcal{M}_{\mathcal{G}\to\mathcal{H}}$ and $\mathcal{M}_{\mathcal{H}\to\mathcal{G}}$ certify*

$$\mathcal{H} \preceq_\kappa \mathcal{G} \text{ and } \mathcal{G} \preceq_\kappa \mathcal{H},$$

where $\kappa = m^{1/(p-1)}\phi^{-9}\log^3 n$.

Remark D.1. If in Algorithm 9, the input gradient $\boldsymbol{g}^{\mathcal{G}}$ has zero cycle-space projection, i.e. $\widehat{\boldsymbol{g}}^{\mathcal{G}} = \boldsymbol{0}$, then the cycle-space gradient terms in Lines (9) and (10) should be set to zero, so that $\mathcal{M}_{\mathcal{G}\to\mathcal{H}}$ is the map $\boldsymbol{f} \to \boldsymbol{B}^{\mathcal{H}}\left(\boldsymbol{B}^{H\top}\boldsymbol{B}^{\mathcal{H}}\right)^{\dagger}\boldsymbol{B}^{\mathcal{G}\top}\boldsymbol{f}$ and $\mathcal{M}_{\mathcal{H}\to\mathcal{G}}$ is the map $\boldsymbol{f} \to \boldsymbol{B}^{\mathcal{G}}\left(\boldsymbol{B}^{\mathcal{G}\top}\boldsymbol{B}^{\mathcal{G}}\right)^{\dagger}\boldsymbol{B}^{H\top}\boldsymbol{f}$. The proof of this case is simpler, we omit all terms that deal with cycle-space projected gradients and everything else stays the same as in the proof given in this section.

To prove this theorem, we first collect a number of observations that will help us. The most basic of these is that Line (3) succeeds in producing a sparsifier in the spectral approximation sense, and with edge set F satisfying $0.5\tau m \leq |F| \leq 2\tau m$. This is a direct consequence of matrix concentration bounds [Tro12].

Lemma D.2. *Consider the edge-vertex incidence matrices with gradients (projected via $\boldsymbol{\psi}^{\mathcal{G}}$) appended as an extra column for both $\mathcal{G}$ and $\mathcal{H}$, $[\boldsymbol{B}^{\mathcal{G}}, \widehat{\boldsymbol{g}}^{\mathcal{G}}]$ and $[\boldsymbol{B}^{\mathcal{H}}, \widetilde{\boldsymbol{g}}^{\mathcal{H}}]$. With high probability we have that for any vector $\boldsymbol{x}$*

$$\tau \left\| \left[\boldsymbol{B}^{\mathcal{G}}, \widehat{\boldsymbol{g}}^{\mathcal{G}}\right] \boldsymbol{x} \right\|_2^2 \approx_{0.1} \left\| \left[\boldsymbol{B}^{\mathcal{H}}, \widetilde{\boldsymbol{g}}^{\mathcal{H}}\right] \boldsymbol{x} \right\|_2^2 \tag{12}$$

and the edge set F of H satisfies $0.5\tau m \leq |F| \leq 2\tau m$.

Proof. The bounds on $|F|$ follow from a scalar Chernoff bound. For the matrix approximation bound, we will invoke matrix Chernoff bounds [Tro12], which give such a bound as long as the rows of $[\boldsymbol{B}^{\mathcal{G}}, \widehat{\boldsymbol{g}}^{\mathcal{G}}]$ are sampled with probaiblity exceeding $c_{sample} \log n$ times their leverage scores.

So it suffices to bound the leverage scores of the rows of this matrix. As $\boldsymbol{B}^{\mathcal{G}}$ and $\widehat{\boldsymbol{g}}^{\mathcal{G}}$ are orthogonal to each other, we can bound the leverage scores of the rows in these two matrices and add them.

The fact that the graph $(V^{\mathcal{G}}, E^{\mathcal{G}})$ has expansion ϕ means that its normalized Laplacian has eigenvalue at least ϕ^{-2}. So the leverage score of a row of $\boldsymbol{B}^{\mathcal{G}}$ is at least $\phi^{-2}d_{\min}$. The leverage score of $\widehat{\boldsymbol{g}}^{\mathcal{G}}_e$ in $\widehat{\boldsymbol{g}}$ on the other hand is at most α/m due to the α-uniform assumption. Thus, the sampling probablity τ meets the requirements of matrix Chernoff bounds, and we get the approximation with high probability.

□

Corollary D.3. *Assuming Equation* (12)*, the graphs underlying $\mathcal{G}$ and $\mathcal{H}$ (with resistances $r^{\mathcal{G}}$ and $r^{\mathcal{H}}$) are spectral approximations of each other:*

$$\tau \boldsymbol{B}^{\mathcal{G}\top} \boldsymbol{B}^{\mathcal{G}} \approx_{0.1} \boldsymbol{B}^{H\top} \boldsymbol{B}^{\mathcal{H}} \tag{13}$$

and the subset of gradient terms chosen after rescaling, $\widetilde{\boldsymbol{g}}^{\mathcal{H}}$, has ℓ_2^2 norm that's bigger by a factor of about τ:

$$\tau \left\|\widehat{\boldsymbol{g}}^{\mathcal{G}}\right\|_2^2 \approx_{0.1} \left\|\widetilde{\boldsymbol{g}}^{\mathcal{H}}\right\|_2^2. \tag{14}$$

Proof. The approximation of graphs follows from considering vectors $\boldsymbol{x}$ with 0 in the last coordinate.

The approximation of ℓ_2^2 norms of vectors follow from considering the indicator vector with 1 in the last column and 0 everywhere else. □

From this spectral approximation, we can also conclude that $(V^{\mathcal{H}}, E^{\mathcal{H}})$ must be an expander, as captured by the next corollary.

Corollary D.4. *Assuming Equation* (12)*, H has conductance at least 0.8ϕ.*

Proof. Let $C_{\mathcal{G}}(S)$ and $C_{\mathcal{H}}(S)$ denote the number of edges of $E^{\mathcal{G}}$ and $E^{\mathcal{H}}$ respectively crossing a cut $S \subseteq V^{\mathcal{G}} = V^{\mathcal{H}}$. Condition (13) implies that cuts are preserved between $(V^{\mathcal{G}}, E^{\mathcal{G}})$ and $(V^{\mathcal{H}}, E^{\mathcal{H}})$: For all $S \subseteq V$ $\tau C_{\mathcal{G}}(S) \approx_{0.1} C_{\mathcal{H}}(S)$, by computing the quadratic form in an indicator vector of S.

The degree of every vertex is also preserved, to up a scaling of τ and a multiplicative error 1 ± 0.1, i.e. $\tau \deg_{\mathcal{H}}(v) \approx_{0.1} \deg_{\mathcal{G}}(v)$. This follows from considering the quadratic form of Condition (13) in the indicator vector of vertex v. This implies for any S,

$$\frac{C_{\mathcal{H}}(S)}{\sum_{v \in S} \deg_{\mathcal{H}}(v)} \approx_{0.2} \frac{C_{\mathcal{G}}(S)}{\sum_{v \in S} \deg_{\mathcal{G}}(v)},$$

from which we conclude the conductance is preserved up to a factor of 0.8. □

We can also conclude from this that $\widehat{\boldsymbol{g}}^{\mathcal{H}}$ is well-spread.

