# Flows, Submodularity, Sparsity, and Beyond: Continuous Optimization Insights into Discrete Problems

by

Kyriakos Axiotis

Diploma, National Technical University of Athens (2016)
S.M., Massachusetts Institute of Technology (2019)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2022



Author .....
Department of Electrical Engineering and Computer Science
August 25, 2022

Certified by .....
Aleksander Mądry
Cadence Design Systems Professor of Computing
Thesis Supervisor

Accepted by .....
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

**Flows, Submodularity, Sparsity, and Beyond: Continuous Optimization Insights into Discrete Problems**

by

Kyriakos Axiotis

Submitted to the Department of Electrical Engineering and Computer Science
on August 25, 2022, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

## Abstract

In this thesis we build on connections between discrete and continuous optimization. In the first part of the thesis we propose faster second-order convex optimization algorithms for classical graph algorithmic problems. Our main contribution is to show that the runtime of interior point methods is closely connected to spectral connectivity notions in the underlying graph, such as electrical conductance and effective resistance. We explore these connections along two orthogonal directions: Making manual interventions to the graph to improve connectivity, or keeping track of connectivity so as to make faster updates. These ideas lead to the first runtime improvement for the minimum cost flow problem in more than 10 years, as well as faster algorithms for problems like negative-weight shortest path and minimum cost perfect matching.

In the second part of the thesis, we investigate efficient optimization algorithms for problems relevant to machine learning that have some discrete element, such as sparse or low rank structure. We introduce a new technique, called adaptive regularization, which eliminates the sparsity performance degradation caused by $\ell_2$ projections onto structured non-convex domains, like the set of sparse vectors or low rank matrices. This leads to improving the sparsity guarantee of one of the most well known sparse optimization algorithms, IHT.

Thesis Supervisor: Aleksander Mądry
Title: Cadence Design Systems Professor of Computing

## Acknowledgments

This thesis is dedicated to the people who inspired, helped, and supported me through all these years.

To my advisor Aleksander Mądry for providing extremely valuable guidance and support for the completion of this thesis. He was one of the few people who always emphasized the importance of shooting high. This helped me concentrate on big questions and understand that meaningful research progress can only be achieved by resisting easy answers.

To my collaborator Adrian Vladu without whom this thesis would surely not have been possible. He taught me a lot of things and his inquisitive spirit led us to countless hours of learning and discovery that shaped my research philosophy.

To my collaborator Maxim Sviridenko who encouraged me to look at the more practical aspects of my work and taught me how important it is to have practical foundations for the research problems that I pick. It's hard to believe that I spent less than 3 months at Yahoo!

To my thesis committee members, Costis Daskalakis and Piotr Indyk.

To Christos Tzamos, who served as a researcher role model since my high school years and for the stimulating collaborations and discussions ever since.

To my Boston friends Christos, Vasso, Themis, Dimitris, Shibani, Giorgos, Manolis, Katerina, Ilias, Konstantina, Sophie, Chara, Dimitris.

To Dimitris, Fotis, Alex, Chenyang, Giorgos, Christos, Panagiotis, with whom we spent a unique semester at Berkeley.

To my lab and office mates Michael, Dimitris, Alex, Arturs, Nishanth, Tselil, Gautam, Andrew, Logan, Alex, Brynmor, Shibani, Quanquan.

To Dimitris Fotakis, who believed in me and fostered a unique research environment at CoReLab NTUA, and Stathis Zachos who was a source of energy and provided a sense of community.

To my friends in Athens.

To my parents Sakis and Eleni and my sister Victoria for providing an immensely supportive home environment full of love.

Finally, to my partner Eva, who has been a pillar of love and support throughout these years and someone I always know I can count on, as well as our dog Pavlito who is a constant source of joy.

# Contents

# Chapter 1

# Introduction

Optimization is a central ingredient in many disciplines, including engineering, science, economics, and operations research, with tasks such as linear programming (LP) or semi-definite programming (SDP) being among the most commonly encountered. Owing to this broad applicability of optimization and especially due to the surge in machine learning (ML) research that is being observed in the last decade, there has been tremendous work in analyzing various aspects of optimization algorithms, as well as developing new ones to tackle emergent challenges. Even though optimization algorithms are generally built to operate in a *continuous* (high-dimensional) space, a lot of practical problems naturally induce *discrete* constraints, such as ones related to selecting a subset of variables, constraints coming from graph structure, or integrality restrictions. Incorporating discrete constraints generally introduces significant challenges, and can easily make a problem depart from the realm of polynomial solvability. Yet, it has been demonstrated time and time again that insights from the continuous world are often a game-changer for solving discrete problems. Classical examples include LP relaxations for approximation algorithms and primal-dual algorithms, and more recent ones are solving approximate maximum flow with gradient descent [152, 97], or $k$-server with mirror descent [27], among many others. In this thesis, we will introduce new connections between continuous and discrete optimization for a variety of problems, including graph algorithms, submodular optimization, and sparse optimization.

### Part I: Optimization on Graphs

Traditional graph algorithms, which make use of combinatorial notions such as paths and cuts and have been extensively studied for decades, seem to be limited by long-standing efficiency barriers. On the other hand, continuous optimization has emerged as a versatile and powerful toolbox to help overcome these limitations. Casting discrete graph problems as continuous optimization tasks allows one to argue about the algorithm in the language of high-dimensional geometry, by applying the bedrock of optimization—gradient descent. This view reveals the geometric properties which affect the performance of the algorithm, effects which the algorithm designer can then try to mitigate by employing either graph-theoretic interventions, or optimization techniques like regularization. This conceptual approach has led to breakthroughs in many problems, including approximate maximum flow [39, 152, 97], minimum cost flow in dense graphs [106, 26, 25], maximum flow in sparse graphs [118, 117, 115, 95, 73], and minimum cost flow in sparse graphs [44, 42, 10, 11]. A catalyst to these results has been the groundbreaking work of Spielman and Teng [155] for solving Laplacian linear systems, which introduced a quiver of modern (spectral) graph-algorithmic ideas.

Interestingly, the flux of innovative ideas between the discrete and the continuous world goes both ways. In other words, not only does continuous optimization prove useful in the study of

graph algorithms, but conversely the study of graph algorithms also gives us deep insights into more general optimization tasks like linear programming, insights which might have been too abstract to conceive in the general setting. At the same time, graph structure is becoming increasingly relevant in machine learning.

**Minimum Cost Flow** Arguably, the problem that reigns in this area is the minimum cost flow problem, whose goal is to find the cheapest way to route a given demand in a capacitated graph. Its importance stems not only from the fact that it is ubiquitous in fields such as transportation and finance, but also because of its generality, as it contains problems like maximum flow, negative-weight shortest path, and minimum-cost bipartite matching as special cases. For many years, the state of the art runtime for sparse graphs was $\widetilde{O}\left(m^{3/2}\log^{O(1)}(U+W)\right)$, where $m$ is the number of edges in the graph and $U, W$ are upper bounds on edge capacities and costs respectively, and it resulted from an appropriate instantiation of interior point method [44]. A natural question to ask then is:

*Can we speed up interior point methods for minimum cost flow?*

In order to understand what this question entails, it is important to have some insights into how interior point methods work. Path-following interior point methods involve an iterative process, each step of which involves solving a linear system of the form $\boldsymbol{A}^\top \boldsymbol{V}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ (1), where $\boldsymbol{A}$ is the constraint matrix of the LP and $\boldsymbol{V} = \operatorname{diag}(\boldsymbol{v})$ is a diagonal matrix of positive weights, one per constraint. In particular, there are two ways in which the structure of the matrix $\boldsymbol{A}^\top \boldsymbol{V}\boldsymbol{A}$ influences the total runtime. One is, it affects how fast we are able to solve all the linear systems of the form (1) that arise throughout the algorithm, and the other one is how well-conditioned the solutions to these systems are, which in turn affects the largest value that the step size can be set to, and consequently determines the number of iterations.

When we are working with graphs, (1) is commonly a Laplacian system on the input graph with edge weights given by $\boldsymbol{v}$. Our first contribution is to relate the *electrical conductance* of this graph with the magnitude of the entries in $\boldsymbol{x}$. In particular, high conductance implies that $\boldsymbol{x}$ has small entries, which can be used to obtain an efficient algorithm for (a modified) interior point method on high-conductance graphs.

Unfortunately, guaranteeing that the conductance is always high is not an easy task, because the edge weights can change significantly throughout the algorithm. Besides, some of the weights eventually converge to 0 (this is because some constraints become tight), so it is guaranteed that some edges have arbitrarily bad conductance. In order to alleviate this issue, we modify the graph to improve its conductance. After necessary post-processing to repair these modifications, this leads to an $\widetilde{O}\left(m^{4/3}\log W\right)$ time algorithm for minimum cost flow on a graph with $m$ edges, unit capacities, and costs bounded by $W$ in absolute value. It also leads to an algorithm with the same runtime for *negative-weight shortest path* and *minimum cost bipartite* $\boldsymbol{b}$*-matching*, improving the runtimes of all these problems.

Even though the unit-capacity result makes considerable progress, it is hardly satisfying for cases when the edge capacities are large. This is because the magnitude of perturbations in the problem parameters depends polynomially on the edge capacities, and polynomial perturbations lead to a polynomial runtime overhead. To alleviate this issue, we refrain from trying to manually improve the graph's conductance, and instead focus on solving (1) faster, and in particular, in *sublinear* time. This dynamic problem can be cast more generally as maintaining the solution to a *dynamically-changing Laplacian system* in sublinear time. Interestingly, it turns out that the runtime of the updates required to dynamically solve (1) is also closely related to the magnitude of entries of the solution

vector $\boldsymbol{x}$. This leads to sublinear runtime, albeit only for unit-capacity graphs. Generalizing this idea to graphs with general capacities requires a substantially more fine-grained analysis, which reveals that the requirement for the entries of $\boldsymbol{x}$ to be small is too strong, and can instead be replaced by an electrical energy argument. This leads to a runtime of $\widetilde{O}\left(m^{3/2-\delta}\log(U+W)\right)$ for some small $\delta > 0$ for minimum cost flow, which is the first runtime improvement in more than ten years.

**Beyond flows** The question about speeding up linear programs is also interesting to be asked in a broader context. When can we efficiently solve (and how efficiently can we solve) a convex program with an exponential number of constraints? A prime example of such a problem is *submodular function minimization (SFM)*. Although still under the umbrella of convex optimization, this problem is much harder than graph problems, and the fastest known algorithm for it runs in $\widetilde{O}\left(n^3\right)$, where $n$ is the size of the ground set of the function [107]. Even though this runtime is too slow for large-scale applications, the SFM instances that arise in practice often have special structure. Specifically, the function to be minimized is often *decomposable*, i.e. can be written as the sum of much simpler (e.g. $O(1)$-sized) submodular functions. Examples include hypergraph cut with $O(1)$-sized edges, image segmentation, and MAP inference.

Algorithms for this problem have used ideas from both traditional (discrete) algorithms (e.g. augmenting paths), and continuous optimization algorithms (e.g. coordinate descent). As none of these approaches have an ideal runtime, but they have complementary strengths and weaknesses, the question arises:

*Can we bridge discrete and continuous approaches for the decomposable submodular minimization problem?*

This part of the thesis will study how to achieve this by designing a hybrid discrete-continuous optimization algorithm, which works by a combination of a preconditioned iterative method and $\widetilde{O}(1)$ calls to a maximum flow oracle. This is quite surprising, as the submodular minimization problem is not directly related to graphs. Using the recent breakthrough result of [37] which gives an almost-linear time maximum flow algorithm, this leads to an almost-linear time algorithm for decomposable submodular minimization, when all functions in the decomposition are $\widetilde{O}(1)$-sized.

## Part II: Optimization Under Sparsity Constraints

How much can we compress a machine learning model while not sacrificing predictive performance? How fast can we produce such a model and what are the types of structured sparsity constraints that we can impose? What is the tradeoff between sparsity and runtime?

Modern machine learning tasks deal with huge amounts of data. For this reason, it is important for the model that is being produced to be as concise as possible, something that is especially crucial in applications with limited processing and storage capabilities. These questions naturally arise in the field of machine learning, and are also relevant in the fields of optimization and compressed sensing. While training a machine learning model, e.g. a linear model, is a continuous optimization task, it is often useful to include additional discrete constraints. Examples include forcing an upper bound on the number of features that are used, imposing a low rank structure to the solution, or injecting relational knowledge by a pre-specified graph. Such use cases are encountered for example in ML feature engineering, neural network model compression, ML design under budget constraints, and relationship-aware models for biology.

**Sparse optimization** Sparse optimization refers to the task of minimizing a function $f(\boldsymbol{x})$ under a *sparsity* constraint $\|\boldsymbol{x}\|_0 \leq s$, which restricts the vector $\boldsymbol{x}$ to have at most $s$ non-zero entries. As is often the case, the introduction of the discrete sparsity constraint results to the problem being NP-hard. The most common way to overcome this issue is to relax the non-convex sparsity constraint to its convex relaxation, i.e. the $\ell_1$ constraint $\|\boldsymbol{x}\|_1 \leq \lambda$, making the problem amenable to convex optimization algorithms while at the same time returning a solution of relaxed sparsity $s' \geq s$. Other approaches include selecting coordinates incrementally in a greedy fashion (e.g. the orthogonal matching pursuit (OMP) algorithm [135] or Frank-Wolfe methods [69]), and performing projected gradient descent on the space of $s$-sparse solutions (e.g. the iterative hard thresholding (IHT) algorithm [19]). Yet, none of these are ideal: For convex $f$, OMP guarantees sparsity $s' \leq O\left(\kappa \log \frac{f(\mathbf{0})-f(\boldsymbol{x}^*)}{\varepsilon}\right) \cdot s$, while IHT guarantees $s' \leq O\left(\kappa^2\right) \cdot s$, where $\kappa$ is the (restricted) condition number of $f$. It is natural to ask whether we can get the best of both worlds, i.e. no dependence on the desired error tolerance $\varepsilon$, and a linear instead of quadratic dependence on the condition number.

*Is there a sparse convex optimization algorithm that guarantees sparsity $s' \leq O(\kappa) \cdot s$?*

In the second part of the thesis, we will show that this question can be answered affirmatively. In fact, by a more careful analysis, we will show that a simple modification to the IHT algorithm is enough to achieve this bound, thus essentially reducing the quadratic condition number dependence of IHT to linear, with no significant runtime overhead. This is made possible by a new technique that we call *adaptive regularization*, which avoids the sparsity performance degradation caused by projections onto structured non-convex domains, like the set of $s$-sparse vectors.

**Low rank optimization** Beyond sparsity, different measures often better capture the sparsity structure of a particular problem. For matrix problems, the most common measure of simplicity is rank, which is the sparsity of the matrix spectrum. The low rank optimization problem asks to minimize a function $f(\boldsymbol{A})$ acting on matrices, under the constraint that $\operatorname{rank}(\boldsymbol{A}) \leq r$. Matrix completion, low rank matrix approximation, and robust principal component analysis, which are central problems in machine learning and signal processing, are all captured by the low rank optimization problem by appropriate choice of $f$. Algorithms for these problems are closely connected to sparse optimization algorithms, and indeed most algorithms for the latter have analogues for the former, albeit with some additional difficulties. In this thesis, we propose simple greedy and local search algorithms for low rank convex optimization, and show their theoretical and practical advantage.

## 1.1 Structure of the Thesis

**Part I: Optimization on Graphs.** In the first part of the thesis, we examine connections between flow algorithms and convex optimization, with the goal on one hand to design and analyze faster graph algorithms that are based on convex optimization, and simultaneously on the other hand to determine how purely graph algorithmic ideas tie back to the efficiency of convex optimization algorithms, even providing insights for problems not directly related to graphs.

- In **Chapter 3**, we will develop a way to speed up the number of iterations of interior point method when applied to the *unit-capacity minimum cost flow* problem, which leads to improving the runtime from $\widetilde{O}\left(m^{10/7}\right)$ [42] to $\widetilde{O}\left(m^{4/3}\right)$. By known reductions [42], this leads

to similarly faster algorithms for negative-weight shortest paths and minimum cost $\boldsymbol{b}$-matching. This chapter is based on the paper *Circulation Control for Faster Minimum Cost Flow in Unit-Capacity Graphs* [10], which is joint work with Aleksander Mądry and Adrian Vladu.

- In **Chapter 4**, we will continue with the minimum cost flow problem, this time with general (polynomially bounded) capacities. We will show that, by closely examining structural properties of electrical flows using spectral graph theory, we can speed up the amortized runtime of each interior point method iteration by discarding a large fraction of the graph. It is based on the paper *Faster Sparse Minimum Cost Flow by Electrical Flow Localization* [11], which is joint work with Aleksander Mądry and Adrian Vladu.

- In **Chapter 5**, we will study the connection between flow problems and submodular minimization. We will show that, when the submodular function is decomposable as the sum of $\widetilde{O}(1)$-sized submodular functions, it can be minimized by $\widetilde{O}(1)$ calls to any maximum flow oracle. It is based on the paper *Decomposable Submodular Function Minimization via Maximum Flow* [9], which is joint work with Adam Karczmarz, Anish Mukherjee, Piotr Sankowski, and Adrian Vladu.

**Part II: Optimization Under Sparsity Constraints.** In the second part of the thesis, we deal with problems arising from resource-constrained machine learning and compressed sensing. We study optimization under sparsity constraints, which are discrete constraints that enforce a bound on the size of the predictor model. Our goal will be to present efficient optimization algorithms that can approach the optimal sparsity level. All the papers on which the second part of the thesis is based are joint works with Maxim Sviridenko.

- In **Chapter 6** we develop algorithms for sparse convex optimization, where the goal is to optimize a convex function under the constraint that the solution has a bounded number of non-zero entries. It will be based on a new technique which we call *adaptive regularization*, and is used to improve the sparsity of the returned solution, based on the paper *Sparse Convex Optimization via Adaptively Regularized Hard Thresholding* [14].

- Then, in **Chapter 7** we show how to alleviate the high computational overhead of the algorithm in the previous chapter, while achieving a similar sparsity bound. This is based on *Iterative Hard Thresholding with Adaptive Regularization: Sparser Solutions Without Sacrificing Runtime* [12].

- In **Chapter 8** we study convex optimization with a low rank constraint, which is related to problems like low rank approximation and robust principal components analysis. This is based on the paper *Local Search Algorithms for Rank-Constrained Convex Optimization* [13].

# Chapter 2

# Background

In this chapter, we will briefly present the background material necessary for following the rest of the thesis. This consists of commonly used notation, fundamental notions from convex analysis that will be extensively used throughout, as well as notation and fundamental theorems about graphs—one of the main objects this thesis is concerned about.

## 2.1 Basic Notation

**Vectors.** For any $k \in \mathbb{Z}_{\geq 0}$, we denote $[k] = \{1, 2, \ldots, k\}$. We will use **bold** to refer to vectors or matrices. We denote by $\mathbf{0}$ the all-zero vector, $\mathbf{1}$ the all-one vector, $\boldsymbol{O}$ the all-zero matrix, and by $\boldsymbol{I}$ the identity matrix (with dimensions understood from the context). Additionally, we will denote by $\mathbf{1}_i$ the $i$-th basis vector, i.e. the vector that is 0 everywhere except at position $i$, where it is 1.

We apply scalar operations to vectors with the interpretation that they are applied element-wise. For example, given two vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, $\boldsymbol{x}/\boldsymbol{y}$ represents the vector whose $i^{\text{th}}$ entry is $x_i/y_i$. Similarly, $\boldsymbol{x}\boldsymbol{y}$ is the element-wise product of $\boldsymbol{x}$ and $\boldsymbol{y}$, and $\boldsymbol{x}^2$ is the element-wise square of vector $\boldsymbol{x}$. We use the inline notation $(\boldsymbol{x}; \boldsymbol{y})$ to represent the concatenation of vectors $\boldsymbol{x}$ and $\boldsymbol{y}$. For any $\boldsymbol{x} \in \mathbb{R}^n$, we also denote the *support* of $\boldsymbol{x}$ by $\operatorname{supp}(\boldsymbol{x}) := \{i \ : \ x_i \neq 0\}$.

Where not ambiguous, we use the uppercase symbol corresponding to a symbol denoting a vector, to represent the diagonal matrix of that vector. In other words, given $\boldsymbol{r} \in \mathbb{R}^n$, $\boldsymbol{R} = \operatorname{diag}(\boldsymbol{r})$ is a matrix whose diagonal entries are $r_1, \ldots, r_n$ and off-diagonal entries are 0.

**Norms and Inner Products.** Given two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ of the same dimension, we use $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ or $\boldsymbol{x}^\top \boldsymbol{y}$ to denote their inner product. For any $p \in \mathbb{R}_{\geq 1}$, we denote the $\ell_p$ norm of some vector $\boldsymbol{x} \in \mathbb{R}^n$ as

$$\|\boldsymbol{x}\|_p = \left( \sum_{i=1}^{n} |x_i|^p \right)^{1/p},$$

as well as the special cases $\|\boldsymbol{x}\|_0 = |\{i \in [n] \ : \ x_i \neq 0\}|$ and $\|\boldsymbol{x}\|_\infty = \max_{i \in [n]} |x_i|$. $\|\boldsymbol{x}\|_0$ is also called the *sparsity* of $\boldsymbol{x}$. For any $\ell_p$ norm and a non-negative weight vector $\boldsymbol{w} \in \mathbb{R}^n_{\geq 0}$, we also define the *weighted $\ell_p$ norm* of a vector $\boldsymbol{x} \in \mathbb{R}^n$ as:

$$\|\boldsymbol{x}\|_{\boldsymbol{w},p} = \left( \sum_{i=1}^{n} w_i \, |x_i|^p \right)^{1/p}.$$

For two matrices $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{m \times n}$, we denote by $\langle \boldsymbol{A}, \boldsymbol{B} \rangle$ their *Frobenius inner product*, i.e. $\sum_{i=1}^{m} \sum_{j=1}^{n} A_{ij} B_{ij}$, which is also equal to the trace $\mathrm{Tr}(\boldsymbol{A}^\top \boldsymbol{B})$. We also let $\|\boldsymbol{A}\|_2$ be the *spectral norm* of $\boldsymbol{A}$, i.e. the largest singular value of $\boldsymbol{A}$, $\|\boldsymbol{A}\|_F := \sqrt{\langle \boldsymbol{A}, \boldsymbol{A} \rangle}$ be the *Frobenius norm* of $\boldsymbol{A}$, and $\|\boldsymbol{A}\|_*$ be the *nuclear norm* of $\boldsymbol{A}$, i.e. the sum of its singular values.

**Definition 2.1.1** (Dual norm)**.** *Given a norm $\|\cdot\|$, its dual norm is defined as*

$$\|\boldsymbol{y}\|_* = \max_{\|\boldsymbol{x}\| \leq 1} \langle \boldsymbol{y}, \boldsymbol{x} \rangle .$$

**Restrictions and Thresholding.** For any vector $\boldsymbol{x} \in \mathbb{R}^n$ and set $S \subseteq [n]$, we denote by $\boldsymbol{x}_S$ the vector that results from $\boldsymbol{x}$ after zeroing out all the entries except those in indices given by $S$. For any $t \in \mathbb{R}$ and $\boldsymbol{x} \in \mathbb{R}^n$, we denote by $\boldsymbol{x}_{\geq t}$ the vector that results from setting all the entries of $\boldsymbol{x}$ that are less than $t$ to 0. For a differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, its gradient $\nabla f(\boldsymbol{x})$, and a set of indices $S \subseteq [n]$, we denote $\nabla_S f(\boldsymbol{x}) := (\nabla f(\boldsymbol{x}))_S$. We also define the *thresholding operator* $H_s(\boldsymbol{x})$ to be equal to $\boldsymbol{x}_R$, where $R$ are the $s$ entries of $\boldsymbol{x}$ with largest absolute value (breaking ties arbitrarily). We override the thresholding operator $H_r(\boldsymbol{A})$ when the argument is a matrix $\boldsymbol{A}$, defining $H_r(\boldsymbol{A}) = \boldsymbol{U} \mathrm{diag}\left(H_r(\boldsymbol{\lambda})\right) \boldsymbol{V}^\top$, where $\boldsymbol{U} \mathrm{diag}(\boldsymbol{\lambda}) \boldsymbol{V}^\top$ is the singular value decomposition of $\boldsymbol{A}$. In other words, $H_r(\boldsymbol{A})$ only keeps the top $r$ singular components of $\boldsymbol{A}$.

For a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, a subset of rows $C \subseteq [m]$ and subset of columns $F \subseteq [n]$, we denote by $\boldsymbol{A}_{CF}$ the matrix that results by setting all the elements $A_{ij}$ where $i \notin C$ or $j \notin F$ to 0 and keeping the rest intact.

## 2.2 Convex Analysis

One of the useful features of using continuous optimization is that it replaces the ad hoc potentials and analytic tools of discrete algorithms by principled, generic mathematical properties and tools. For example, one can analyze algorithms for a discrete optimization problem by turning it into a continuous function in high-dimensional space, then studying fundamental analytic properties of this function, such as convexity, smoothness, and Lipschitzness. In this section, we will go over the most widely used of these properties. As we will be mostly dealing with differentiable functions throughout the thesis, we tailor our definitions for differentiable functions. Given a function $f : \mathbb{R}^n \to \mathbb{R}$, we will denote its gradient at $\boldsymbol{x}$ by $\nabla f(\boldsymbol{x}) \in \mathbb{R}^n$ and its *Hessian* at $\boldsymbol{x}$ (when defined) by $\nabla^2 f(\boldsymbol{x}) \in \mathbb{R}^{n \times n}$.

**Convexity.** The most important notion will undoubtedly be *convexity*. This is partially because it comes hand in hand with a polynomial time algorithm for minimizing such a function. In fact, when posed with an algorithmic problem about which not much is known, the principled approach is to first attempt to reduce it to minimizing a convex function. Let us get some insight into what convexity is. A set $S \subseteq \mathbb{R}^n$ is convex if it is closed under convex combinations, i.e. for any $\boldsymbol{x}, \boldsymbol{y} \in S$ and $\tau \in [0, 1]$, the solution $\boldsymbol{z} = \tau \boldsymbol{x} + (1 - \tau) \boldsymbol{y}$ that lies in the straight line between $\boldsymbol{x}$ and $\boldsymbol{y}$, also belongs to $S$. A function $f : \mathbb{R}^n \to \mathbb{R}$ is then convex if all its level sets are convex. Alternatively, if any linear Taylor approximation of $f$ lower bounds the value of $f$ everywhere, i.e. for any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ we have $f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle$.

The power of this simple property comes from the fact that it is always possible to reach an *optimal* solution by following a straight line from any *feasible* solution, while always staying feasible. What is somewhat of a mystery is why this property, which on its surface can appear quite restrictive, is so commonly encountered in practice, and even when it fails to hold, the optimization frameworks

that are built on convex foundations are still practically applicable and with surprisingly good performance.

The most basic yet important algorithm for convex functions is the *gradient descent algorithm*, which minimizes a convex function by repeatedly moving in the negative direction of the gradient: $\boldsymbol{x}' = \boldsymbol{x} - \eta \nabla f(\boldsymbol{x})$, where $\eta > 0$ is a problem-dependent (and often iteration-dependent) parameter called the *step size*.

**Lipschitzness and smoothness.** While convexity is sufficient to obtain a polynomial time algorithm, functions encountered often have more "niceness" properties. In particular, if the value or the gradient of a function change erratically while moving in the problem domain, it can be easy to *overshoot* the optimum, thus requiring the algorithm to use very small step sizes and compromising efficiency.

It is known that, as long as the function is *Lipschitz*, the (sub)gradient descent (or more generally *mirror descent*) algorithm can be analyzed to give useful convergence bounds. The Lipschitzness property is expressed by a bound on the norm of the gradient of the function. and it gives a bound on how fast the value of the function can change in the underlying norm. Specifically, $f : \mathbb{R}^n \to \mathbb{R}$ is called *L-Lipschitz* with respect to a norm $\|\cdot\|$ if for any $\boldsymbol{x} \in \mathbb{R}^n$ we have $\|\nabla f(\boldsymbol{x})\|_* \leq L$.

In some cases, it is beneficial to exploit the *smoothness* of the function, i.e. how fast the *gradient* changes in the underlying norm. Algorithmically, this gives us a predictable change in the gradient along the update step, and thus also a lower bound on the step size and a convergence analysis of gradient descent. Smoothness also gives an upper bound on the error of the first-order Taylor expansion of $f$. $f : \mathbb{R}^n \to \mathbb{R}$ is called *$\beta$-smooth* with respect to a norm $\|\cdot\|$ if for any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ we have $f(\boldsymbol{y}) \leq f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle + (\beta/2) \|\boldsymbol{y} - \boldsymbol{x}\|^2$.

**Strong convexity and condition number.** Sometimes, the rate of change of the gradient can also be *lower bounded*. Together with smoothness, this property called *strong convexity* implies that not only the gradient is correlated with the direction to the optimal solution, but is strongly correlated. Together, the smoothness and strong convexity properties lead to very favorable, linear convergence rates for gradient descent. Concretely, $f : \mathbb{R}^n \to \mathbb{R}$ is called *$\alpha$-strongly convex* with respect to a norm $\|\cdot\|$ if for any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ we have $f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle + (\alpha/2) \|\boldsymbol{y} - \boldsymbol{x}\|^2$. If $f$ is both smooth and strongly convex with respect to the $\ell_2$ norm, then we can define the *condition number* $\kappa \geq 1$ as $\beta/\alpha$, where $\beta > 0$ is the smallest constant (or infimum) such that $f$ is $\beta$-smooth and $\alpha > 0$ is the largest constant (or supremum) such that $f$ is $\alpha$-strongly convex.