Corollary D.5. *Assuming Equation* (12)*, the projection of $\boldsymbol{g}^{\mathcal{H}}$ onto the cycle-space of H, $\widehat{\boldsymbol{g}}^{\mathcal{H}}$ has ℓ_1 and ℓ_2 norms that are close to τ times the corresponding terms in $\mathcal{G}$:*

$$\left\|\widehat{\boldsymbol{g}}^{\mathcal{H}}\right\|_2^2 \approx_{0.2} \tau \left\|\widehat{\boldsymbol{g}}^{\mathcal{G}}\right\|_2^2 \tag{15}$$

$$\left\|\widehat{\boldsymbol{g}}^{\mathcal{H}}\right\|_1 \approx_{O(\alpha\phi^{-6}\log^2 n)} \tau \left\|\widehat{\boldsymbol{g}}^{\mathcal{G}}\right\|_1 \tag{16}$$

And $\widehat{\boldsymbol{g}}^{\mathcal{H}}$ is $O(\alpha\phi^{-6}\log^2 n)$-well spread, i.e. (as $|F|$ is the number of entries of $\widehat{\boldsymbol{g}}^{\mathcal{H}}$)

$$\left\|\widehat{\boldsymbol{g}}^{\mathcal{H}}\right\|_\infty^2 \leq \frac{O(\alpha\phi^{-6}\log^2 n)}{|F|} \left\|\widehat{\boldsymbol{g}}^{\mathcal{H}}\right\|_2^2. \tag{17}$$

Proof. We first show $\left\|\widehat{\boldsymbol{g}}^{\mathcal{H}}\right\|_2^2 \approx_{0.5} \tau\left\|\widehat{\boldsymbol{g}}^{\mathcal{G}}\right\|_2^2$. Note that $\left\|\widehat{\boldsymbol{g}}^{\mathcal{G}}\right\|_2^2 = \left\|\boldsymbol{B}^{\mathcal{G}}\boldsymbol{0} + \widehat{\boldsymbol{g}}^{\mathcal{G}}\right\|_2^2$ Now, consider $\boldsymbol{x} = \begin{pmatrix}\boldsymbol{0}\\1\end{pmatrix}$. By Equation (12),

$$\tau\left\|\widehat{\boldsymbol{g}}^{\mathcal{G}}\right\|_2^2 = \tau\left\|\left[\boldsymbol{B}^{\mathcal{G}}, \widehat{\boldsymbol{g}}^{\mathcal{G}}\right]\boldsymbol{x}\right\|_2^2 \approx_{0.1} \left\|\left[\boldsymbol{B}^{\mathcal{H}}, \widetilde{\boldsymbol{g}}^{\mathcal{H}}\right]\boldsymbol{x}\right\|_2^2$$
$$\geq \min_{\boldsymbol{y}}\left\|\left[\boldsymbol{B}^{\mathcal{H}}, \widetilde{\boldsymbol{g}}^{\mathcal{H}}\right]\begin{pmatrix}\boldsymbol{y}\\1\end{pmatrix}\right\|_2^2 = \left\|\widehat{\boldsymbol{g}}^{\mathcal{H}}\right\|_2^2.$$

Thus $\tau\left\|\widehat{\boldsymbol{g}}^{\mathcal{G}}\right\|_2^2 \geq 0.9\left\|\widehat{\boldsymbol{g}}^{\mathcal{H}}\right\|_2^2$.

The definition of $\widehat{\boldsymbol{g}}^{\mathcal{G}}$ ensures $\left\|\widehat{\boldsymbol{g}}^{\mathcal{G}}\right\|_2^2 = \min_{\boldsymbol{y}}\left\|\boldsymbol{B}^{\mathcal{G}}\boldsymbol{y} + \widehat{\boldsymbol{g}}^{\mathcal{G}}\right\|_2^2$. Letting $\boldsymbol{y}^{\mathcal{H}} \in \arg\min_{\boldsymbol{y}}\left\|\boldsymbol{B}^{\mathcal{H}}\boldsymbol{y} + \widetilde{\boldsymbol{g}}^{\mathcal{H}}\right\|_2^2$, we get from the definition of $\widehat{\boldsymbol{g}}^{\mathcal{H}}$, that

$$\left\|\widehat{\boldsymbol{g}}^{\mathcal{H}}\right\|_2^2 = \left\|\left[\boldsymbol{B}^{\mathcal{H}}, \widetilde{\boldsymbol{g}}^{\mathcal{H}}\right]\begin{pmatrix}\boldsymbol{y}^{\mathcal{H}}\\1\end{pmatrix}\right\|_2^2 \approx_{0.1} \tau\left\|\left[\boldsymbol{B}^{\mathcal{G}}, \widehat{\boldsymbol{g}}^{\mathcal{G}}\right]\begin{pmatrix}\boldsymbol{y}^{\mathcal{H}}\\1\end{pmatrix}\right\|_2^2$$
$$\geq \tau\min_{\boldsymbol{y}}\left\|\left[\boldsymbol{B}^{\mathcal{G}}, \widehat{\boldsymbol{g}}^{\mathcal{G}}\right]\begin{pmatrix}\boldsymbol{y}\\1\end{pmatrix}\right\|_2^2 = \tau\left\|\widehat{\boldsymbol{g}}^{\mathcal{G}}\right\|_2^2.$$

Thus we also have $\tau\left\|\widehat{\boldsymbol{g}}^{\mathcal{H}}\right\|_2^2 \geq 0.9\left\|\widehat{\boldsymbol{g}}^{\mathcal{G}}\right\|_2^2$, allowing us to conclude that Equation (15) is satisfied.

Next, observe that by Lemma 5.2, since H has conductance at least 0.8ϕ,

$$\left\|\widehat{\boldsymbol{g}}^{\mathcal{H}}\right\|_\infty = \left\|\left(I - \boldsymbol{B}^{\mathcal{H}}\left(\boldsymbol{B}^{\mathcal{H}\top}\boldsymbol{B}^{\mathcal{H}}\right)^\dagger\boldsymbol{B}^{\mathcal{H}\top}\right)\widetilde{\boldsymbol{g}}^{\mathcal{H}}\right\|_\infty \leq \left\|\left(I - \boldsymbol{B}^{\mathcal{H}}\left(\boldsymbol{B}^{\mathcal{H}\top}\boldsymbol{B}^{\mathcal{H}}\right)^\dagger\boldsymbol{B}^{\mathcal{H}\top}\right)\right\|_\infty\left\|\widetilde{\boldsymbol{g}}^{\mathcal{H}}\right\|_\infty$$
$$\leq O(\phi^{-3}\log n)\left\|\widehat{\boldsymbol{g}}^{\mathcal{G}}\right\|_\infty,$$

where in the last step we also used $\left\|\widetilde{\boldsymbol{g}}^{\mathcal{H}}\right\|_\infty \leq \left\|\widehat{\boldsymbol{g}}^{\mathcal{G}}\right\|_\infty$, since the former vector consists of a subset of the entries of the latter. Furthermore, by combining the above inequality with the assumption that $\widehat{\boldsymbol{g}}^{\mathcal{G}}$ is α-well-spread, and Equation (15) holds, we get

$$\left\|\widehat{\boldsymbol{g}}^{\mathcal{H}}\right\|_\infty^2 \leq O(\phi^{-6}\log^2 n)\frac{\alpha}{m}\left\|\widehat{\boldsymbol{g}}^{\mathcal{G}}\right\|_2^2 \leq \frac{O(\alpha\phi^{-6}\log^2 n)}{\tau m}\left\|\widehat{\boldsymbol{g}}^{\mathcal{H}}\right\|_2^2.$$

As $\widehat{\boldsymbol{g}}^{\mathcal{H}}$ has $|F| \leq 2\tau m$ entries, this shows that it is $O(\alpha\phi^{-6}\log^2 n)$-well-spread, which establishes Equation (17).