**Sparse supports.** In some applications like compressed sensing, the function is only going to be well-conditioned along *sparse* directions. This leads to a generalization of all the previous definitions. Specifically, a function is called *$\beta$-smooth* at sparsity level $s \geq 1$ if for any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ with $\|\boldsymbol{y} - \boldsymbol{x}\|_0 \leq s$ we have

$$f(\boldsymbol{y}) \leq f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle + (\beta/2) \|\boldsymbol{y} - \boldsymbol{x}\|^2 .$$

We denote the smallest such constant $\beta$ by $\beta_s$ and call it the *restricted smoothness constant* (RSS constant). Completely analogously, $f$ is called *$\alpha$-strongly convex* at sparsity level $s \geq 1$ if for any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ with $\|\boldsymbol{y} - \boldsymbol{x}\|_0 \leq s$ we have

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle + (\alpha/2) \|\boldsymbol{y} - \boldsymbol{x}\|^2 .$$

We denote the largest such constant $\alpha$ by $\alpha_s$ and call it the *restricted strong convexity constant* (RSC constant). Then, the *restricted condition number* (RCN) of $f$ is denoted by $\kappa_s \geq 1$ and defined as $\beta_s/\alpha_s$. The properties defined above are also closely connected to the well-known restricted isometry property, which is commonly used in compressed sensing. We say that a function $f$ has the *restricted isometry property (RIP)* at sparsity level $s$ if $\beta_s, \alpha_s \in \mathbb{R}_{>0}$. Then, the *RIP constant* of $f$ at sparsity level $s$ is defined as $\delta_s = \frac{\kappa_s - 1}{\kappa_s + 1}$.[1]

**Projections.** The previously outlined gradient descent algorithm works for the problem of unconstrained minimization (i.e. minimizing over $\mathbb{R}^n$). If there are additional constraints, the minimization is over a subset of $\mathbb{R}^n$. This is tackled by incorporating an additional step to gradient descent, which projects the solution iterate onto the feasible set. The resulting algorithm is often called *projected gradient descent*. It is thus important to introduce notation for projections. Given a subspace $\mathcal{V} \subseteq \mathbb{R}^n$, we will denote the orthogonal ($\ell_2$) projection operator onto $\mathcal{V}$ as $\boldsymbol{\Pi}_{\mathcal{V}} \in \mathbb{R}^{n \times n}$. In particular, for any matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ we denote by $\mathrm{im}(\boldsymbol{A}) = \{\boldsymbol{A}\boldsymbol{x} \mid \boldsymbol{x} \in \mathbb{R}^n\}$ the *image* of $\boldsymbol{A}$ and by $\ker(\boldsymbol{A}) = \{\boldsymbol{x} \mid \boldsymbol{A}^\top \boldsymbol{x} = \boldsymbol{0}\}$ the *kernel* of $\boldsymbol{A}$. Therefore, $\boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{A})} = \boldsymbol{A}\left(\boldsymbol{A}^\top \boldsymbol{A}\right)^+ \boldsymbol{A}^\top$ is the orthogonal projection onto the image of $\boldsymbol{A}$ and $\boldsymbol{\Pi}_{\ker(\boldsymbol{A}^\top)} = \boldsymbol{I} - \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{A})}$ the orthogonal projection onto the kernel of $\boldsymbol{A}^\top$, where $(\cdot)^+$ denotes the Moore-Penrose matrix pseudoinverse.

## 2.3 Graphs and Flows

Graphs are significant in their own right because of their direct applications, but a perhaps even more crucial aspect of their significance comes because of their role in the context of bridging discrete and continuous optimization. They are the most simple mathematical objects that have a meaningful and non-trivial dual nature: Either as *combinatorial* objects consisting of vertices and edges connecting these, or as *algebraic* objects representing linear operators in a high-dimensional space.

**Graphs as sets of vertices and edges.** A (directed) graph $G = (V, E)$ is a mathematical object consisting of a set of vertices $V$ and a set of edges $E$. Each edge (or sometimes arc, in the case of directed graphs) $e$ of $G$ is a link that connects a certain (ordered) pair of vertices $(u, v)$ to each other. $u$ is called the *tail* of edge $e$, and $v$ is called its *head*. We denote by $E^+(u)$ the set of edges that have $u$ as their tail and $E^-(u)$ those that have $u$ as their head. When undirected, will write $e \sim v$ to denote the set of edges $e \in E$ that have $v$ as their endpoint. A *walk* in $G$ is a sequence of vertices, where each pair of successive vertices are connected by an edge, and its length is the number of vertices minus 1. A *path* is simply a walk without repeated vertices. We also use the notation $G = (V, E, \boldsymbol{c})$ to denote a *weighted* graph, where $\boldsymbol{c} \in \mathbb{R}^{|E|}$ is a cost vector and each edge $e$ is associated by a real-valued cost $c_e$. When a graph $G(V, E)$ is clear from context, we will use $n = |V|$ to denote the number of vertices and $m = |E|$ to denote the number of edges.

**Graphs as linear operators.** An alternative way to view a graph is to encode it as a matrix. One common way to do so is by the *adjacency matrix* of a graph. This matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ has one row and one column for each vertex, and a real number in the corresponding position $(u, v)$ that is equal to 0 if the edge $(u, v)$ does not exist in the graph, and 1 (or some other number if the graph is weighted) if the edge exists. One way to glimpse at the usefulness of the adjacency matrix is to raise it to a natural power $\boldsymbol{A}^k$ and notice that the entry at position $(u, v)$ of this matrix is equal to the total number of distinct walks from $u$ to $v$ with length exactly $k$. The cumbersomeness of

[1] We note that this is a more general definition than the one usually given, but it is equivalent up to a rescaling of $f$.

computing the same quantity in the combinatorial version shows the power of the dual nature of graphs. For various technical reasons, we will be instead using a related but different matrix called the *incidence matrix* $\boldsymbol{B} \in \mathbb{R}^{m \times n}$. The rows of this matrix correspond to edges and the columns to vertices, and the row corresponding to edge $e = (u, v)$ is all-zero except 1 at position $u$ and $-1$ at position $v$ (if the graph is undirected, then it can be made directed by arbitrarily orienting each edge). When applied to vectors, $\boldsymbol{B}$ turns vertex potentials into edge potential drops, and $\boldsymbol{B}^\top$ turns flows into vertex demands.

**Flows.** Given a graph $G$ we view a flow in $G$ as a vector $\boldsymbol{f} \in \mathbb{R}^m$ that assigns a value to each arc of $G$. If this value is negative we interpret it as having a flow of $|f_e|$ flowing in the direction opposite to the arc orientation. This convention is especially useful when discussing flows in undirected graphs.

We will be working with flows in $G$ that satisfy a certain *demand* $\boldsymbol{d} \in \mathbb{R}^n$ such that $\sum_u d_u = 0$. We say that a flow $\boldsymbol{f}$ satisfies or routes demand $\boldsymbol{d}$ if it satisfies the flow conservation constraints with respect to the demands. That is:

$$\sum_{e \in E^+(u)} f_e - \sum_{e \in E^-(u)} f_e = d_u, \quad \text{for all } u \in V\,, \tag{2.1}$$

or simply $\boldsymbol{B}^\top \boldsymbol{f} = \boldsymbol{d}$ by using the incidence matrix. Intuitively, these constraints enforce that the net balance of the total in-flow into vertex $u$ and the total out-flow leaving that vertex is equal to $d_u$. A flow for which the demand vector $\boldsymbol{d}$ is zero everywhere is called a *circulation*.

We say that a flow $\boldsymbol{f}$ is *feasible* (or that it respects capacities) in $G$ if it obeys the capacity constraints:

$$0 \leq f_e \leq u_e, \quad \text{for all } e \in E\,, \tag{2.2}$$

where $\boldsymbol{u} \in \mathbb{R}^m$ is a vector of arc capacities.

We can now arrive at the definition of the minimum cost flow problem, which is one of the most widely studied and general flow problems:

**Definition 2.3.1** (Minimum cost flow)**.** *Given a directed graph $G = (V, E, \boldsymbol{c})$ with demands $\boldsymbol{d} \in \mathbb{R}^n$ and capacities $\boldsymbol{u} \in \mathbb{R}^m_{>0}$, the* minimum cost flow problem *asks to compute a flow $\boldsymbol{f}$ that*

- *routes the demand: $\boldsymbol{B}^\top \boldsymbol{f} = \boldsymbol{d}$,*
- *respects the capacities: $\boldsymbol{0} \leq \boldsymbol{f} \leq \boldsymbol{u}$, and*
- *minimizes the cost: $\langle \boldsymbol{c}, \boldsymbol{f} \rangle$.*

*We will denote such an instance of minimum cost flow by the tuple $(G, \boldsymbol{c}, \boldsymbol{d}, \boldsymbol{u})$.*

If we drop the capacity constraint, we obtain the *transshipment problem*. If we also replace the demand by a unit $s - t$ demand, we obtain the *shortest path problem*. If we instead drop the cost minimization constraint and replace the demand by an (appropriately scaled) $s - t$ demand, we obtain the *maximum flow problem*. We can thus see why the above formulation is so widely studied and powerful.

## 2.4 Electrical Flows and Laplacians

**Electrical flows.** The flow problems mentioned in the previous section can be formulated as $\ell_\infty$ or $\ell_1$ flow minimization under appropriate constraints. For various reasons, it is natural and useful

to also study $\ell_2$ flow minimization. Natural, because the $\ell_2$ norm is the objective minimized by flow processes in nature, e.g. in electricity, hydraulics, and random walks. It also leads to a more robust solution. For example, penalizing the $\ell_1$ norm of the flow provides an optimal solution for computing the shortest path in a routing problem. However, when subject to real world situations like unexpected network changes or diversity requirements, it is desirable to have a more robust solution (i.e. consisting of more than a single path), even if it is not optimal. As for the usefulness, $\ell_2$ minimization can serve as a proxy for $\ell_1$ or $\ell_\infty$ minimization, thus allowing one to solve these with a potentially simpler $\ell_2$ oracle.

Concretely, given a graph $G(V, E)$ with edge resistances $\boldsymbol{r} \in \mathbb{R}^m_{>0}$, a flow $\boldsymbol{f} \in \mathbb{R}^m$ is called an *electrical flow* if it minimizes the *energy* $\sum_{e \in E} r_e f_e^2$ among all flows routing the demand of $\boldsymbol{f}$ (i.e. $\boldsymbol{d} = \boldsymbol{B}^\top \boldsymbol{f}$). Electrical flows are very closely connected to *graph Laplacians*. In particular, the electrical flow $\boldsymbol{f}$ induced by the demand $\boldsymbol{d}$ with resistances $\boldsymbol{r}$ is given by $\boldsymbol{f} = \boldsymbol{R}^{-1}\boldsymbol{B}\boldsymbol{L}^+\boldsymbol{d}$, where $\boldsymbol{L} = \boldsymbol{B}^\top \boldsymbol{R}^{-1}\boldsymbol{B}$ is the Laplacian of $G$ with resistances $\boldsymbol{r}$.

**Potential embeddings.** The dual notion to that of a flow is the *potential embedding*. In fact, the energy optimality of a flow $\boldsymbol{f}$ can be certified by a potential embedding $\boldsymbol{\phi} \in \mathbb{R}^n$ that satisfies Ohm's law for all edges $e = (u, v)$, i.e. $\phi_v - \phi_u = r_e f_e$, or just $\boldsymbol{B}\boldsymbol{\phi} = \boldsymbol{r}\boldsymbol{f}$ by using the incidence matrix. Given resistances $\boldsymbol{r}$, we will say that a potential embedding $\boldsymbol{\phi}$ *induces* the flow $\boldsymbol{f} = \frac{\boldsymbol{B}\boldsymbol{\phi}}{\boldsymbol{r}}$, and conversely, given an electrical flow, we will say that it induces the potentials that certify its energy optimality. The *energy of a potential embedding* is defined to be equal to the energy of the electrical flow induced by $\boldsymbol{\phi}$:

$$E_{\boldsymbol{r}}(\boldsymbol{\phi}) = \sum_{e \in E} \frac{(\boldsymbol{B}\boldsymbol{\phi})_e^2}{r_e}\,.$$

It should be noted that even though every potential embedding induces an electrical flow with some demand, not every flow is electrical. For example, a flow cycle can never be electrical. This can also be witnessed by the fact that potential embeddings live in $n$-dimensional space but flows live in $m$-dimensional space, so there are "more" flows than potential embeddings.

**Electrical energy statements.** For any valid demand and resistances, there exists an electrical flow that routes the demand. It useful to abstract away from the electrical flow, which is the solution to an $\ell_2$ minimization problem, and directly deal with the energy required to route a particular demand. This will be a very useful potential throughout, since it is much more stable than tracking the electrical flow.

For any valid demand ($\langle \mathbf{1}, \boldsymbol{d} \rangle = 0$) and resistances $\boldsymbol{r} > \mathbf{0}$, we denote by

$$\mathcal{E}_{\boldsymbol{r}}(\boldsymbol{d}) = \min_{\boldsymbol{\phi}:\, \boldsymbol{L}\boldsymbol{\phi} = \boldsymbol{d}} E_{\boldsymbol{r}}(\boldsymbol{\phi})$$

the total energy that is required to route the demand $\boldsymbol{d}$ with resistances $\boldsymbol{r}$, where $\boldsymbol{L} = \boldsymbol{B}^\top \boldsymbol{R}^{-1}\boldsymbol{B}$ is the Laplacian of $G$. We extend this definition for a $\boldsymbol{d}$ that is not a demand vector ($\langle \mathbf{1}, \boldsymbol{d} \rangle \neq 0$), as $\mathcal{E}_{\boldsymbol{r}}(\boldsymbol{d}) = \mathcal{E}_{\boldsymbol{r}}\left(\boldsymbol{d} - \frac{\langle \mathbf{1}, \boldsymbol{d} \rangle}{n} \cdot \mathbf{1}\right)$. We now outline various properties of energy, that show its predictability and make it a suitable quantity to be used as a potential in the analysis of algorithms.