Next, to prove that Equation (16) holds, we first observe that for any γ-well-spread vector on $\boldsymbol{x}$ with t coordinates, since $\|\boldsymbol{x}\|_1\|\boldsymbol{x}\|_\infty \geq \|\boldsymbol{x}\|_2^2$,

$$\|\boldsymbol{x}\|_1 \geq \frac{\|\boldsymbol{x}\|_2^2}{\|\boldsymbol{x}\|_\infty} \geq \frac{t}{\gamma}\|\boldsymbol{x}\|_\infty \geq \frac{t^{1/2}}{\gamma}\|\boldsymbol{x}\|_2.$$

So $\frac{t^{1/2}}{\gamma}\|\boldsymbol{x}\|_2 \leq \|\boldsymbol{x}\|_1 \leq t^{1/2}\|\boldsymbol{x}\|_2$. As $\widehat{\boldsymbol{g}}^{\mathcal{G}}$ and $\widehat{\boldsymbol{g}}^{\mathcal{H}}$ are α and $O(\alpha\phi^{-6}\log^2 n)$-well-spread respectively, we then get

$$\Omega(1)\frac{\frac{|F|}{(\alpha\phi^{-6}\log^2 n)^2}\left\|\widehat{\boldsymbol{g}}^{\mathcal{H}}\right\|_2^2}{m\left\|\widehat{\boldsymbol{g}}^{\mathcal{G}}\right\|_2^2} \leq \frac{\left\|\widehat{\boldsymbol{g}}^{\mathcal{H}}\right\|_1^2}{\left\|\widehat{\boldsymbol{g}}^{\mathcal{G}}\right\|_1^2} \leq O(1)\frac{|F|\left\|\widehat{\boldsymbol{g}}^{\mathcal{H}}\right\|_2^2}{\frac{m}{\alpha^2}\left\|\widehat{\boldsymbol{g}}^{\mathcal{G}}\right\|_2^2}.$$

Combining this with $\left\|\widehat{\boldsymbol{g}}^{\mathcal{H}}\right\|_2^2 \approx_{0.2} \tau \left\|\widehat{\boldsymbol{g}}^{\mathcal{G}}\right\|_2^2$, and $0.5\tau \leq \frac{|F|}{m} \leq 2\tau$, we get

$$\Omega(1)\frac{\tau^2}{(\alpha\phi^{-6}\log^2 n)^2} \leq \frac{\left\|\widehat{\boldsymbol{g}}^{\mathcal{H}}\right\|_1^2}{\left\|\widehat{\boldsymbol{g}}^{\mathcal{G}}\right\|_1^2} \leq O(1)\alpha^2\tau^2.$$

From this we can directly conclude that Equation (16) holds. ☐

Also, we can show that for any $\boldsymbol{b}$ and θ, the optimum ℓ_2 as well as ℓ_p energies are close to the per degree lower bounds. We start with the lower bounds.

Lemma D.6. *Consider any graph with degrees $\boldsymbol{D}$, uniform r and s, gradient $\boldsymbol{g}$ decomposable into $\boldsymbol{g} = \widehat{\boldsymbol{g}} + \boldsymbol{B}\boldsymbol{\psi}$ where $\widehat{\boldsymbol{g}}$ is the cycle-space projection of $\boldsymbol{g}$, and any θ, any flow $\boldsymbol{f}$ such that:*

1. *$\boldsymbol{f}$ has residues $\boldsymbol{b}$: $\boldsymbol{B}^\top \boldsymbol{f} = \boldsymbol{b}$, and*
2. *$\boldsymbol{f}$ has dot product $\theta + \boldsymbol{b}^\top\boldsymbol{\psi}$ with $\boldsymbol{g}$, i.e. $\boldsymbol{g}^\top \boldsymbol{f} = \theta + \boldsymbol{b}^\top\boldsymbol{\psi}$*

must satisfy

$$\sum_e \boldsymbol{r}_e \boldsymbol{f}_e^2 + \boldsymbol{s}_e|\boldsymbol{f}_e|^p \geq \Omega\left(r \cdot \|\boldsymbol{b}\|_{\boldsymbol{D}^{-1}}^2 + r \cdot \frac{\theta^2}{\|\widehat{\boldsymbol{g}}\|_2^2} + s \cdot \left\|\boldsymbol{D}^{-1}\boldsymbol{b}\right\|_\infty^p + s \cdot \left(\frac{\theta}{\|\widehat{\boldsymbol{g}}\|_1}\right)^p\right).$$

Proof. First, note that because $\boldsymbol{B}^\top \boldsymbol{f} = \boldsymbol{b}$, we have

$$\widehat{\boldsymbol{g}}^\top \boldsymbol{f} = (\boldsymbol{g} - \boldsymbol{B}\boldsymbol{\psi})^\top \boldsymbol{f} = \boldsymbol{g}^\top \boldsymbol{f} - \boldsymbol{\psi}^\top\left(\boldsymbol{B}^\top \boldsymbol{f}\right) = \boldsymbol{g}^\top \boldsymbol{f} - \boldsymbol{x}^\top \boldsymbol{b} = \theta.$$

That is, the dot of $\boldsymbol{f}$ against $\widehat{\boldsymbol{g}}$ must be θ.

The total energy is a sum of the ℓ_2^2 and ℓ_p^p terms. First, we will give two different lower bounds on the ℓ_2^2, and can hence also conclude that the average of the two lower bounds is another lower bound. We then do the same for the ℓ_p^p terms and add the lower bounds together for a lower bound on the overall objective. We will do so separately, by matching the ℓ_2^2 terms to the electrical energy, and the ℓ_p^p terms to the minimum congestion.

1. $\|\boldsymbol{f}\|_2^2 \geq \|\boldsymbol{b}\|_{\boldsymbol{D}^{-1}}^2$: here we use the fact that the minimum energy of the electrical flow is given by

   $$\boldsymbol{b}^\top \boldsymbol{L}^\dagger \boldsymbol{b},$$

   and that the graph Laplacian is dominated by twice its diagonal

   $$\boldsymbol{L} \preceq 2\boldsymbol{D},$$

   to get

   $$\|\boldsymbol{b}\|_{\boldsymbol{L}^\dagger}^2 \geq \|\boldsymbol{b}\|_{\frac{1}{2}\boldsymbol{D}^{-1}}^2.$$

2. $\|\boldsymbol{f}\|_2^2 \geq \frac{\theta^2}{\|\widehat{\boldsymbol{g}}\|_2^2}$ is by rearranging Cauchy-Schwarz inequality, which in its simplest form gives

   $$\|\boldsymbol{f}\|_2 \cdot \|\widehat{\boldsymbol{g}}\|_2 \geq \left|\boldsymbol{f}^\top \widehat{\boldsymbol{g}}\right| = |\theta|.$$

3. $\|\boldsymbol{f}\|_p^p \geq \left\|\boldsymbol{D}^{-1}\boldsymbol{b}\right\|_\infty^p$ is because if we have a residue of $\boldsymbol{b}_u$ at some vertex, then some edge incident to u must have flow at least
$$\frac{\boldsymbol{b}_u}{\boldsymbol{d}_u}$$
on it. The p-th power of that lower bounds the overall p-norm energy.

4. $\|\boldsymbol{f}\|_p^p \geq (\frac{|\theta|}{\|\widehat{\boldsymbol{g}}\|_1})^p$ uses a similar lower bound on $\|\boldsymbol{f}\|_\infty$, except using Holder's inequality on ℓ_∞ and ℓ_1 norms to obtain
$$\|\boldsymbol{f}\|_\infty \|\widehat{\boldsymbol{g}}\|_1 \geq |\theta|,$$
which rearranges to give
$$\|\boldsymbol{f}\|_\infty \geq \frac{|\theta|}{\|\widehat{\boldsymbol{g}}\|_1}.$$

□

Before proving upper bounds on the energy required to route a flow in the graph, we state a lemma that upper bounds the energy required to route the "electrical" component of the flow, i.e. the projection of the flow orthogonal to the cycle space.