**Fact 2.4.1** (Energy statements)**.** *Consider a graph $G(V, E)$ with resistances $\boldsymbol{r}$.*

- *For any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, we have $\sqrt{\mathcal{E}_{\boldsymbol{r}}(\boldsymbol{x} + \boldsymbol{y})} \leq \sqrt{\mathcal{E}_{\boldsymbol{r}}(\boldsymbol{x})} + \sqrt{\mathcal{E}_{\boldsymbol{r}}(\boldsymbol{y})}$.*

- *Consider a vector* $\boldsymbol{d} \in \mathbb{R}^n$*. Then,*

$$\max_{\boldsymbol{\phi}: E_{\boldsymbol{r}}(\boldsymbol{\phi}) \leq 1} \langle \boldsymbol{d}, \boldsymbol{\phi} \rangle = \sqrt{\mathcal{E}_{\boldsymbol{r}}(\boldsymbol{d})}\,.$$

- *For any resistances* $\boldsymbol{r}' \leq \alpha \boldsymbol{r}$ *for some* $\alpha \geq 1$ *and any* $\boldsymbol{d} \in \mathbb{R}^n$*, we have* $\mathcal{E}_{\boldsymbol{r}'}(\boldsymbol{d}) \leq \alpha \cdot \mathcal{E}_{\boldsymbol{r}}(\boldsymbol{d})$

*Proof.* We let $\boldsymbol{L} = \boldsymbol{B}^\top \boldsymbol{R}^{-1} \boldsymbol{B}$ be the Laplacian of $G$ with resistances $\boldsymbol{r}$. For the first one, we have $\sqrt{\mathcal{E}_{\boldsymbol{r}}(\boldsymbol{x}+\boldsymbol{y})} = \|\boldsymbol{x}+\boldsymbol{y}\|_{\boldsymbol{L}^+} \leq \|\boldsymbol{x}\|_{\boldsymbol{L}^+} + \|\boldsymbol{y}\|_{\boldsymbol{L}^+} = \sqrt{\mathcal{E}_{\boldsymbol{r}}(\boldsymbol{x})} + \sqrt{\mathcal{E}_{\boldsymbol{r}}(\boldsymbol{y})}$, where we used the triangle inequality.

The second one follows since $E_{\boldsymbol{r}}(\boldsymbol{\phi}) = \|\boldsymbol{\phi}\|_{\boldsymbol{L}}^2$ and $\mathcal{E}_{\boldsymbol{r}}(\boldsymbol{d}) = \|\boldsymbol{d}\|_{\boldsymbol{L}^+}^2$ and the norms $\|\cdot\|_{\boldsymbol{L}}$ and $\|\cdot\|_{\boldsymbol{L}^+}$ are dual.

For the third one, we note that $\left(\boldsymbol{B}^\top \boldsymbol{R}'^{-1} \boldsymbol{B}\right)^+ \preceq \alpha \cdot \left(\boldsymbol{B}^\top \boldsymbol{R}^{-1} \boldsymbol{B}\right)^+$, and so $\mathcal{E}_{\boldsymbol{r}'}(\boldsymbol{d}) = \|\boldsymbol{d}\|^2_{(\boldsymbol{B}^\top \boldsymbol{R}'^{-1} \boldsymbol{B})^+} \leq \alpha \|\boldsymbol{d}\|^2_{(\boldsymbol{B}^\top \boldsymbol{R}^{-1} \boldsymbol{B})^+} = \alpha \cdot \mathcal{E}_{\boldsymbol{r}}(\boldsymbol{d})$. □

Other than its use as a potential, we will also use energy as a *metric*. Just like in the context of shortest paths the right distance metric to use is the shortest path distance, in the context of electrical flows the right distance metric is the *effective resistance*. Roughly, effective resistance measures how close two vertices (or sets of vertices) of the graph are by how much energy is needed to transport flow between them. In contrast to shortest path distance, which only depends on the length of the shortest path between the two vertices, effective resistance also takes the number of alternative (edge-disjoint) paths into account.

**Definition 2.4.2** (Effective resistances)**.** *Consider a graph* $G(V, E)$ *with resistances* $\boldsymbol{r}$ *and any pair of vertices* $u, v \in V$*. We denote by* $R_{eff}(u, v)$ *the energy required to route* 1 *unit of flow from* $u$ *to* $v$*, i.e.* $R_{eff}(u, v) = \mathcal{E}_{\boldsymbol{r}}\left(\mathbf{1}_u - \mathbf{1}_v\right)$*. This is called the* effective resistance between $u$ and $v$*. We extend this definition to work with vertex subsets* $X, Y \subseteq V$*, such that* $R_{eff}(X, Y)$ *is the effective resistance between the vertices* $x, y$ *that result from contracting* $X$ *and* $Y$*. When used as an argument of* $R_{eff}$*, an edge* $e = (u, v) \in E$ *is treated as the vertex subset* $\{u, v\}$*.*

**Asymptotic notation.** We use $\widetilde{O}\left(t\right)$ to denote a quantity that is $O(t \log^k t)$ for some constant $k \geq 0$. When the main problem parameter is clear from context (e.g. the number of edges in a graph), we will use $\widetilde{O}\left(1\right)$ to denote a quantity that is $O(\log^k m)$ for some constant $k \geq 0$.

# Part I

# Optimization on Graphs

# Chapter 3

# Circulation Control for Faster Minimum Cost Flow in Unit-Capacity Graphs

## 3.1 Introduction

Finding the least costly way to route a demand through a network is a fundamental algorithmic primitive. Within the context of algorithmic graph theory it is captured as the *minimum cost flow problem*, in which given a graph with costs on its arcs and a set of demands on its vertices, one needs to find a flow that routes the demand while minimizing its cost. This problem has received significant attention [4] and inspired the development of new algorithmic techniques. For example, Orlin's network simplex algorithm [133] offered an explanation of the excellent behavior that the simplex method exhibits in practice when applied to flow problems. More broadly, the recent progress on algorithms for the flow problems [44, 39, 118, 105, 152, 97, 117, 136, 42, 150, 153, 115, 114] has been an instance of the general approach to graph algorithms that leverages the tools of continuous optimization, rather than classical combinatorial techniques. Also, there exist efficient reductions that enable us to leverage algorithms for the minimum cost flow problem to solve a host of other fundamental problems, including the maximum flow problem, the minimum cost bipartite matching problem, and the shortest path problem with negative weights.

### 3.1.1 Our Contributions

In this chapter, we present an $m^{4/3+o(1)} \log W$-time algorithm for the minimum cost flow problem in graphs with unit capacities, where $W$ denotes the bound on the magnitude of the arc costs. This improves upon the previously known $\widetilde{O}(m^{10/7} \log W)$ running time bound of Cohen et al. [42] and matches the running times of the recent algorithms due to Liu and Sidford [115, 114] for the unit capacity maximum flow problem.

Similarly to most of the relevant prior work, our algorithm at its core relies on an interior point method, but the variant of the interior point method we design and employ is directly attuned to the combinatorial properties of the graph. In particular, in contrast to [42], we do not rely on a reduction to the bipartite perfect $\boldsymbol{b}$-matching problem (which requires a sophisticated analysis). Instead, our algorithm operates directly in the space of circulations of the original graph.

One can also draw an analogy between the network simplex method [133] and ours. The former navigated the corners of a feasible polytope and improved an existing suboptimal solution through pushing flow around cycles. In contrast, we iteratively improve our existing suboptimal solution by augmenting it with circulations, but navigate through the strict interior of the polytope, seeking to

keep a specific condition called *centrality* satisfied. Also, while in the network simplex case, the key difficulty is in finding the right pivoting rule, our approach shifts the attention towards finding the right circulation to augment the flow with so as to maintain the centrality invariant.

A key ingredient of our approach is a custom preconditioning method, which enables us to control the flows we use to update the solution in each iteration. We derive a new way to tie the conductance of the graph to a certain guarantee on the flows computed in the preconditioned graph. This allows us to perform a better, tighter analysis of the quality of the preconditioner we use.

On a more technical level, our work provides a number of insights into the underlying interior point method. In particular, in our $m^{11/8+o(1)} \log W$-time algorithm (that we develop first), the progress steps we perform in order to reduce the duality gap of our current solution are cast as a refinement procedure, which simply attempts to correct a residual. This procedure is very similar to iterative refinement—widely used in the more restricted case of minimizing convex quadratic functions [167, 81]. Also, in contrast to the classic approach for maintaining constraint feasibility during the interior point method update step—which relies on controlling the $\ell_2$ norm of the relative updates to the slack variables—we want to perform steps for which it is only guaranteed that these relative updates are small in $\ell_\infty$ norm. To this end, we employ a custom residual correction procedure that works by re-weighting the capacity constraints. (It is worth noting that a similar procedure has been used in [115].[1])

This paves the way for the final algorithm that has the further improved running time of $m^{4/3+o(1)} \log W$. As a matter of fact, the key bottleneck to obtaining a faster algorithm using the above approach is the need to ensure that the residual error in the solution obtained after performing a step bounded in $\ell_\infty$ norm can be reduced to zero. This requires increasing the weights on the constraint barriers, and these weight increases are exactly what limits the exponent in the running time to 11/8. The step problem we need to solve, however, is well conditioned within a local $\ell_\infty$ ball around the current iterate. Therefore, being able to certify that the point returned by solving the step problem optimally lies within this local $\ell_\infty$ ball, implies that we can efficiently find it using a direct optimization subroutine. This latter observation is the key insight in the very recent preprint by Liu-Sidford [114] that enables them to improve the running time for maximum flow in graphs with unit capacity. We employ this insight in our setting in order to obtain an improved running time for unit-capacity minimum cost flow as well.

Finally, in order to guarantee that the $\ell_\infty$ norm of each progress step is indeed as small as needed, we employ a convex optimization subroutine with mixed $\ell_2$ and $\ell_p$ terms [103], instead of solving a linear system of equations in each update step as is typically done. (Such subroutine was similarly used by Liu and Sidford [115, 114], in a slightly different form.)

### 3.1.2 Previous Work

Due to the size of the existing literature on the studied problems, we focus our discussion only on the works that are the most relevant to our results and refer the reader to [74] and Section 1.2 in [42] for a more detailed discussion.

In 2013, Mądry [118] developed an algorithm that produces an optimal solution to the unit capacity maximum flow problem in $\widetilde{O}(m^{10/7})$ time and thus improves over a long standing running time barrier of $\widetilde{O}(n^{3/2})$ in the case of sparse graphs. An important characteristic of this approach was the careful tracking of an electrical energy quantity which allowed to control the step size. The underlying approach was then simplified by providing a more direct correspondence between the update steps of the interior point method and computing augmenting paths via electrical flow

[1] While our analysis aims to enforce a small $\ell_2$ norm of the residual error, [115] seek to control the $\ell_4$ norm of the congestion vector. These techniques turn out to be largely equivalent.

computations [117]. This framework has been also extended to a more general setting of unit capacity minimum cost flow [42], achieving a running time of $\widetilde{O}(m^{10/7} \log W)$, where $W$ upper bounds the largest cost of a graph edge in absolute value.

In a different context, motivated by new developments involving regression problems [52, 28, 1], Kyng et al. [103] studied the $\ell_p$ regression problems on graphs, obtaining an algorithm which runs in $m^{1+o(1)}$ time for a range of large values of $p$. This algorithm's running time was subsequently further improved by Adil and Sachdeva [2].

Liu and Sidford [115] have recently obtained an improved algorithm for the unit capacity maximum flow problem with a running time of $m^{11/8+o(1)}$. One of their key insights was that the work on $\ell_p$-regression problems enables treating energy control as a self-contained problem in each iteration of the interior point method, rather than maintaining energy as a global potential over the whole course of the algorithm, which was the case in previous work. Then, in their recent follow-up work, Liu and Sidford [114] strengthen the step problem primitive by directly optimizing a regularized log barrier function as opposed to performing a sequence of regularized Newton step. This led to a running time of $m^{4/3+o(1)}$ for the unit capacity maximum flow problem.

### 3.1.3 Organization of the Chapter

We begin with technical preliminaries in Section 3.2. In Section 3.3, we present our interior point framework specialized to minimum cost flow, and provide a basic analysis which yields an algorithm running in $\widetilde{O}(m^{3/2} \log W)$ time. We further refine our framework in Section 3.4, where we develop the key tools needed for our results, giving a faster, $m^{11/8+o(1)} \log W$-time algorithm for obtaining the solution to a slightly perturbed instance of the original minimum cost flow problem. In Section 3.5 we then show how to use existing combinatorial techniques to repair this perturbed instance. Finally, in Section 3.6, we demonstrate how to combine the framework developed in the previous sections with an insight from the recent work of Liu and Sidford [114] to achieve the final running time of $m^{4/3+o(1)} \log W$.

## 3.2 Preliminaries

The *unit capacity minimum cost flow problem* is to find a flow $\boldsymbol{f} \in \mathbb{R}^m$ that meets the *unit capacity constraints* $0 \leq f_e \leq 1$ for all $e \in E$ and routes the demand $\boldsymbol{d}$, while minimizing the *cost* $\sum_{e \in E} c_e f_e$.

**Cycle Basis.** A set of circulations $\mathcal{C}$ in $G$ is called a *cycle basis* if any circulation in $G$ can be expressed as a linear combination of circulations in $\mathcal{C}$. If $G$ is connected, the dimension of a cycle basis of $G$ is $m - n + 1$.

## 3.3 Minimum Cost Flow by Circulation Improvement

In this section we present our (customized) interior point method–based framework for solving the minimum cost flow problem, setting the foundations for the faster algorithm of Section 3.4.

### 3.3.1 LP Formulation and Interior Point Method

We first cast the minimum cost flow problem as a linear program that we then proceed to solve using an interior point method.

**LP formulation.** It will be useful to consider the parametrization of a flow in terms of the circulation space of the graph. The goal of this re-parametrization is to initialize the interior point method with an initial flow $\boldsymbol{f}_0$ which routes the prescribed demand $\boldsymbol{d}$, then iteratively improve it by adding circulations to get a flow which routes the same demand $\boldsymbol{d}$ but has lower duality gap. It is noteworthy that the specific parametrization of the circulation space is irrelevant to the interior point method, due to its affine invariance. We will elaborate on this point later. For us it will be a useful tool for understanding the centrality condition arising from the interior point method and applying more aggressive progress steps.

Given the (connected) underlying graph $G = (V, E)$, let $\boldsymbol{C} \in \mathbb{R}^{m \times (m-n+1)}$ be a matrix whose columns encode the characteristic vectors of a basis for $G$'s circulation space.

In order to construct such a matrix, we let $C_1, C_2, ..., C_{m-n+1}$ be an arbitrary cycle basis for $G$, where we ignore the arc orientations. An easy way to produce one is to consider a spanning tree $T \subseteq G$. For each arc $(u, v) \in E$ which is not in $T$, consider the unique path in $T$ connecting $v$ and $u$. The arcs on this path along with the arc $(u, v)$ determine a cycle in the basis. More specifically, consider the set of arcs of $G$ present in $C_i$, sorted according to the order in which they are visited along the cycle, starting with the off-tree arc $(u, v)$, then continuing with those witnessed along the tree path from $v$ to $u$. If an arc $e \in E$ has the opposite orientation to the one corresponding to the traversal of the cycle, we represent it as $\bar{e}$, otherwise we write it just as $e$.

Now, letting $C_i$ consist of a subset of arcs in $E$, each of which appears either with its original orientation $e$, or the opposite orientation $\bar{e}$, we write the $i^{th}$ column of matrix $\boldsymbol{C}$ as follows.

$$\boldsymbol{C}_{e,i} = \begin{cases} 1\,, & \text{if } e \in C_i\,, \\ -1\,, & \text{if } \bar{e} \in C_i\,, \\ 0\,, & \text{otherwise.} \end{cases}$$

We can now use $\boldsymbol{C}$ to represent any circulation in $G$. Given any $\boldsymbol{x} \in \mathbb{R}^{m-n+1}$ we have that $\boldsymbol{f} = \boldsymbol{C}\boldsymbol{x}$ is a circulation. Furthermore the sign of each coordinate $f_e$, $e = (u, v) \in E$, shows whether $\boldsymbol{f}_e$ is a flow that runs in the same direction as $e$ or vice-versa, i.e. $f_e > 0$ if $\boldsymbol{f}$ carries flow from $u$ to $v$, and similarly $f_e < 0$ if $\boldsymbol{f}$ carries flow from $v$ to $u$. On the other hand, for any circulation $\boldsymbol{f} \in \mathbb{R}^m$ there exists an $\boldsymbol{x} \in \mathbb{R}^{m-n+1}$ such that $\boldsymbol{f} = \boldsymbol{C}\boldsymbol{x}$ (or in other words the image of $\boldsymbol{C}$ is the space of circulations).

Now let $\boldsymbol{f}_0$ be a flow in $G$ such that for each arc $e \in E$, $0 < (\boldsymbol{f}_0)_e < 1$, and $\boldsymbol{f}_0$ routes the demand $\boldsymbol{d}$. The minimum cost flow problem can be cast as the following linear program:

$$\begin{aligned} &\min \langle \boldsymbol{c}, \boldsymbol{C}\boldsymbol{x} \rangle \\ &\boldsymbol{0} \leq \boldsymbol{f}_0 + \boldsymbol{C}\boldsymbol{x} \leq \boldsymbol{1}\,. \end{aligned} \tag{3.1}$$

We see that the objective value of this linear program differs by a term of $\langle \boldsymbol{c}, \boldsymbol{f}_0 \rangle$ from the original objective. We did not include it here, since it is a constant. It is useful to also consider its dual:

$$\begin{aligned} &\max -\langle \boldsymbol{1} - \boldsymbol{f}_0, \boldsymbol{y}^+ \rangle - \langle \boldsymbol{f}_0, \boldsymbol{y}^- \rangle \\ &\boldsymbol{C}^\top \left( \boldsymbol{y}^+ - \boldsymbol{y}^- \right) = -\boldsymbol{C}^\top c \\ &\boldsymbol{y}^+, \boldsymbol{y}^- \geq \boldsymbol{0}\,. \end{aligned} \tag{3.2}$$

The objective we are left to solve simply suggests that in order to find the minimum cost flow in the graph with unit capacities, we equivalently have to find the minimum cost circulation in the residual graph under shifted capacity constraints. This carries a significant similarity with the *network simplex* algorithm [133], which has been used in the past as a specialization of the

simplex method to the minimum cost flow problem. It essentially consisted of maintaining a solution routing the prescribed demand $\boldsymbol{d}$, and iteratively improving it by pushing flow around a cycle, while satisfying capacity constraints. Rather than performing such updates, which always maintain a flow on the boundary of polytope corresponding to the set of feasible solutions, the interior point method maintains a more sophisticated condition on these intermediate solutions. Another similar approach can be found in [98], where updates are iteratively pushed around cycles in order to solve Laplacian linear systems.

Like these methods, our approach will be to repeatedly improve the cost of the solution by pushing augmenting circulations. Crucially, maintaining a solution centrality condition, stemming from interior point methods, will allow us to make significant progress during each augmentation step. Figure 3-1 is the high-level procedure for this algorithm. It consists of an initialization procedure Initialize which is described in Section 3.3.3, repeated circulation augmentations using procedure Augment as described in Section 3.3.4, and finally a standard procedure Repair to round the returned solution with low duality gap to an optimal integral one (described in Section 3.5). The faster algorithm of Section 3.4 will also follow the same format, but the Augment and Repair routines will be more sophisticated.

MinCostFlow$(G, \boldsymbol{c}, \boldsymbol{d}; \varepsilon)$

1. $G', \boldsymbol{c}', \boldsymbol{w}, \boldsymbol{f}, \mu \leftarrow$ Initialize$(G, \boldsymbol{c}, \boldsymbol{d})$.
2. While $\mu m > \varepsilon$:
3. $\quad \boldsymbol{w}, \boldsymbol{f}, \mu \leftarrow$ Augment$(G', \boldsymbol{c}', \varepsilon; \boldsymbol{w}, \boldsymbol{f}, \mu)$.
4. $\boldsymbol{f} \leftarrow$ Repair$(G', \boldsymbol{c}', \boldsymbol{d}; \boldsymbol{f})$.
5. Return $\boldsymbol{f}$.

Figure 3-1: Minimum Cost Flow by Circulation Improvement

**Barrier Formulation.** In order to apply an interior point method on (3.1), we need to replace the feasibility constraints by a convex barrier function. We seek a nearly optimal solution, i.e. one that has small duality gap. The vanilla interior point method consists of iteratively finding the optimizer $\boldsymbol{x}_\mu$ for a family of functions parametrized by $\mu > 0$

$$\min_{\boldsymbol{x} \in \mathbb{R}^{m-n+1}} F_\mu^{\boldsymbol{w}}(\boldsymbol{x}) = \frac{1}{\mu} \cdot \langle \boldsymbol{c}, \boldsymbol{C}\boldsymbol{x} \rangle - \sum_{e \in E} \left( w_e^+ \cdot \log(\mathbf{1} - \boldsymbol{f}_0 - \boldsymbol{C}\boldsymbol{x})_e + w_e^- \cdot \log(\boldsymbol{f}_0 + \boldsymbol{C}\boldsymbol{x})_e \right) . \tag{3.3}$$

where $w_e^+, w_e^- > 0$ are weights on the flow capacity constraints. In order to find the optimizer $\boldsymbol{x}_\mu$, one performs Newton method on $F_\mu^{\boldsymbol{w}}$, after warm starting with $\boldsymbol{x}_{\mu(1+\delta)}$ for some $\delta > 0$.

While classical methods maintain $\boldsymbol{w} = \mathbf{1}$ at all times, this extra parameter has been introduced in previous work in order to allow the method to make progress more aggressively. To simplify notation we define the slack vector $\boldsymbol{s} = (\boldsymbol{s}^+; \boldsymbol{s}^-)$ as

$$\boldsymbol{s}^+ = \mathbf{1} - \boldsymbol{f}_0 - \boldsymbol{C}\boldsymbol{x} , \tag{3.4}$$

$$\boldsymbol{s}^- = \boldsymbol{f}_0 + \boldsymbol{C}\boldsymbol{x} , \tag{3.5}$$

representing the upper slack (i.e. the distance of the current flow $\boldsymbol{f} = \boldsymbol{f}_0 + \boldsymbol{C}\boldsymbol{x}$ to the upper capacity constraint of $\boldsymbol{f} \leq \mathbf{1}$) and the lower slack (i.e. the distance from the current flow to the lower capacity constraint $\mathbf{0} \leq \boldsymbol{f}$). We will use the vector $\boldsymbol{w} = (\boldsymbol{w}^+; \boldsymbol{w}^-)$ to represent the weights for the two sets of barriers that we are using.

### 3.3.2 Optimality and Duality Gap

In order to describe the method and analyze it, it is important to understand the optimality conditions for $F_\mu^{\boldsymbol{w}}$. We say that a vector $\boldsymbol{x}$ which minimizes $F_\mu^{\boldsymbol{w}}$ is *central* (or satisfies *centrality*). This condition is described in the following lemma, whose proof can be found in Appendix 9.1.5.

**Lemma 3.3.1.** *The vector $\boldsymbol{x}$ is a minimizer for $F_\mu^{\boldsymbol{w}}$ if and only if*

$$\boldsymbol{C}^\top \left( \frac{\boldsymbol{w}^+}{\boldsymbol{s}^+} - \frac{\boldsymbol{w}^-}{\boldsymbol{s}^-} \right) = -\frac{\boldsymbol{C}^\top \boldsymbol{c}}{\mu}. \tag{3.6}$$

*Furthermore the vector $\boldsymbol{y} = (\boldsymbol{y}^+; \boldsymbol{y}^-)$ with $\boldsymbol{y}^+ = \mu \cdot \frac{\boldsymbol{w}^+}{\boldsymbol{s}^+}$, $\boldsymbol{y}^- = \mu \cdot \frac{\boldsymbol{w}^-}{\boldsymbol{s}^-}$ is a feasible dual vector, and the duality gap of the primal-dual solution $(\boldsymbol{x}, \boldsymbol{y})$ is exactly $\mu \|\boldsymbol{w}\|_1$.*

Maintaining the centrality condition (3.6) will be the key challenge in obtaining a faster interior point method for this linear program. This emphasizes the fact that the aim of this method is to construct a feasible set of slacks $\boldsymbol{s}^+ = \mathbf{1} - \boldsymbol{f}_0 - \boldsymbol{C}\boldsymbol{x} > \mathbf{0}$ and $\boldsymbol{s}^- = \boldsymbol{f}_0 + \boldsymbol{C}\boldsymbol{x} > \mathbf{0}$ such that $\boldsymbol{C}^\top \left( \frac{\boldsymbol{w}^+}{\boldsymbol{s}^+} - \frac{\boldsymbol{w}^-}{\boldsymbol{s}^-} \right) = -\frac{\boldsymbol{C}^\top \boldsymbol{c}}{\mu}$ for a very small $\mu > 0$. It is important to note that even though the existence of such an $\boldsymbol{x}$ needs to be guaranteed, it is not necessary to explicitly maintain it. This will be apparent in the definition below.

**Definition 3.3.2** ($\mu$-central flow)**.** *Given weights $\boldsymbol{w} = (\boldsymbol{w}^+; \boldsymbol{w}^-) \in \mathbb{R}^{2m}_{>0}$, a flow $\mathbf{0} < \boldsymbol{f} < \mathbf{1}$ is called* $\mu$-central *with respect to $\boldsymbol{w}$ if for some cycle basis matrix $\boldsymbol{C} \in \mathbb{R}^{m \times (m-n+1)}$,*

$$\boldsymbol{C}^\top \left( \frac{\boldsymbol{w}^+}{\mathbf{1} - \boldsymbol{f}} - \frac{\boldsymbol{w}^-}{\boldsymbol{f}} \right) = -\frac{\boldsymbol{C}^\top \boldsymbol{c}}{\mu} \tag{3.7}$$

*for some $\mu > 0$. We will call the parameter $\mu$ the* centrality *of $\boldsymbol{f}$ with respect to $\boldsymbol{w}$.*

It should be noted that the precise choice of cycle basis $\boldsymbol{C}$ in the above definition is irrelevant, as the property is invariant under the choice of cycle basis.

### 3.3.3 Initialization

The initialization procedure description and analysis is standard and thus deferred to Section 9.1.1. From now on we assume that $G$ is the graph produced by the procedure in Section 9.1.1, together with a $\mu$-central flow with $\mu \leq 2\|\boldsymbol{c}\|_2$.

### 3.3.4 The Augmentation Procedure and Routing the Residual

The AUGMENT procedure can be seen in Figure 3-2. It consists of computing an augmenting flow $\widetilde{\boldsymbol{f}}$ by solving a linear system, augmenting the current solution by that flow, and finally calling a correction procedure in order to enforce the centrality of the solution.

We think of the interior point method as iteratively augmenting a feasible flow $\mathbf{0} < \boldsymbol{f} < \mathbf{1}$ satisfying $\boldsymbol{C}^\top \left( \frac{\boldsymbol{w}^+}{\mathbf{1}-\boldsymbol{f}} - \frac{\boldsymbol{w}^-}{\boldsymbol{f}} \right) = -\frac{\boldsymbol{C}^\top \boldsymbol{c}}{\mu}$ into another flow $\boldsymbol{f}'$ such that $\boldsymbol{C}^\top \left( \frac{\boldsymbol{w}^+}{\mathbf{1}-\boldsymbol{f}'} - \frac{\boldsymbol{w}^-}{\boldsymbol{f}'} \right) = -\frac{\boldsymbol{C}^\top \boldsymbol{c}}{\mu'}$

AUGMENT($G, \boldsymbol{c}, \varepsilon; \boldsymbol{w}, \boldsymbol{f}, \mu$)

- Given $\boldsymbol{f}$ : $\mu$-central flow with respect to weights $\boldsymbol{w}$.
- Returns $\boldsymbol{f}''$ : $\mu'$-central flow with respect to weights $\boldsymbol{w}'$.

1. Let $\widetilde{\boldsymbol{f}}$ be a solution to (3.10), where
$$\boldsymbol{h} = \delta \left( \frac{\boldsymbol{w}^+}{\boldsymbol{1} - \boldsymbol{f}} - \frac{\boldsymbol{w}^-}{\boldsymbol{f}} \right).$$
2. Compute the congestion vector $\boldsymbol{\rho} = (\boldsymbol{\rho}^+; \boldsymbol{\rho}^-)$ as $\rho_e^+ = \frac{\widetilde{f}_e}{1 - f_e}$, $\rho_e^- = \frac{-\widetilde{f}_e}{f_e}$, cf. (3.11).
3. Augment flow $\boldsymbol{f}' = \boldsymbol{f} + \widetilde{\boldsymbol{f}}$.
4. Correct residual given by $\boldsymbol{h}' = -\left( \frac{\boldsymbol{w}^+}{\boldsymbol{1} - \boldsymbol{f}'} - \frac{\boldsymbol{w}^-}{\boldsymbol{f}'} + \frac{\boldsymbol{c}}{\mu/(1+\delta)} \right)$ using Lemma 3.3.9 and get new weights $\boldsymbol{w}'$, flow $\boldsymbol{f}''$, and centrality parameter $\mu'$.
5. Return $\boldsymbol{w}', \boldsymbol{f}'', \mu'$.

Figure 3-2: Circulation improvement step

where $\mu' = \mu/(1+\delta)$ for some $\delta > 0$. The magnitude of $\delta$ for which we are able to do so dictates the rate of the convergence of the method, since in $O(\delta^{-1})$ such iterative steps, the parameter $\mu$ is reduced to half, and hence the duality gap also reduces by a factor of $1/2$, per Lemma 3.3.1.

We can interpret such an iterative step as a residual-fixing procedure. Given a flow $\boldsymbol{f}$ with slacks $\boldsymbol{s}^+ = \boldsymbol{1} - \boldsymbol{f} > \boldsymbol{0}, \boldsymbol{s}^- = \boldsymbol{f} > \boldsymbol{0}$, we consider the residual

$$\boldsymbol{C}^\top \left( \frac{\boldsymbol{w}^+}{\boldsymbol{s}^+} - \frac{\boldsymbol{w}^-}{\boldsymbol{s}^-} + \frac{\boldsymbol{c}}{\mu'} \right) = \nabla F_{\mu'}^{\boldsymbol{w}}(\boldsymbol{x}),$$

which is exactly the amount by which the target condition (3.6) fails to be satisfied. We denote the current residual as $\nabla F_{\mu}^{\boldsymbol{w}}(\boldsymbol{x}) = -\boldsymbol{C}^\top \boldsymbol{h}$ for some $\boldsymbol{h} \in \mathbb{R}^m$. The goal of the iterative step is therefore to provide a feasible update rule to the flow such that the residual shrinks in some metric. To do so, we must first define a few useful notions.