Lemma D.7. *Consider a graph G with degrees $\boldsymbol{D}$, conductance ϕ, and edge-vertex incidence matrix $\boldsymbol{B}$, and any demand $\boldsymbol{b} \perp \boldsymbol{1}$. Define the electrical flow $\boldsymbol{f} = \boldsymbol{B}\left(\boldsymbol{B}^\top \boldsymbol{B}\right)^\dagger \boldsymbol{b}$. Then $\sum_e \boldsymbol{f}_e^2 \leq 2\phi^{-2}\|\boldsymbol{b}\|_{\boldsymbol{D}^{-1}}^2$.*

The proof relies on first Cheeger's Inequality (e.g. see [Spi18]):

Theorem D.8. *(Cheeger's Inequality) Consider a graph G with degrees $\boldsymbol{D}$, conductance ϕ, and adjancency matrix $\boldsymbol{A}$. Then*

$$0.5\phi^2 \leq \min_{\boldsymbol{x} \perp \boldsymbol{D}\boldsymbol{1}} \frac{\boldsymbol{x}^\top(\boldsymbol{D}-\boldsymbol{A})\boldsymbol{x}}{\boldsymbol{x}^\top \boldsymbol{D}\boldsymbol{x}} \leq 2\phi$$

We will also need the following helpful fact.

Fact D.9. *Suppose $\boldsymbol{M} = \boldsymbol{X}\boldsymbol{A}\boldsymbol{X}^\top$ where $\boldsymbol{A}$ is symmetric and $\boldsymbol{X}$ is non-singular, and that $\boldsymbol{P}$ is the projection orthogonal to the kernel of $\boldsymbol{M}$, i.e. $\boldsymbol{P} = \boldsymbol{M}^\dagger \boldsymbol{M} = \boldsymbol{M}\boldsymbol{M}^\dagger$. Then $\boldsymbol{M}^\dagger = \boldsymbol{P}\boldsymbol{X}^{-\top}\boldsymbol{A}^\dagger \boldsymbol{X}^{-1}\boldsymbol{P}$.*

Proof of Lemma D.7. Note $\boldsymbol{B}^\top \boldsymbol{B} = \boldsymbol{D} - \boldsymbol{A} = \boldsymbol{L}$, where $\boldsymbol{A}$ is the adjacency matrix of the graph, and $\boldsymbol{L}$ is its Laplacian. Also $\sum_e \boldsymbol{f}_e^2 = (\boldsymbol{B}\left(\boldsymbol{B}^\top \boldsymbol{B}\right)^\dagger \boldsymbol{b})^\top \boldsymbol{B}\left(\boldsymbol{B}^\top \boldsymbol{B}\right)^\dagger \boldsymbol{b} = \boldsymbol{b}^\top \left(\boldsymbol{B}^\top \boldsymbol{B}\right)^\dagger \boldsymbol{b} = \boldsymbol{b}^\top \boldsymbol{L}^\dagger \boldsymbol{b}$. By Theorem D.8, we get that for $\boldsymbol{x} \perp \boldsymbol{D}\boldsymbol{1}$

$$\boldsymbol{x}^\top \boldsymbol{L}\boldsymbol{x} \geq 0.5\phi^2 \boldsymbol{x}^\top \boldsymbol{D}\boldsymbol{x}.$$

Substituting $\boldsymbol{y} = \boldsymbol{D}^{1/2}\boldsymbol{x}$ changes the constraint to $\boldsymbol{D}^{-1/2}\boldsymbol{y} \perp \boldsymbol{D}\boldsymbol{1}$ i.e. $\boldsymbol{y} \perp \boldsymbol{D}^{1/2}\boldsymbol{1}$. The inequality now states
$$\boldsymbol{y}^\top \boldsymbol{D}^{-1/2}\boldsymbol{L}\boldsymbol{D}^{-1/2}\boldsymbol{y} \geq 0.5\phi^2 \boldsymbol{y}^\top \boldsymbol{y}.$$
If we let $\boldsymbol{Q}$ denote the projection orthogonal to $\boldsymbol{D}^{1/2}\boldsymbol{1}$, we can summarize the inequality and orthogonality constraint in one condition using the Loewner order as

$$\boldsymbol{Q}\boldsymbol{D}^{-1/2}\boldsymbol{L}\boldsymbol{D}^{-1/2}\boldsymbol{Q} \succeq 0.5\phi^2 \boldsymbol{Q}.$$

Note that the null space of $\boldsymbol{D}^{-1/2}\boldsymbol{L}\boldsymbol{D}^{-1/2}$ is spanned by $\boldsymbol{D}^{1/2}\mathbf{1}$, as $\mathbf{1}$ spans the null space of $\boldsymbol{L}$. So in fact $\boldsymbol{Q}\boldsymbol{D}^{-1/2}\boldsymbol{L}\boldsymbol{D}^{-1/2}\boldsymbol{Q} = \boldsymbol{D}^{-1/2}\boldsymbol{L}\boldsymbol{D}^{-1/2}$, and we can conclude

$$\boldsymbol{D}^{-1/2}\boldsymbol{L}\boldsymbol{D}^{-1/2} \succeq 0.5\phi^2\boldsymbol{Q}.$$

From this we conclude that,

$$(\boldsymbol{D}^{-1/2}\boldsymbol{L}\boldsymbol{D}^{-1/2})^\dagger \preceq 2\phi^{-2}\boldsymbol{Q}^\dagger$$

as $\boldsymbol{A} \succeq \boldsymbol{B}$ implies $\boldsymbol{A}^\dagger \preceq \boldsymbol{B}^\dagger$ when $\boldsymbol{A}$ and $\boldsymbol{B}$ have the same null space. Hence by Fact D.9 and $\boldsymbol{Q} = \boldsymbol{Q}^\dagger$, we then get

$$\boldsymbol{Q}\boldsymbol{D}^{1/2}\boldsymbol{L}^\dagger\boldsymbol{D}^{1/2}\boldsymbol{Q} \preceq 2\phi^{-2}\boldsymbol{Q}.$$

This we can rewrite as for all $\boldsymbol{y} \perp \boldsymbol{D}^{1/2}\mathbf{1}$.

$$\boldsymbol{y}^\top\boldsymbol{D}^{1/2}\boldsymbol{L}^\dagger\boldsymbol{D}^{1/2}\boldsymbol{y} \leq 2\phi^{-2}\boldsymbol{y}^\top\boldsymbol{y}.$$

Substituting $\boldsymbol{z} = \boldsymbol{D}^{1/2}\boldsymbol{y}$ changes the constraint to $\boldsymbol{D}^{-1/2}\boldsymbol{z} \perp \boldsymbol{D}^{1/2}\mathbf{1}$ i.e. $\boldsymbol{z} \perp \mathbf{1}$. Thus we have that for all $\boldsymbol{z} \perp \mathbf{1}$.

$$\boldsymbol{z}^\top\boldsymbol{L}^\dagger\boldsymbol{z} \preceq 2\phi^{-2}\boldsymbol{z}^\top\boldsymbol{D}^{-1}\boldsymbol{z}.$$

Taking $\boldsymbol{z} = \boldsymbol{b}$, we then get $\boldsymbol{b}^\top\left(\boldsymbol{B}^\top\boldsymbol{B}\right)^\dagger\boldsymbol{b} \leq 2\phi^{-2}\|\boldsymbol{b}\|_{\boldsymbol{D}^{-1}}^2$. ☐

Lemma D.10. *Consider a graph G on n vertices with degrees $\boldsymbol{D}$, conductance ϕ, and edge-vertex incidence matrix $\boldsymbol{B}$, and any demand $\boldsymbol{b} \perp \mathbf{1}$. Define the electrical flow $\boldsymbol{f} = \boldsymbol{B}\left(\boldsymbol{B}^\top\boldsymbol{B}\right)^\dagger\boldsymbol{b}$. Then $\|\boldsymbol{f}\|_\infty \leq O(\phi^{-3}\log(n))\left\|\boldsymbol{D}^{-1}\boldsymbol{b}\right\|_\infty$.*