**Definition 3.3.3** (Energy of the residual)**.** *Given a residual $-\boldsymbol{C}^\top \boldsymbol{h}$ for some $\boldsymbol{h} \in \mathbb{R}^m$ and positive vectors $\boldsymbol{w} = (\boldsymbol{w}^+; \boldsymbol{w}^-) \in \mathbb{R}^{2m}, \boldsymbol{s} = (\boldsymbol{s}^+; \boldsymbol{s}^-) \in \mathbb{R}^{2m}$ we define the energy to route $\boldsymbol{h}$ with resistances determined by $(\boldsymbol{w}, \boldsymbol{s})$ as*

$$\mathcal{E}_{\boldsymbol{w},\boldsymbol{s}}(\boldsymbol{h}) = \min_{\widetilde{\boldsymbol{y}}: \boldsymbol{C}^\top(\widetilde{\boldsymbol{y}} + \boldsymbol{h}) = \boldsymbol{0}} \frac{1}{2} \sum_{e \in E} \left( \frac{w_e^+}{(s_e^+)^2} + \frac{w_e^-}{(s_e^-)^2} \right)^{-1} (\widetilde{y}_e)^2. \tag{3.8}$$

The following lemma gives equivalent formulations for the energy which will be useful for our analysis. Its proof can be found in Appendix 9.1.5.

**Lemma 3.3.4** (Equivalent energy formulations)**.** *Given $\boldsymbol{h} \in \mathbb{R}^m$ and $\boldsymbol{w}, \boldsymbol{s} \in \mathbb{R}^{2m}$, one can write:*

$$\mathcal{E}_{\boldsymbol{w},\boldsymbol{s}}(\boldsymbol{h}) = \max_{\widetilde{\boldsymbol{f}}=\boldsymbol{C}\widetilde{\boldsymbol{x}}} \langle \boldsymbol{h}, \widetilde{\boldsymbol{f}} \rangle - \frac{1}{2} \sum_{e \in E} \left( \frac{w_e^+}{(s_e^+)^2} + \frac{w_e^-}{(s_e^-)^2} \right) (\widetilde{f}_e)^2 \,. \tag{3.9}$$

*Equivalently, the following conditions are satisfied:*

$$\widetilde{\boldsymbol{f}} = \boldsymbol{C}\widetilde{\boldsymbol{x}} \,, \tag{3.10}$$
$$\boldsymbol{C}^\top \left( \frac{\boldsymbol{w}^+}{(\boldsymbol{s}^+)^2} + \frac{\boldsymbol{w}^-}{(\boldsymbol{s}^-)^2} \right) \widetilde{\boldsymbol{f}} = \boldsymbol{C}^\top \boldsymbol{h} \,.$$

*The latter equality can be re-stated in terms of the congestion vector*

$$\boldsymbol{\rho} := \left( \boldsymbol{\rho}^+ = \frac{\widetilde{\boldsymbol{f}}}{\boldsymbol{s}^+}; \boldsymbol{\rho}^- = \frac{-\widetilde{\boldsymbol{f}}}{\boldsymbol{s}^-} \right) \tag{3.11}$$

*as*

$$\boldsymbol{C}^\top \left( \frac{\boldsymbol{w}^+ \boldsymbol{\rho}^+}{\boldsymbol{s}^+} - \frac{\boldsymbol{w}^- \boldsymbol{\rho}^-}{\boldsymbol{s}^-} \right) = \boldsymbol{C}^\top \boldsymbol{h} \,. \tag{3.12}$$

*The energy can then be written as*

$$\mathcal{E}_{\boldsymbol{w},\boldsymbol{s}}(\boldsymbol{h}) = \frac{1}{2} \sum_{e \in E} \left( w_e^+ (\rho_e^+)^2 + w_e^- (\rho_e^-)^2 \right) = \frac{1}{2} \sum_{e \in E} \left( \frac{w_e^+}{(s_e^+)^2} + \frac{w_e^-}{(s_e^-)^2} \right) (\widetilde{f}_e)^2 \,.$$

Using Lemma 3.3.4 we can prove that under certain conditions we can update the flow while simultaneously maintaining its feasibility and reducing the residual. Let us first define a residual correction step.

**Definition 3.3.5** (Residual correction)**.** *Let $\boldsymbol{w}, \boldsymbol{s} \in \mathbb{R}^{2m}$ where $\boldsymbol{w} > \boldsymbol{0}$ and $\boldsymbol{f} = \boldsymbol{f}_0 + \boldsymbol{C}\boldsymbol{x}$ is a feasible flow with slacks $\boldsymbol{s}^+ = \boldsymbol{1} - \boldsymbol{f} > 0$, $\boldsymbol{s}^- = \boldsymbol{f} > 0$, and let $\nabla F_\mu^{\boldsymbol{w}}(\boldsymbol{x}) = -\boldsymbol{C}^\top \boldsymbol{h}$ be the corresponding residual. A residual correction step is defined as an update to the $\boldsymbol{x}$ vector, and implicitly to the flow vector $\boldsymbol{f}$ via:*

$$\boldsymbol{x}' = \boldsymbol{x} + \widetilde{\boldsymbol{x}} \,, \tag{3.13}$$
$$\boldsymbol{f}' = \boldsymbol{f} + \boldsymbol{C}\widetilde{\boldsymbol{x}} = \boldsymbol{f} + \widetilde{\boldsymbol{f}} \,, \tag{3.14}$$

*where $\widetilde{\boldsymbol{x}}$ is the solution to the linear system*

$$\widetilde{\boldsymbol{f}} = \boldsymbol{C}\widetilde{\boldsymbol{x}} \,, \tag{3.15}$$
$$\boldsymbol{C}^\top \left( \frac{\boldsymbol{w}^+}{(\boldsymbol{s}^+)^2} + \frac{\boldsymbol{w}^-}{(\boldsymbol{s}^-)^2} \right) \widetilde{\boldsymbol{f}} = \boldsymbol{C}^\top \boldsymbol{h} \,. \tag{3.16}$$

Since $\widetilde{\boldsymbol{f}}$ is a circulation, this shows that the residual correction steps of the vanilla interior point method preserve the demand $\boldsymbol{d}$ by adding an augmenting circulation to the current flow $\boldsymbol{f}$. It is also important to ensure that such updates do not break the LP feasibility constraints, i.e. $\boldsymbol{0} \leq \boldsymbol{f} \leq \boldsymbol{1}$ at all times. This will be made true by appropriately scaling the residual, thus enforcing $\|\boldsymbol{\rho}\|_\infty < 1/4$, or equivalently $\left| \widetilde{f}_e \right| < \frac{1}{4} \min \{1 - f_e, f_e\}$ for all $e \in E$.

It is worth noting that the flow $\widetilde{\boldsymbol{f}}$ which corresponds to solving the linear system from (3.13-3.14) can be computed in $\widetilde{O}(m)$ time using a fast Laplacian solver [155, 98, 41, 137]. This may not be immediately obvious given the cycle basis formulation. But, in fact, this follows very easily from writing the linear system solve as a convex quadratic minimization problem. We do not go into further detail here, since we will elaborate more on this topic in Section 3.4.

In order to analyze the algorithm, we use energy as a potential function.

**Lemma 3.3.6** (Energy after residual correction)**.** *Let $\boldsymbol{w} \in \mathbb{R}^{2m}$ where $\boldsymbol{w} > \mathbf{0}$, and $\boldsymbol{f} = \boldsymbol{f}_0 + \boldsymbol{C}\boldsymbol{x}$ be a flow vector with slacks $\boldsymbol{s} > \mathbf{0}$. Let $\boldsymbol{c} \in \mathbb{R}^m$ and the residual $-\boldsymbol{C}^\top \boldsymbol{h} = \nabla F_\mu^{\boldsymbol{w}}(\boldsymbol{x}) = \boldsymbol{C}^\top \left( \frac{\boldsymbol{w}^+}{\boldsymbol{s}^+} - \frac{\boldsymbol{w}^-}{\boldsymbol{s}^-} + \frac{\boldsymbol{c}}{\mu} \right)$ for some $\mu > 0$ and $\boldsymbol{\rho} \in \mathbb{R}^{2m}$ be the corresponding congestion vector. Then a residual correction step produces a new flow $\boldsymbol{f}' = \boldsymbol{f}_0 + \boldsymbol{C}\boldsymbol{x}'$ with slacks $\boldsymbol{s}'$ and residual $-\boldsymbol{C}^\top \boldsymbol{h}' = \nabla F_\mu^{\boldsymbol{w}}(\boldsymbol{x}') = \boldsymbol{C}^\top \left( \frac{\boldsymbol{w}^+}{(\boldsymbol{s}^+)'} - \frac{\boldsymbol{w}^-}{(\boldsymbol{s}^-)'} + \frac{\boldsymbol{c}}{\mu} \right)$ such that*

$$\mathcal{E}_{\boldsymbol{w},\boldsymbol{s}'}(\boldsymbol{h}') \leq \frac{1}{2} \sum_{e \in E} \left( w_e^+ (\rho_e^+)^4 + w_e^- (\rho_e^-)^4 \right) .$$

The proof can be found in Appendix 9.1.5. As a corollary, we show that if the energy required to route the residual is small to begin with, after performing a step from Lemma 3.3.6, it quickly contracts.

**Corollary 3.3.7.** *Let $\boldsymbol{w} \in \mathbb{R}^{2m}$ where $\boldsymbol{w} \geq \mathbf{1}$, and $\boldsymbol{f} = \boldsymbol{f}_0 + \boldsymbol{C}\boldsymbol{x}$ be a flow vector with slacks $\boldsymbol{s} > \mathbf{0}$ and residual $\nabla F_\mu^{\boldsymbol{w}}(\boldsymbol{x}) = -\boldsymbol{C}^\top \boldsymbol{h}$. Updating $\boldsymbol{x}$ and $\boldsymbol{f}$ via a residual correction step as in Lemma 3.3.6 yields a new flow $\boldsymbol{f}' = \boldsymbol{f}_0 + \boldsymbol{C}\boldsymbol{x}'$ with slacks $\boldsymbol{s}' > \mathbf{0}$ and residual $\nabla F_\mu^{\boldsymbol{w}}(\boldsymbol{x}') = -\boldsymbol{C}^\top \boldsymbol{h}'$ such that*

$$\mathcal{E}_{\boldsymbol{w},\boldsymbol{s}'}(\boldsymbol{h}') \leq 2 \cdot \mathcal{E}_{\boldsymbol{w},\boldsymbol{s}}(\boldsymbol{h})^2 .$$

We defer the proof to Appendix 9.1.5.

Corollary 3.3.7 shows that with a good initialization, residual correction steps decrease the energy of the residual very fast. In other words, starting from a $\mu$-central flow $\boldsymbol{f}$ one can quickly obtain a $\mu'$-central flow $\boldsymbol{f}'$ with a smaller duality gap, i.e. with $\mu' < \mu$. This is the workhorse of the vanilla interior point method that we proceed to analyze in Section 3.3.5.

Before that, we introduce an additional residual correction step that ensures that our solution is always *exactly* central. Since residual correction reduces the residual very fast, i.e. energy gets reduced to $\varepsilon$ in $O(\log \log \varepsilon^{-1})$ steps, we can intuitively think of it as a method which effectively removes the residual in $\widetilde{O}(1)$ steps. To make this intuition rigorous, we first reduce the residual energy to $m^{-O(1)}$, then we force optimality conditions by changing the weights $\boldsymbol{w}$. While this perfect correction step is generally unnecessary, it will make the description and analysis of our algorithm somewhat cleaner since we will always be able to assume exact centrality.

**Lemma 3.3.8** (Perfect correction)**.** *Let $\boldsymbol{w} \in \mathbb{R}^{2m}$ be a set of weights such that $\mathbf{1} \leq \boldsymbol{w}$, and let $\boldsymbol{f} = \boldsymbol{f}_0 + \boldsymbol{C}\boldsymbol{x}$ be a flow vector with slacks $\boldsymbol{s} > \mathbf{0}$ and residual $\nabla F_\mu^{\boldsymbol{w}}(\boldsymbol{x}) = -\boldsymbol{C}^\top \boldsymbol{h}$ such that $\mathcal{E}_{\boldsymbol{w},\boldsymbol{s}}(\boldsymbol{h}) \leq \varepsilon \leq 1/100$. Then one can compute weights $\boldsymbol{w}' \in \mathbb{R}^{2m}$ such that $\boldsymbol{w} \leq \boldsymbol{w}'$ and $\|\boldsymbol{w}' - \boldsymbol{w}\|_1 \leq \|\boldsymbol{w}\|_1 \cdot 4\sqrt{\varepsilon}$ for which the residual $\nabla F_{\mu'}^{\boldsymbol{w}'}(\boldsymbol{x}) = \mathbf{0}$, where $\mu' \leq \mu(1 + 2\sqrt{\varepsilon})$.*

The complete proof can be found in Appendix 9.1.5. This lemma shows that after performing residual correction until the energy becomes smaller than $m^{-20}/4$, we can slightly increase the weights from $\boldsymbol{w}$ to $\boldsymbol{w}'$ such that for the new objective, $\boldsymbol{x}$ exactly satisfies the optimality condition, as in Equation (3.6). The effect of this perfect correction is an extremely small increase in the sum of weights and the current duality gap. While this step is not essential, it enables us to ensure that

for all the essential points in our analysis we are able to assume *exact* centrality, which comes at a negligible expense, but makes all of our proofs much cleaner. This is summarized in the following lemma, whose proof can be found in Appendix 9.1.5:

**Lemma 3.3.9.** *Suppose that* $\nabla F_{\mu}^{\boldsymbol{w}}(\boldsymbol{x}) = -\boldsymbol{C}^{\top}\boldsymbol{g}$ *for some vector* $\boldsymbol{g}$*, and*

$$\mathcal{E}_{\boldsymbol{w},\boldsymbol{s}}(\boldsymbol{g}) \leq 1/4\,.$$

*Then using* $O(\log\log\|\boldsymbol{w}\|_1)$ *iterations of a vanilla residual correction step, we can obtain a new instance with weights* $\boldsymbol{w}' \geq \boldsymbol{w}$ *and* $\mu' \leq \mu(1+\frac{1}{2}\|\boldsymbol{w}\|_1^{-11})$ *such that*

$$\nabla F_{\mu'}^{\boldsymbol{w}'}(\boldsymbol{x}') = \boldsymbol{0}\,.$$

*and* $\|\boldsymbol{w}'-\boldsymbol{w}\|_1 \leq \|\boldsymbol{w}\|_1^{-10}$.