Proof. We first note that if $\boldsymbol{f}^*$ is the optimal routing of $\boldsymbol{b}$ in G, then

$$\left\|\boldsymbol{D}^{-1}\boldsymbol{b}\right\|_\infty \leq \|\boldsymbol{f}^*\|_\infty \leq \phi^{-1}\left\|\boldsymbol{D}^{-1}\boldsymbol{b}\right\|_\infty,$$

as per Example 1.4 of [She13]. Secondly, we note that by Lemma 5.2, the electrical flow $\boldsymbol{f}^{\mathcal{E}} = \boldsymbol{B}\left(\boldsymbol{B}^\top\boldsymbol{B}\right)^\dagger\boldsymbol{b} = \boldsymbol{B}\left(\boldsymbol{B}^\top\boldsymbol{B}\right)^\dagger\boldsymbol{B}^\top\boldsymbol{f}^*$ satisfies

$$\|\boldsymbol{f}^{\mathcal{E}}\|_\infty = \left\|\boldsymbol{B}\left(\boldsymbol{B}^\top\boldsymbol{B}\right)^\dagger\boldsymbol{B}^\top\boldsymbol{f}^*\right\|_\infty \leq \left\|\boldsymbol{B}\left(\boldsymbol{B}^\top\boldsymbol{B}\right)^\dagger\boldsymbol{B}^\top\right\|_{\infty\to\infty}\|\boldsymbol{f}^*\|_\infty \leq O(\phi^{-3}\log n)\left\|\boldsymbol{D}^{-1}\boldsymbol{b}\right\|_\infty.$$

☐

Lemma D.11. *On an expander G with degrees $\boldsymbol{D}$, conductance ϕ, and gradient $\boldsymbol{g}$ whose projection into the cycle space of G, $\widehat{\boldsymbol{g}}$ is α-well-spread, for any demand $\boldsymbol{b} \perp \mathbf{1}$ and dot θ with $\widehat{\boldsymbol{g}}$, the flow given by*

$$\boldsymbol{f} = \boldsymbol{B}\left(\boldsymbol{B}^\top\boldsymbol{B}\right)^\dagger\boldsymbol{b} + \frac{\theta}{\|\widehat{\boldsymbol{g}}\|_2^2}\widehat{\boldsymbol{g}} \tag{18}$$

satisfies

$$\sum_e r\boldsymbol{f}_e^2 + \sum_e s|\boldsymbol{f}_e|^p \leq$$
$$O_p\left(r \cdot \phi^{-2}\|\boldsymbol{b}\|_{\boldsymbol{D}^{-1}}^2 + r \cdot \left(\frac{|\theta|}{\|\widehat{\boldsymbol{g}}\|_2}\right)^2 + s \cdot m \cdot (\phi^{-3}\log n\left\|\boldsymbol{D}^{-1}\boldsymbol{b}\right\|_\infty)^p + s \cdot m \cdot \left(\frac{|\theta|\alpha^{1/2}}{\|\widehat{\boldsymbol{g}}\|_1}\right)^p\right).$$

Proof. We first bound the quadratic term $\sum_e r\boldsymbol{f}_e^2$. Let us write $\boldsymbol{f}^{\mathcal{E}} = \boldsymbol{B}\left(\boldsymbol{B}^\top \boldsymbol{B}\right)^\dagger \boldsymbol{b}$ and $\boldsymbol{f}^{\mathcal{C}} = \frac{\theta}{\|\widehat{\boldsymbol{g}}\|_2^2}\widehat{\boldsymbol{g}}$, and note (in fact, appealing to orthogonality would save an additional factor of 2)

$$\sum_e \boldsymbol{f}_e^2 = \sum_e (\boldsymbol{f}_e^{\mathcal{E}} + \boldsymbol{f}_e^{\mathcal{C}})^2 \leq \sum_e 2(\boldsymbol{f}_e^{\mathcal{E}})^2 + 2(\boldsymbol{f}_e^{\mathcal{C}})^2.$$

Then we observe by Lemma D.7 that $\sum_e (\boldsymbol{f}_e^{\mathcal{E}})^2 \leq \phi^{-2}\|\boldsymbol{b}\|_{\boldsymbol{D}^{-1}}^2$. Furthermore,

$$\sum_e (\boldsymbol{f}_e^{\mathcal{C}})^2 = \boldsymbol{f}^{\mathcal{C}\top}\boldsymbol{f}^{\mathcal{C}} = \left(\frac{|\theta|}{\|\widehat{\boldsymbol{g}}\|_2}\right)^2.$$

Combining these equations gives

$$\sum_e r\boldsymbol{f}_e^2 \leq 2r \cdot \phi^{-2}\|\boldsymbol{b}\|_{\boldsymbol{D}^{-1}}^2 + 2r \cdot \left(\frac{|\theta|}{\|\widehat{\boldsymbol{g}}\|_2}\right)^2.$$

We then bound the p-th power term,

$$\sum_e s|\boldsymbol{f}_e|^p = \sum_e s\left|\boldsymbol{f}_e^{\mathcal{E}} + \boldsymbol{f}_e^{\mathcal{C}}\right|^p \leq \sum_e s2^p(\left|\boldsymbol{f}_e^{\mathcal{E}}\right|^p + \left|\boldsymbol{f}_e^{\mathcal{C}}\right|^p) \leq ms2^p \cdot (\left\|\boldsymbol{f}^{\mathcal{E}}\right\|_\infty^p + \left\|\boldsymbol{f}^{\mathcal{C}}\right\|_\infty^p)$$

Now by Lemma D.10, we have $\left\|\boldsymbol{f}^{\mathcal{E}}\right\|_\infty = \left\|\boldsymbol{B}\left(\boldsymbol{B}^\top \boldsymbol{B}\right)^\dagger \boldsymbol{b}\right\|_\infty \leq O(\phi^{-3}\log n)\left\|\boldsymbol{D}^{-1}\boldsymbol{b}\right\|_\infty$ and by the α-well-spreadness of $\widehat{\boldsymbol{g}}$

$$\left\|\boldsymbol{f}^{\mathcal{C}}\right\|_\infty = \frac{|\theta|}{\|\widehat{\boldsymbol{g}}\|_2^2}\|\widehat{\boldsymbol{g}}\|_\infty \leq \frac{|\theta|}{\|\widehat{\boldsymbol{g}}\|_2^2}\left(\frac{\alpha}{m}\|\widehat{\boldsymbol{g}}\|_2^2\right)^{1/2} \leq \frac{\alpha^{1/2}|\theta|}{m^{1/2}\|\widehat{\boldsymbol{g}}\|_2} \leq \frac{\alpha^{1/2}|\theta|}{\|\widehat{\boldsymbol{g}}\|_1},$$

where in the last step we used $\|\widehat{\boldsymbol{g}}\|_1 \leq m^{1/2}\|\widehat{\boldsymbol{g}}\|_2$.

□

Proof of Theorem 4.10. Refer to the pseudo-code in Algorithm 9. We first collect the facts that we have established about the sampling procedure. In Line 3, $E^{\mathcal{H}}$ is formed from $E^{\mathcal{G}}$ by sampling each edge independently with probability τ. It follows that the expected number of edges in $E^{\mathcal{H}}$ is τm, and since $\tau > \log n/m$, a standard scalar Chernoff bound shows that $\left|E^{\mathcal{H}}\right| \leq 2\tau m$ with high probability. The parameters $r^{\mathcal{H}} = \tau \cdot r^{\mathcal{G}}$ and $s^{\mathcal{H}} = \tau^p \cdot s^{\mathcal{G}}$ are set in Line 4. By Lemma D.2, with high probability the sampling in Line 3 guarantees Equation (12). Note also that

- Equation (13) implies $\tau\boldsymbol{D}^{\mathcal{H}} \approx_{0.1} \boldsymbol{D}^{\mathcal{G}}$, by considering the quadratic form in each of the standard basis vectors.
- By Corollary D.5, $\widehat{\boldsymbol{g}}^{\mathcal{H}}$ is $O(\alpha\phi^{-6}\log^2 n)$-well-spread, and $\left\|\widehat{\boldsymbol{g}}^{\mathcal{H}}\right\|_2^2 \approx \tau\left\|\widehat{\boldsymbol{g}}^{\mathcal{G}}\right\|_2^2$ and $\left\|\widehat{\boldsymbol{g}}^{\mathcal{H}}\right\|_1 \approx_{O(\alpha\phi^{-6}\log^2 n)} \tau\left\|\widehat{\boldsymbol{g}}^{\mathcal{G}}\right\|_1$.
- As G has conductance at least ϕ, by Corollary D.4, H has conductance at least 0.8ϕ.