### 3.3.5 Vanilla Interior Point Method

At this point we are ready to describe a basic interior point method, which requires $\widetilde{O}\left(m^{1/2}\right)$ iterations. The following lemma shows that once we have a $\mu$-central flow, we can scale down $\mu$ by a significant factor such that the new residual can be routed with low energy. The proof appears in Appendix 9.1.5.

**Lemma 3.3.10.** *Let* $\boldsymbol{f} = \boldsymbol{f}_0 + \boldsymbol{C}\boldsymbol{x}$ *be a* $\mu$*-central flow with respect to weights* $\boldsymbol{w} \in \mathbb{R}^{2m}$*, and with slacks* $\boldsymbol{s}$*. Let* $\delta = \frac{1}{(2\|\boldsymbol{w}\|_1)^{1/2}}$*,* $\mu' = \mu/(1+\delta)$*, and the corresponding residual* $\nabla F_{\mu'}^{\boldsymbol{w}}(\boldsymbol{x}) = -\boldsymbol{C}^{\top}\boldsymbol{h}'$*. Then one has that*

$$\mathcal{E}_{\boldsymbol{w},\boldsymbol{s}}(\boldsymbol{h}') \leq 1/4\,.$$

Together with Corollary 3.3.7 this enables us to recover a simple analysis of the classical $\widetilde{O}\left(m^{1/2}\right)$ iteration bound. This is shown in the following lemma, whose proof appears in Appendix 9.1.5.

**Lemma 3.3.11.** *Given a* $\mu^0$*-central flow with respect to weights* $\mathbf{1}$ *and* $\mu^0 = m^{O(1)}$*, we can obtain a minimum cost flow solution with duality gap at most* $\varepsilon = 1/m^{O(1)}$ *using* $\widetilde{O}\left(m^{1/2}\right)$ *calls to the residual correction procedure (Definition 3.3.5).*

As previously discussed, each iteration of the interior point point method can be implemented in $\widetilde{O}(m)$ time using fast Laplacian solvers. This carries over to an algorithm with a total running time of $\widetilde{O}(m^{3/2}\log W)$, matching that of previous classical algorithms.

As we saw in the proof above the major shortcoming of this method consists of only being able to scale down the parameter $\mu$ by $1+1/\Omega(m^{1/2})$ in every step, the reason being that while advancing from $\mu$ to a smaller $\mu'$ we only rely on Lemma 3.3.10 to certify the fact that the new residual can be corrected.

Instead, in what follows we choose to employ a stronger method to produce a new point which satisfies centrality and has a smaller parameter $\mu$. To do so we employ a more sophisticated procedure for producing the new iterate, which also forces some more drastic changes in the weights $\boldsymbol{w}$.

Finally, we make an important observation concerning the vectors that need to be maintained throughout the algorithm.

**Observation 3.3.12.** *A linear system solving oracle which returns the vector* $\boldsymbol{C}\widetilde{\boldsymbol{x}}$ *rather than* $\widetilde{\boldsymbol{x}}$ *when solving the linear system described in Equations (3.11), (3.12) suffices to execute the algorithm. This is because we never need to maintain the explicit solution* $\boldsymbol{x}$ *but rather only the flow* $\boldsymbol{f} = \boldsymbol{C}\boldsymbol{x}$ *and corresponding slacks, as the interior point method works in the space of slacks.*

## 3.4 A Faster Algorithm for Minimum Cost Flow

Our improved algorithm will be based on the interior point method framework that was developed in Section 3.3. The main bottleneck for the running time of that algorithm stems from the fact that the augmenting circulation we compute might not allow us to decrease the duality gap by more than a factor of $1 + 1/\Omega(\sqrt{m})$, as otherwise it is generally impossible to guarantee that the circulation will never congest some edges by more than the available capacity. Hence the iteration bound of $\widetilde{O}(m^{1/2})$, common to standard interior point methods.

We alleviate this difficulty by adding an $\ell_p$ regularization term for the augmenting flow in the objective (3.9), similarly to [115]. In [115], the authors follow the idea of [117] by computing augmenting $s$-$t$ flows. A crucial ingredient is the fact that the congestion of these resulting augmenting flows is then immediately bounded by using a result from [117] which states that as long as there is enough $s$-$t$ residual capacity, these flows come together with an electrical potential embedding, where no edge is too stretched.

However, this property is specific to the $s$-$t$ maximum flow problem. To apply a similar argument for the minimum cost flow problem, one would need to guarantee that *all* cuts of the graph have sufficient residual capacity, which is not automatically enforced as in the case of $s$-$t$ max flow. In order to enforce this cut property, we further regularize our objective in a different way. We do this by temporarily superimposing a star on top of our graph, thus obtaining an augmented graph. This transformation improves the conductance properties of the graph, ensuring that there is enough residual capacity in all cuts of the graph.

In Section 3.4.1, we describe the regularized step problem and outline the guarantees of the solution. In particular, the bias introduced by the regularizers implies that the augmenting flow is not a circulation anymore, and that we have introduced an additional residual for our solution in the barrier objective. We bound the magnitude of both of these perturbations and "undo" them at a later stage. Finally, we present our electrical stretch guarantee, which serves as the crucial ingredient in both preserving feasibility and maintaining centrality.

In Section 3.4.2 we state our choice of regularization parameters and their consequences.

Even though the electrical stretch guarantee suffices for all purposes if the interior point method barrier terms are unweighted, as soon as weights come in the guarantee is affected. In particular, for any edge whose forward and backward weights are too imbalanced, the electrical stretch and congestion bounds that we obtain loosen. In Section 3.4.3 we deal with this issue by ensuring that the forward and backward weights for each edge are always relatively balanced, while introducing an additional demand perturbation.

In Section 3.4.4 we provide the full view of the algorithm, which consists of combining all the ingredients of the previous sections, together with a residual routing scheme that includes both vanilla centering steps and constraint re-weighting to obtain an $\ell_\infty$-based interior point method rather than an $\ell_4$-based one, as achieved by the vanilla algorithm.

As we mentioned, the solution obtained by the interior point method is for a minimum cost flow problem with a slightly perturbed demand. In Section 3.5, we outline an approach given in [42] that given this solution, one can turn it into an optimal solution for the original demand, as long as the total demand perturbation is small.

### 3.4.1 Regularized Newton Step

The initialization procedure from Section 3.3.3 produces a solution with large duality gap, i.e. $O(\mu m)$ where $\mu \leq 2\|\boldsymbol{c}\|_2$. Our goal will be to reduce this by gradually lowering the parameter $\mu$, while maintaining centrality. While in general to achieve this we require solving a sequence of linear

ModifiedAugment$(G, \boldsymbol{c}, \varepsilon; \boldsymbol{w}, \boldsymbol{f}, \mu)$

- Given $\boldsymbol{f}$ : $\mu$-central flow with respect to weights $\boldsymbol{w}$.
- Returns $\boldsymbol{f}''$ : $\mu'$-central flow with respect to weights $\boldsymbol{w}'''$ and with perturbed demand.

1. $\boldsymbol{w}, \boldsymbol{f} \leftarrow$ BalanceWeights$(G; \boldsymbol{w}, \boldsymbol{f})$.
2. Augment graph $G$ to $G_\star$.
3. Let $\widetilde{\boldsymbol{f}}$ be a minimizer to (3.18) where $h_e = \delta \left( \frac{w_e^+}{1-f_e} - \frac{w_e^-}{f_e} \right)$ for each $e \in E$ and $\Delta \boldsymbol{h}$ be the residual perturbation.
4. Augment flow $\boldsymbol{f}' \leftarrow \boldsymbol{f} + \widetilde{\boldsymbol{f}}$.
5. Compute congestion vector $\boldsymbol{\rho} = (\boldsymbol{\rho}^+; \boldsymbol{\rho}^-)$ as $\rho_e^+ = \frac{\widetilde{f}_e}{1-f_e}$, $\rho_e^- = \frac{-\widetilde{f}_e}{f_e}$.
6. Compute new slacks $(\boldsymbol{s}^+)' = \mathbf{1} - \boldsymbol{f}'$ and $(\boldsymbol{s}^-)' = \boldsymbol{f}'$.
7. Correct residual for congested edges:

$$(w_e^+)' = \begin{cases} w_e^+ + \frac{(s_e^+)'}{(s_e^-)'} \cdot w_e^- (\rho_e^-)^2 \,, & \text{if } |\rho_e^-| \geq C_\infty \,, \\ w_e^+ \,, & \text{otherwise,} \end{cases}$$

$$(w_e^-)' = \begin{cases} w_e^- + \frac{(s_e^-)'}{(s_e^+)'} \cdot w_e^+ (\rho_e^+)^2 \,, & \text{if } |\rho_e^+| \geq C_\infty \,, \\ w_e^- \,, & \text{otherwise.} \end{cases}$$

8. Correct perturbed residual given by $\boldsymbol{h}' = -\left( \frac{(\boldsymbol{w}^+)'}{(\boldsymbol{s}^+)'} - \frac{(\boldsymbol{w}^-)'}{(\boldsymbol{s}^-)'} + \frac{\boldsymbol{c}}{\mu/(1+\delta)} + \Delta \boldsymbol{h} \right)$ using Lemma 3.4.19 and get new weights $\boldsymbol{w}''$, flow $\boldsymbol{f}''$, and centrality parameter $\mu'$.
9. Compute new slacks $(\boldsymbol{s}^+)'' = \mathbf{1} - \boldsymbol{f}''$ and $(\boldsymbol{s}^-)'' = \boldsymbol{f}''$.
10. Adjust weights to restore exact centrality:

$$(w_e^+)''' = (w_e^+)'' + \max\left\{0, -(s_e^+)'' \cdot \Delta h_e\right\} \,,$$
$$(w_e^-)''' = (w_e^-)'' + \max\left\{0, (s_e^-)'' \cdot \Delta h_e\right\} \,.$$

11. Return $\boldsymbol{w}''', \boldsymbol{f}'', \mu'$.

Figure 3-3: Modified circulation improvement step

systems of equations (as we saw in Section 3.3.5), here we choose to solve a slightly perturbed linear system.

In order to do so, we modify the optimization problem from (3.9) by adding two regularization terms, which will force the produced solution to be well-behaved. In addition, we allow the newly produced flow $\widetilde{\boldsymbol{f}}$, which we will use to update the current solution, to not be a circulation, as long as the demand it routes is small in $\ell_1$ norm. While this breaks the structure of the problem we are solving, it only does so mildly – therefore once the interior point method has finished running we can repair the broken demand using combinatorial techniques.

**Mixed Objective.** To specify the regularized objective, we first augment the graph $G$ with $O(m)$ extra edges, which are responsible for routing a subset of the flows that would otherwise force the output of the objective to be too degenerate.

**Definition 3.4.1** (Weighted degree)**.** *Given a graph $G(V, E)$ and a weight vector $\boldsymbol{w} = (\boldsymbol{w}^+; \boldsymbol{w}^-) \in \mathbb{R}^{2m}$, the* weighted degree *of $v \in V$ in $G$ with respect to $\boldsymbol{w}$ is defined as $d_v^{\boldsymbol{w}} = \sum_{e \sim v} (w_e^+ + w_e^-)$.*

**Definition 3.4.2.** *Given a graph $G = (V, E)$ we define the augmented graph $G_\star = (V \cup \{v_\star\}, E_\star)$, where $E_\star = E \cup E'$ and $E'$ is obtained by constructing $\lceil d_v^{\boldsymbol{w}} \rceil$ parallel edges $(v, v_\star)$ for each $v \in V$.*

*Furthermore, if $\boldsymbol{C}$ is a cycle basis for $G$, we let $\boldsymbol{C}_\star$ be a cycle basis for $G_\star$ obtained by appending columns to $\boldsymbol{C}$, i.e.*

$$\boldsymbol{C}_\star = \begin{bmatrix} \boldsymbol{C} & \boldsymbol{P}_1 \\ \boldsymbol{0} & \boldsymbol{P}_2 \end{bmatrix}. \tag{3.17}$$

We observe that $|E'| = \sum_{v \in V} \lceil d_v^{\boldsymbol{w}} \rceil \leq \sum_{v \in V} (d_v^{\boldsymbol{w}} + 1) \leq 3\|\boldsymbol{w}\|_1$. We can now write the regularized objective.

**Definition 3.4.3.** *Given a vector $\boldsymbol{h}$, we define the regularized objective as*

$$\begin{aligned} \max_{\widetilde{\boldsymbol{f}} = \boldsymbol{C}_\star \widetilde{\boldsymbol{x}}} \sum_{e \in E} h_e \cdot \widetilde{f}_e - \frac{1}{2} \sum_{e \in E} (\widetilde{f}_e)^2 \cdot \left( \frac{w_e^+}{(s_e^+)^2} + \frac{w_e^-}{(s_e^-)^2} \right) \\ - \frac{R_\star}{2} \sum_{e \in E'} (\widetilde{f}_e)^2 - \frac{R_p}{p} \sum_{e \in E \cup E'} (\widetilde{f}_e)^p, \end{aligned} \tag{3.18}$$

*where $p > 2$ is an even positive integer, and $R_\star$, $R_p$ are some appropriately chosen non-negative scalars.*

While this objective might seem difficult to handle, the fact that we are solving a problem on graphs makes it feasible for our purposes. In particular, the works of [103, 2] show that this objective can be solved to high precision in time $O(m^{1+o(1)})$, whenever $p$ is sufficiently large. We will make this statement more rigorous, but for simplicity let us for now assume that we can solve (3.18) exactly. In Appendix 9.1.3 we will show how to handle the solver error.

Let us now understand the effect of the augmenting edges $E'$. Since they allow routing some of the flow through $v_\star$, if we look at the restriction of $\widetilde{\boldsymbol{f}}$ to the edges of $G$ we see that it stops being a circulation. Let $\widetilde{\boldsymbol{d}}$ be the demand routed by the restriction of $\widetilde{\boldsymbol{f}}$ to $G$. We will see that $\widetilde{\boldsymbol{f}}$ satisfies optimality conditions for an objective similar to (3.18) among all flows that route the demand $\widetilde{\boldsymbol{d}}$ in $G$.

Before that, we give a useful lemma that, given a residual $-\boldsymbol{C}^\top \boldsymbol{h}$, can be used to certify an upper bound on the energy required to route it. We capture this via the following definition.