We can now establish $\mathcal{G} \preceq_\kappa \mathcal{H}$. Suppose $\boldsymbol{f}^{\mathcal{G}}$ is a flow in G with $\boldsymbol{B}^{\mathcal{G}}\boldsymbol{f}^{\mathcal{G}} = \boldsymbol{b}$ and $\widehat{\boldsymbol{g}}^{\mathcal{G}\top}\boldsymbol{f}^{\mathcal{G}} = \theta$. Then $\boldsymbol{g}^{\mathcal{G}\top}\boldsymbol{f}^{\mathcal{G}} = \theta + \boldsymbol{\psi}^\top \boldsymbol{b}$. By Lemma D.6, we then get that

$$\sum_e r^{\mathcal{G}}(\boldsymbol{f}_e^{\mathcal{G}})^2 + \sum_e s^{\mathcal{G}}\left|\boldsymbol{f}_e^{\mathcal{G}}\right|^p$$
$$\geq \Omega\left(r^{\mathcal{G}} \cdot \|\boldsymbol{b}\|^2_{(\boldsymbol{D}^{\mathcal{G}})^{-1}} + r^{\mathcal{G}} \cdot \frac{\theta^2}{\left\|\widehat{\boldsymbol{g}}^{\mathcal{G}}\right\|_2^2} + s^{\mathcal{G}} \cdot \left\|(\boldsymbol{D}^{\mathcal{G}})^{-1}\boldsymbol{b}\right\|_\infty^p + s^{\mathcal{G}} \cdot \left(\frac{\theta}{\left\|\widehat{\boldsymbol{g}}^{\mathcal{G}}\right\|_1}\right)^p \right).$$

Applying our flow map from $\mathcal{G}$ to $\mathcal{H}$

$$\begin{aligned}\boldsymbol{f}^{\mathcal{H}} = \mathcal{M}_{\mathcal{G}\to\mathcal{H}}(\boldsymbol{f}^{\mathcal{G}}) &= \boldsymbol{B}^{\mathcal{H}}\left(\boldsymbol{B}^{H\top}\boldsymbol{B}^{\mathcal{H}}\right)^\dagger \boldsymbol{B}^{\mathcal{G}\top}\boldsymbol{f}^{\mathcal{G}} + \frac{1}{\left\|\widehat{\boldsymbol{g}}^{\mathcal{H}}\right\|_2^2}\widehat{\boldsymbol{g}}^{\mathcal{H}}\widehat{\boldsymbol{g}}^{\mathcal{G}\top}\boldsymbol{f}^{\mathcal{G}} \\ &= \boldsymbol{B}^{\mathcal{H}}\left(\boldsymbol{B}^{H\top}\boldsymbol{B}^{\mathcal{H}}\right)^\dagger \boldsymbol{b} + \frac{1}{\left\|\widehat{\boldsymbol{g}}^{\mathcal{H}}\right\|_2^2}\widehat{\boldsymbol{g}}^{\mathcal{H}}\theta.\end{aligned}$$

We note that by construction, we can readily verify $\boldsymbol{B}^{\mathcal{H}}\boldsymbol{f}^{\mathcal{H}} = \boldsymbol{b}$, $\widehat{\boldsymbol{g}}^{H\top}\boldsymbol{f}^{\mathcal{H}} = \theta$, and $\boldsymbol{g}^{H\top}\boldsymbol{f}^{\mathcal{H}} = \theta + \boldsymbol{\psi}^\top\boldsymbol{b}$. So by applying Lemma D.11 to $\frac{1}{\kappa}\boldsymbol{f}^{\mathcal{H}}$, we get

$$\begin{aligned}&\sum_e r^{\mathcal{H}}\left(\frac{\left|\boldsymbol{f}_e^{\mathcal{H}}\right|}{\kappa}\right)^2 + \sum_e s^{\mathcal{H}}\left(\frac{\left|\boldsymbol{f}_e^{\mathcal{H}}\right|}{\kappa}\right)^p \\ &\quad\leq O_p\Bigg(r^{\mathcal{H}}\|\boldsymbol{b}\|^2_{(\boldsymbol{D}^{\mathcal{H}})^{-1}}\left(\frac{\phi^{-1}}{\kappa}\right)^2 + r^{\mathcal{H}} \cdot \left(\frac{|\theta|}{\left\|\widehat{\boldsymbol{g}}^{\mathcal{H}}\right\|_2}\right)^2\kappa^{-2} \\ &\quad + s^{\mathcal{H}} \cdot m \cdot \left(\phi^{-2}\left\|(\boldsymbol{D}^{\mathcal{H}})^{-1}\boldsymbol{b}\right\|_\infty\right)^p\kappa^{-p} + s^{\mathcal{H}} \cdot m \cdot \left(\frac{|\theta|(\alpha\phi^{-6}\log^2 n)^{1/2}}{\left\|\widehat{\boldsymbol{g}}^{\mathcal{H}}\right\|_1}\right)^p \kappa^{-p}\Bigg) \\ &\quad\leq O_p\Bigg((\tau r^{\mathcal{G}}) \cdot \frac{1}{\tau}\|\boldsymbol{b}\|^2_{(\boldsymbol{D}^{\mathcal{G}})^{-1}}\left(\frac{\phi^{-1}}{\kappa}\right)^2 + (\tau r^{\mathcal{G}}) \cdot \frac{1}{\tau} \cdot \left(\frac{|\theta|}{\left\|\widehat{\boldsymbol{g}}^{\mathcal{G}}\right\|_2}\right)^2 \kappa^{-2} + \\ &\quad (\tau^p s^{\mathcal{G}}) \cdot \tau^{-p} \cdot \left\|(\boldsymbol{D}^{\mathcal{H}})^{-1}\boldsymbol{b}\right\|_\infty^p \left(\frac{m^{1/p}\phi^{-2}}{\kappa}\right)^p + (\tau^p s^{\mathcal{G}}) \cdot \tau^{-p} \cdot \left(\frac{|\theta|}{\left\|\widehat{\boldsymbol{g}}^{\mathcal{G}}\right\|_1}\right)^p \left(\frac{m^{1/p}(\alpha\phi^{-6}\log^2 n)^{3/2}}{\kappa}\right)^p\Bigg).\end{aligned}$$

Our goal is to ensure

$$\boldsymbol{g}^{H\top}(\frac{1}{\kappa}\boldsymbol{f}^{\mathcal{H}}) - \left(\sum_e r^{\mathcal{H}}\left(\frac{\left|\boldsymbol{f}_e^{\mathcal{H}}\right|}{\kappa}\right)^2 + \sum_e s^{\mathcal{H}}\left(\frac{\left|\boldsymbol{f}_e^{\mathcal{H}}\right|}{\kappa}\right)^p\right) \leq \frac{1}{\kappa}\left(\boldsymbol{g}^{\mathcal{G}\top}\boldsymbol{f}^{\mathcal{G}} - \left(\sum_e r^{\mathcal{G}}(\boldsymbol{f}_e^{\mathcal{G}})^2 + \sum_e s^{\mathcal{G}}\left|\boldsymbol{f}_e^{\mathcal{G}}\right|^p\right)\right).$$

Because the linear terms cancel out, we can use the upper and lower bounds established above to say that this inequality holds provided the following conditions are satisfied (for a C_p which is a constant greater than 1 that depends on p):

- $C_p\left(\frac{\phi^{-1}}{\kappa}\right)^2 \leq 1/\kappa.$

- $C_p \kappa^{-2} \leq 1/\kappa.$

- $C_p\left(\frac{m^{1/p}\phi^{-2}}{\kappa}\right)^p \leq 1/\kappa.$

- $C_p\left(\frac{m^{1/p}(\alpha\phi^{-6}\log^2 n)^{3/2}}{\kappa}\right)^p \leq 1/\kappa.$

Recalling that α is a constant, it follows that there exists a constant C_p' (depending on p), s.t. all of the above conditions are satisfied, provided

$$\kappa \geq C_p' \max\left(m^{1/(p-1)}\phi^{-2p/(p-1)}, m^{1/(p-1)}\phi^{-9p/(p-1)}(\log n)^{3p/(p-1)}, \phi^{-2}\right)$$

And this in turn is implied by the stronger condition, $\kappa \geq C_p'(m^{1/(p-1)}\phi^{-9}\log^3 n)$, which is hence sufficient to ensure $\mathcal{G} \preceq_\kappa \mathcal{H}$.

We can then show $\mathcal{H} \preceq_\kappa \mathcal{G}$ with a very similar calculation. We include it for completeness. Suppose $\boldsymbol{f}^{\mathcal{H}}$ is a flow in H with $\boldsymbol{B}^{\mathcal{H}}\boldsymbol{f}^{\mathcal{H}} = \boldsymbol{b}$ and $\widehat{\boldsymbol{g}}^{H\top}\boldsymbol{f}^{\mathcal{H}} = \theta$. Then $\boldsymbol{g}^{H\top}\boldsymbol{f}^{\mathcal{H}} = \theta + \boldsymbol{\psi}^\top \boldsymbol{b}$.

By Lemma D.6, we then get that

$$\sum_e r^{\mathcal{H}}(\boldsymbol{f}_e^{\mathcal{H}})^2 + \sum_e s^{\mathcal{H}}|\boldsymbol{f}_e^{\mathcal{H}}|^p \geq$$

$$\Omega\left(r^{\mathcal{H}} \cdot \|\boldsymbol{b}\|^2_{(\boldsymbol{D}^{\mathcal{H}})^{-1}} + r^{\mathcal{H}} \cdot \frac{\theta^2}{\left\|\widehat{\boldsymbol{g}}^{\mathcal{H}}\right\|_2^2} + s^{\mathcal{H}} \cdot \left\|(\boldsymbol{D}^{\mathcal{H}})^{-1}\boldsymbol{b}\right\|_\infty^p + s^{\mathcal{H}} \cdot \left(\frac{\theta}{\left\|\widehat{\boldsymbol{g}}^{\mathcal{H}}\right\|_1}\right)^p\right). \geq$$

$$\Omega\left(r^{\mathcal{H}} \cdot \|\boldsymbol{b}\|^2_{(\boldsymbol{D}^{\mathcal{H}})^{-1}} + r^{\mathcal{H}} \cdot \frac{\theta^2}{\left\|\widehat{\boldsymbol{g}}^{\mathcal{H}}\right\|_2^2} + s^{\mathcal{H}} \cdot \left\|(\boldsymbol{D}^{\mathcal{H}})^{-1}\boldsymbol{b}\right\|_\infty^p + s^{\mathcal{H}} \cdot \left(\frac{\theta}{\left\|\widehat{\boldsymbol{g}}^{\mathcal{H}}\right\|_1}\right)^p\right).$$

Applying our flow map from $\mathcal{H}$ to $\mathcal{G}$

$$\begin{aligned}\boldsymbol{f}^{\mathcal{G}} = \mathcal{M}_{\mathcal{H}\to\mathcal{G}}(\boldsymbol{f}^{\mathcal{H}}) &= \boldsymbol{B}^{\mathcal{G}}\left(\boldsymbol{B}^{\mathcal{G}\top}\boldsymbol{B}^{\mathcal{G}}\right)^\dagger \boldsymbol{B}^{H\top}\boldsymbol{f}^{\mathcal{H}} + \frac{1}{\left\|\widehat{\boldsymbol{g}}^{\mathcal{G}}\right\|_2^2}\widehat{\boldsymbol{g}}^{\mathcal{G}}\widehat{\boldsymbol{g}}^{H\top}\boldsymbol{f}^{\mathcal{H}} \\ &= \boldsymbol{B}^{\mathcal{G}}\left(\boldsymbol{B}^{\mathcal{G}\top}\boldsymbol{B}^{\mathcal{G}}\right)^\dagger \boldsymbol{b} + \frac{1}{\left\|\widehat{\boldsymbol{g}}^{\mathcal{G}}\right\|_2^2}\widehat{\boldsymbol{g}}^{\mathcal{G}}\theta.\end{aligned}$$

Again, by construction, we have $\boldsymbol{B}^{\mathcal{G}}\boldsymbol{f}^{\mathcal{G}} = \boldsymbol{b}$, $\widehat{\boldsymbol{g}}^{\mathcal{G}\top}\boldsymbol{f}^{\mathcal{G}} = \theta$, and $\boldsymbol{g}^{\mathcal{G}\top}\boldsymbol{f}^{\mathcal{G}} = \theta + \boldsymbol{\psi}^\top\boldsymbol{b}$. So by applying

Lemma D.11 to $\frac{1}{\kappa}\boldsymbol{f}^{\mathcal{G}}$, we get

$$\sum_e r^{\mathcal{G}}\left(\frac{\left|\boldsymbol{f}_e^{\mathcal{G}}\right|}{\kappa}\right)^2+\sum_e s^{\mathcal{G}}\left(\frac{\left|\boldsymbol{f}_e^{\mathcal{G}}\right|}{\kappa}\right)^p$$

$$\leq O_p\left(r^{\mathcal{G}}\|\boldsymbol{b}\|^2_{(\boldsymbol{D}^{\mathcal{G}})^{-1}}\left(\frac{\phi^{-1}}{\kappa}\right)^2+r^{\mathcal{G}}\cdot\left(\frac{|\theta|}{\left\|\widehat{\boldsymbol{g}}^{\mathcal{G}}\right\|_2}\right)^2\kappa^{-2}\right.$$

$$\left.+s^{\mathcal{G}}\cdot m\cdot\left(\phi^{-2}\left\|(\boldsymbol{D}^{\mathcal{G}})^{-1}\boldsymbol{b}\right\|_\infty\right)^p\kappa^{-p}+s^{\mathcal{G}}\cdot m\cdot\left(\frac{|\theta|(\alpha\phi^{-6}\log^2 n)^{1/2}}{\left\|\widehat{\boldsymbol{g}}^{\mathcal{G}}\right\|_1}\right)^p\kappa^{-p}\right)$$

$$\leq O_p\left((\tau^{-1}r^{\mathcal{H}})\cdot\tau\|\boldsymbol{b}\|^2_{(\boldsymbol{D}^{\mathcal{H}})^{-1}}\left(\frac{\phi^{-1}}{\kappa}\right)^2+(\tau^{-1}r^{\mathcal{H}})\cdot\tau\cdot\left(\frac{|\theta|}{\left\|\widehat{\boldsymbol{g}}^{\mathcal{H}}\right\|_2}\right)^2\kappa^{-2}+\right.$$

$$\left.(\tau^{-p}s^{\mathcal{H}})\cdot\tau^p\cdot\left\|(\boldsymbol{D}^{\mathcal{H}})^{-1}\boldsymbol{b}\right\|_\infty^p\left(\frac{m^{1/p}\phi^{-2}}{\kappa}\right)^p+(\tau^{-p}s^{\mathcal{H}})\cdot\tau^p\cdot\left(\frac{|\theta|}{\left\|\widehat{\boldsymbol{g}}^{\mathcal{H}}\right\|_1}\right)^p\left(\frac{m^{1/p}(\alpha\phi^{-6}\log^2 n)^{3/2}}{\kappa}\right)^p\right).$$

Now, we want to guarantee

$$\boldsymbol{g}^{\mathcal{G}\top}(\frac{1}{\kappa}\boldsymbol{f}^{\mathcal{G}})-\left(\sum_e r^{\mathcal{G}}\left(\frac{\left|\boldsymbol{f}_e^{\mathcal{G}}\right|}{\kappa}\right)^2+\sum_e s^{\mathcal{G}}\left(\frac{\left|\boldsymbol{f}_e^{\mathcal{G}}\right|}{\kappa}\right)^p\right)\leq\frac{1}{\kappa}\left(\boldsymbol{g}^{H\top}\boldsymbol{f}^{\mathcal{H}}-\left(\sum_e r^{\mathcal{H}}(\boldsymbol{f}_e^{\mathcal{H}})^2+\sum_e s^{\mathcal{H}}|\boldsymbol{f}_e^{\mathcal{H}}|^p\right)\right).$$

Again the linear terms agree, and termwise verification shows that $\kappa\geq C_p'(m^{1/(p-1)}\log^3(n)\phi^{-9})$ is sufficient to give $\mathcal{H}\preceq_\kappa\mathcal{G}$.

□

E Using Approximate Projections

Finally, we need to account for the errors in computing the cycle projections $\widehat{\boldsymbol{g}}$ of the gradients $\boldsymbol{g}$. This error arise due to the use of iterative methods in Laplacian solvers used to evaluate $(\boldsymbol{B}^\top\boldsymbol{B})^\dagger$. As we only perform such projections on expanders, we can in fact use iterative methods. However, a dependence of $\log(1/\epsilon)$ in the error ϵ still remain.

We first formalize the exact form of this error. Kelner et al. [KOSZ13] showed that a Laplacian solver can converge in error proportional to that of the electrical flow. That is, for a slightly higher overhead of $O(\log(n/\epsilon))$, we can obtain a vector $\widetilde{\boldsymbol{g}}$ such that

$$\|\widehat{\boldsymbol{g}}-\widetilde{\boldsymbol{g}}\|_2\leq\epsilon\|\widehat{\boldsymbol{g}}\|_2\leq\epsilon\|\boldsymbol{g}\|_2.$$

This was also generalized to a black-box reduction between solvers for vertex solutions and flows subsequently [CKM+14]. As a result, we will work this guarantee with errors relative to $\widehat{\boldsymbol{g}}$.

For the partitioning stage, this error occurs in two places: for computing the norm of the projection, and for identifying edges with high contributions (aka. non-uniform) for removal.

For the former, a constant factor error in the norm of $\widehat{\boldsymbol{g}}$ will only lead to a constant factor increase in:

1. The uniformity of the true projected gradient,
2. The factor of decrease in the norm of the projected gradient from one step to next.

For both of these, such constant factor slowdowns can be absorbed by an increase in the thresholds, which in turn result in a higher uniformity parameter in decompositions returned. As this uniformity parameter only affects the number of edges sampled in Theorem 4.10, they only accumulate to a larger overhead in the $m^{O(1/\sqrt{p})}$ term in the overall running time.

The other invocation of projections is in the sparsification of expanders in Algorithm 9. Here the decomposition of $\boldsymbol{g}^{\mathcal{G}}$ into a circulation and potential flows is necessary for the construction of the gradient of the sampled graph, $\mathcal{H}$.

While an approximate energy minimizing circulation $\widetilde{\boldsymbol{g}}$ will not have $\boldsymbol{g} - \widetilde{\boldsymbol{g}}$ being a potential flow, we can instead perturb $\boldsymbol{g}$ slightly in this instance. Specifically, we can also compute a set of approximate potentials $\widetilde{\boldsymbol{\psi}}$ so that

$$\left\| \boldsymbol{g} - \left(\widetilde{\boldsymbol{g}} + \boldsymbol{B}\widetilde{\boldsymbol{\psi}} \right) \right\|_2 \leq \epsilon \|\boldsymbol{g}\|_2.$$

That is, we can perturb the initial $\boldsymbol{g}$ based on the result of this solve so that we have an exact decomposition of it into a circulation and a potential flow. The error of this perturbation is then incorporated in the same manner as terminating when $\|\widehat{\boldsymbol{g}}\|_2$ is too small in Case 2b of Theorem 4.9. Specifically, the additive error of this goes into the additive trailing terms of the guarantees of the ultra-sparsifier shown in Theorem 3.6.

Finally, the projection of the sampled gradient $\widetilde{\boldsymbol{g}}^{\mathcal{H}}$ into $\widehat{\boldsymbol{g}}^{\mathcal{H}}$ also carries such an error term. By picking ϵ to be in the $1/poly(n)$ range, we ensure that both the ℓ_2 and ℓ_1 norms of $\widehat{\boldsymbol{g}}^{\mathcal{H}}$ is close to their true terms. This in turn leads to constant factor errors in the lower and upper bounds on objectives give in Lemmas D.6 and D.11, and thus a constant factor increase in the overall approximation factors.

Therefore, it suffices to set ϵ in these approximate projection algorithms to be within $poly(n)$ factors of the δ by which UltraSparsify is invoked by the overall recursive preconditioning scheme. The choice of parameters in Theorem 3.7 then gives that it suffices to have $\log(1/\epsilon) \leq \widetilde{O}(1)$ in all projection steps. In other words, all the projections can be performed in time nearly-liner in the sizes of the graphs.

F ℓ_p-norm Semi-Supervised Learning on Graphs.

In this appendix, we briefly describe how to convert Problem (3), into a form that can be solved using our algorithm for smoothed p-norm flows as stated in Theorem 1.1.

Recall that formally, given a graph $G = (V, E)$ and a labelled subset of the nodes $T \subset V$ with labels $\boldsymbol{s}_T \in \mathbb{R}^T$, we can write the problem as

$$\min_{\substack{\boldsymbol{x} \in \Re^V \\ \text{s.t. } \boldsymbol{x}_T = \boldsymbol{s}_T}} \sum_{u \sim v} |\boldsymbol{x}_u - \boldsymbol{x}_v|^p.$$

Taking a Lagrange dual now results in the problem

$$\max_{\boldsymbol{f} : \left(\boldsymbol{B}^\top \boldsymbol{f}\right)_{V \setminus T} = 0} \boldsymbol{g}^\top \boldsymbol{f} - \sum_{u \sim v} |\boldsymbol{f}_{uv}|^q.$$

where $q = \frac{1}{1-1/p}$, and the gradient $\boldsymbol{g}$ is given by $\boldsymbol{g} = \boldsymbol{B}_{:,T} \boldsymbol{s}_T$. We cannot directly solve this formulation, since the net incoming flow at vertices in T is unknown. However, notice that the flow is preserved at all other vertices, so summed across all of T, the net flow must be zero. Thus if we merge all the vertices in T into one vertex, while turning edges in $T \times T$ into self-loops, the problem is now a circulation. Note that the optimal flow on each self-loop can be computed exactly. Now the resulting problem can be solved to high accuracy using Theorem 1.1. Meanwhile, mapping the flow back to the original flow, it can be shown that the optimal flow arises as a simple non-linear function of some voltages $\boldsymbol{x}$: $\boldsymbol{f}_e = (\boldsymbol{B}\boldsymbol{x})_e^q$. This means that if we have $\boldsymbol{f}$ to high enough accuracy, we can get an almost optimal set of voltages, e.g. by looking at flow along edges of a tree to compute a set of voltages $\boldsymbol{x}$ that are a $(1 + 1/\text{poly}(m))$ multiplicative accuracy solution to Problem (3). Since we call the algorithm of Theorem 1.1 using to solve a $\frac{p}{p-1}$-flow problem, where $p < 2$, the running time will be on the order of $2^{O((\frac{p}{p-1})^{3/2})} m^{1+O(\sqrt{\frac{p-1}{p}})}$. This in turn can be further simplified as $2^{O((\frac{1}{p-1})^{3/2})} m^{1+O(\sqrt{p-1})}$, since $p < 2$. For $p = 1 + \frac{1}{\sqrt{\log n}}$, this is time is bounded by $m^{1+o(1)}$.