**Definition 3.4.4.** *Given a vector $\boldsymbol{h}$, weights $\boldsymbol{w}$, and slacks $\boldsymbol{s}$, we define*

$$\mathcal{E}^{\max}(\boldsymbol{h},\boldsymbol{w},\boldsymbol{s}) = \frac{1}{2}\sum_{e\in E} h_e^2 \cdot \left(\frac{w_e^+}{(s_e^+)^2} + \frac{w_e^-}{(s_e^-)^2}\right)^{-1} . \tag{3.19}$$

**Lemma 3.4.5.** *Given weights $\boldsymbol{w}$, slacks $\boldsymbol{s}$, and a residual $-\boldsymbol{C}^\top \boldsymbol{h}$, we have that*

$$\mathcal{E}_{\boldsymbol{w},\boldsymbol{s}}(\boldsymbol{h}) \le \mathcal{E}^{\max}(\boldsymbol{h},\boldsymbol{w},\boldsymbol{s}) .$$

*Furthermore, if $\boldsymbol{h} = \delta\left(\frac{\boldsymbol{w}^+}{\boldsymbol{s}^+} - \frac{\boldsymbol{w}^-}{\boldsymbol{s}^-}\right)$, we have that*

$$\mathcal{E}^{\max}(\boldsymbol{h},\boldsymbol{w},\boldsymbol{s}) \le \frac{1}{2}\delta^2 \left\|\boldsymbol{w}\right\|_1 .$$

The proof of this lemma is given in Appendix 9.1.5. We are now ready to state the lemma that gives guarantees for the restriction of $\widetilde{f}$ to $G$.

**Lemma 3.4.6** (Optimality in the non-augmented graph)**.** *Let $\widetilde{\boldsymbol{f}}_\star = \boldsymbol{C}_\star \widetilde{\boldsymbol{x}}_\star$ be the optimizer of the regularized objective from (3.18), and let $\widetilde{\boldsymbol{f}}$ be its restriction to the edges of $G$. Let $\widetilde{\boldsymbol{d}}$ be the demand routed by $\widetilde{\boldsymbol{f}}$ in $G$. Then $\widetilde{\boldsymbol{f}}$ optimizes the objective*

$$\max_{\substack{\widetilde{\boldsymbol{f}}:\\ \widetilde{\boldsymbol{f}} \text{ routes } \widetilde{\boldsymbol{d}} \text{ in G}}} \left\langle \boldsymbol{h}, \widetilde{\boldsymbol{f}} \right\rangle - \frac{1}{2}\sum_{e\in E} (\widetilde{f}_e)^2 \cdot \left(\frac{w_e^+}{(s_e^+)^2} + \frac{w_e^-}{(s_e^-)^2}\right) - \frac{R_p}{p}\sum_{e\in E} (\widetilde{f}_e)^p . \tag{3.20}$$

*Furthermore*

$$\boldsymbol{C}^\top \left(\frac{\boldsymbol{w}^+}{(\boldsymbol{s}^+)^2} + \frac{\boldsymbol{w}^-}{(\boldsymbol{s}^-)^2}\right) \cdot \widetilde{\boldsymbol{f}} = \boldsymbol{C}^\top (\boldsymbol{h} + \Delta\boldsymbol{h}) , \tag{3.21}$$

*where $\Delta\boldsymbol{h} = -R_p(\widetilde{\boldsymbol{f}})^{p-1}$, and*

$$\|\widetilde{\boldsymbol{d}}\|_1 \le \left(\frac{6\|\boldsymbol{w}\|_1 \cdot \mathcal{E}^{\max}(\boldsymbol{h},\boldsymbol{w},\boldsymbol{s})}{R_\star}\right)^{1/2} , \tag{3.22}$$

$$\|\widetilde{\boldsymbol{f}}_\star\|_p \le \left(\frac{p\cdot \mathcal{E}^{\max}(\boldsymbol{h},\boldsymbol{w},\boldsymbol{s})}{R_p}\right)^{1/p} . \tag{3.23}$$

*Finally, the energy required to route the perturbed residual can be bounded by the energy required to route the original residual:*

$$\frac{1}{2}\sum_{e\in E} (\widetilde{f}_e)^2 \left(\frac{w_e^+}{(s_e^+)^2} + \frac{w_e^-}{(s_e^-)^2}\right) \le 4\cdot \mathcal{E}^{\max}(\boldsymbol{h},\boldsymbol{w},\boldsymbol{s}) . \tag{3.24}$$

*Proof.* Let $\widetilde{\boldsymbol{f}}_\star = \widetilde{\boldsymbol{f}} + \widetilde{\boldsymbol{f}}'$ where $\widetilde{\boldsymbol{f}}'$ is the restriction of $\widetilde{\boldsymbol{f}}_\star$ to the edges incident to $v_\star$. By computing the first order derivative in $\widetilde{\boldsymbol{x}}_\star$, optimality conditions for (3.18) imply that for any circulation $\boldsymbol{g}$ in $G_\star$

one has that

$$\left\langle \boldsymbol{g}, \begin{bmatrix} \boldsymbol{h} - \widetilde{\boldsymbol{f}} \cdot \left( \frac{\boldsymbol{w}^+}{(\boldsymbol{s}^+)^2} + \frac{\boldsymbol{w}^-}{(\boldsymbol{s}^-)^2} \right) - R_p \cdot (\widetilde{\boldsymbol{f}})^{p-1} \\ -R_\star \cdot \widetilde{\boldsymbol{f}}' - R_p \cdot (\widetilde{\boldsymbol{f}}')^{p-1} \end{bmatrix} \right\rangle = 0\,. \tag{3.25}$$

Therefore, restricting ourselves to circulations supported only in the non-preconditioned graph $G$, one has that for any circulation in $\boldsymbol{g}' = \boldsymbol{C}\boldsymbol{z}$ in $G$:

$$\left\langle \boldsymbol{g}', \boldsymbol{h} - \widetilde{\boldsymbol{f}} \cdot \left( \frac{\boldsymbol{w}^+}{(\boldsymbol{s}^+)^2} + \frac{\boldsymbol{w}^-}{(\boldsymbol{s}^-)^2} \right) - R_p \cdot (\widetilde{\boldsymbol{f}})^{p-1} \right\rangle = 0\,, \tag{3.26}$$

and equivalently

$$\left\langle \boldsymbol{z}, \boldsymbol{C}^\top \left( \boldsymbol{h} - \widetilde{\boldsymbol{f}} \cdot \left( \frac{\boldsymbol{w}^+}{(\boldsymbol{s}^+)^2} + \frac{\boldsymbol{w}^-}{(\boldsymbol{s}^-)^2} \right) - R_p \cdot (\widetilde{\boldsymbol{f}})^{p-1} \right) \right\rangle = 0\,. \tag{3.27}$$

Since this holds for any test vector $\boldsymbol{z}$, it must be that the second term in the inner product is 0. Rearranging, it yields the identity from (3.21).

Now we verify that this is the first order optimality condition for the objective in (3.20). Parametrizing the flows that route $\boldsymbol{d}$ via $\widetilde{\boldsymbol{f}} = \boldsymbol{C}\boldsymbol{x} + \widetilde{\boldsymbol{f}}_{\boldsymbol{d}}$ where $\widetilde{\boldsymbol{f}}_{\boldsymbol{d}}$ is an arbitrary flow which routes $\boldsymbol{d}$ in $G$, and setting the derivative with respect to $\boldsymbol{x}$ equal to $\mathbf{0}$, we obtain exactly (3.21).

Let us proceed to bound the norm of the demand routed by $\widetilde{\boldsymbol{f}}$. Consider the value of the objective in (3.18) after truncating it to only the first two terms, which we can write as:

$$\sum_{e \in E} h_e \cdot \widetilde{f}_e - \frac{1}{2} \sum_{e \in E} (\widetilde{f}_e)^2 \cdot \left( \frac{w_e^+}{(s_e^+)^2} + \frac{w_e^-}{(s_e^-)^2} \right) \tag{3.28}$$

$$\leq \frac{1}{2} \sum_{e \in E} h_e^2 \cdot \left( \frac{w_e^+}{(s_e^+)^2} + \frac{w_e^-}{(s_e^-)^2} \right)^{-1} \tag{3.29}$$

$$= \mathcal{E}^{\max}(\boldsymbol{h}, \boldsymbol{w}, \boldsymbol{s})\,, \tag{3.30}$$

where we used the fact that $\langle \boldsymbol{a}, \boldsymbol{b} \rangle \leq \frac{1}{2}\|\boldsymbol{a}\|^2 + \frac{1}{2}\|\boldsymbol{b}\|^2$.

Note that the value of the regularized objective (3.18) is at least 0 since we can always substitute $\widetilde{\boldsymbol{x}} = \mathbf{0}$ and obtain exactly 0. By re-arranging,

$$\frac{R_\star}{2} \sum_{e \in E'} (\widetilde{f}'_e)^2 \tag{3.31}$$

$$\leq \sum_{e \in E} h_e \cdot \widetilde{f}_e - \frac{1}{2} \sum_{e \in E} (\widetilde{f}_e)^2 \cdot \left( \frac{w_e^+}{(s_e^+)^2} + \frac{w_e^-}{(s_e^-)^2} \right) - \frac{R_p}{p} \sum_{e \in E \cup E'} (\widetilde{f}_\star)_e^p \tag{3.32}$$

$$\leq \sum_{e \in E} h_e \cdot \widetilde{f}_e - \frac{1}{2} \sum_{e \in E} (\widetilde{f}_e)^2 \cdot \left( \frac{w_e^+}{(s_e^+)^2} + \frac{w_e^-}{(s_e^-)^2} \right) \tag{3.33}$$

$$\leq \mathcal{E}^{\max}(\boldsymbol{h}, \boldsymbol{w}, \boldsymbol{s})\,, \tag{3.34}$$

where we also used the fact that the last term of (3.32) is non-positive and (3.30). Therefore (3.34)

enables us to upper bound

$$\sum_{e\in E'}(\widetilde{f}'_e)^2 \le \frac{2}{R_\star}\mathcal{E}^{\max}(\boldsymbol{h},\boldsymbol{w},\boldsymbol{s})\,, \tag{3.35}$$

which implies that

$$\sum_{e\in E'}\left|\widetilde{f}'_e\right| \le |E'|^{1/2}\cdot\left(\sum_{e\in E'}(\widetilde{f}'_e)^2\right)^{1/2} \tag{3.36}$$

$$\le \left(3\|\boldsymbol{w}\|_1\cdot\frac{2}{R_\star}\mathcal{E}^{\max}(\boldsymbol{h},\boldsymbol{w},\boldsymbol{s})\right)^{1/2} \tag{3.37}$$

$$= \left(\frac{6\|\boldsymbol{w}\|_1\cdot\mathcal{E}^{\max}(\boldsymbol{h},\boldsymbol{w},\boldsymbol{s})}{R_\star}\right)^{1/2}, \tag{3.38}$$

a quantity that upper bounds the demand perturbation. Using a similar argument we can upper bound $\left\|\widetilde{\boldsymbol{f}}_\star\right\|_p$. We have

$$\frac{R_p}{p}\sum_{e\in E\cup E'}(\widetilde{f}_\star)_e^p \tag{3.39}$$

$$\le \sum_{e\in E}h_e\cdot\widetilde{f}_e - \frac{1}{2}\sum_{e\in E}(\widetilde{f}_e)^2\cdot\left(\frac{w_e^+}{(s_e^+)^2}+\frac{w_e^-}{(s_e^-)^2}\right) - \frac{R_\star}{2}\sum_{e\in E'}(\widetilde{f}'_e)^2 \tag{3.40}$$

$$\le \sum_{e\in E}h_e\cdot\widetilde{f}_e - \frac{1}{2}\sum_{e\in E}(\widetilde{f}_e)^2\cdot\left(\frac{w_e^+}{(s_e^+)^2}+\frac{w_e^-}{(s_e^-)^2}\right) \tag{3.41}$$

$$\le \mathcal{E}^{\max}(\boldsymbol{h},\boldsymbol{w},\boldsymbol{s})\,, \tag{3.42}$$

thus concluding that

$$\left\|\widetilde{\boldsymbol{f}}_\star\right\|_p = \left(\sum_{e\in E\cup E'}(\widetilde{f}_\star)_e^p\right)^{1/p} \le \left(\frac{p\cdot\mathcal{E}^{\max}(\boldsymbol{h},\boldsymbol{w},\boldsymbol{s})}{R_p}\right)^{1/p}. \tag{3.43}$$

Finally, once more using the same argument, we have that

$$\begin{aligned}
&\frac{1}{4}\sum_{e\in E}(\widetilde{f}_e)^2\cdot\left(\frac{w_e^+}{(s_e^+)^2}+\frac{w_e^-}{(s_e^-)^2}\right)\\
&\le \sum_{e\in E}h_e\cdot\widetilde{f}_e - \frac{1}{4}\sum_{e\in E}(\widetilde{f}_e)^2\cdot\left(\frac{w_e^+}{(s_e^+)^2}+\frac{w_e^-}{(s_e^-)^2}\right) - \frac{R_\star}{2}\sum_{e\in E'}(\widetilde{f}'_e)^2 - \frac{R_p}{p}\sum_{e\in E\cup E'}(\widetilde{f}_\star)_e^p\\
&\le \sum_{e\in E}h_e\cdot\widetilde{f}_e - \frac{1}{4}\sum_{e\in E}(\widetilde{f}_e)^2\cdot\left(\frac{w_e^+}{(s_e^+)^2}+\frac{w_e^-}{(s_e^-)^2}\right)\\
&\le 2\cdot\mathcal{E}^{\max}(\boldsymbol{h},\boldsymbol{w},\boldsymbol{s})\,,
\end{aligned}$$

therefore

$$\frac{1}{2}\sum_{e\in E}(\widetilde{f}_e)^2\cdot\left(\frac{w_e^+}{(s_e^+)^2}+\frac{w_e^-}{(s_e^-)^2}\right)\leq 4\cdot\mathcal{E}^{\max}(\boldsymbol{h},\boldsymbol{w},\boldsymbol{s})\,.$$

□

Finally, we present an important property of the solution of the regularized Newton step, which will be crucial for obtaining the final result.

**Lemma 3.4.7.** *Let $\widetilde{\boldsymbol{f}}_\star$ be the solution of the regularized objective (3.18) and $\widetilde{\boldsymbol{f}}$ its restriction on $G$, and suppose that $\|\boldsymbol{w}\|_1\geq 3$. Then one has that over the edges $e\in E$:*

$$\left|\left(\frac{w_e^+}{(s_e^+)^2}+\frac{w_e^-}{(s_e^-)^2}+R_p\cdot\widetilde{f}_e^{p-2}\right)\widetilde{f}_e-h_e\right|\leq\hat{\gamma}\,,\tag{3.44}$$

*where*

$$\hat{\gamma}=\left(R_\star+R_p\cdot\|\widetilde{\boldsymbol{f}}_\star\|_\infty^{p-2}\right)^{1/2}\cdot\left\|\frac{\boldsymbol{h}}{\sqrt{(\boldsymbol{w}^+-\boldsymbol{w}^-)\left(\frac{\boldsymbol{w}^+}{(\boldsymbol{s}^+)^2}+\frac{\boldsymbol{w}^-}{(\boldsymbol{s}^-)^2}\right)}}\right\|_\infty\cdot 32\log\|\boldsymbol{w}\|_1\,.$$

*Furthermore, this implies that*

$$\left(\frac{w_e^+}{(s_e^+)^2}+\frac{w_e^-}{(s_e^-)^2}\right)\cdot\left|\widetilde{f}_e\right|\leq|h_e|+\hat{\gamma}\,.\tag{3.45}$$

Since the proof is technical, we defer it to Section 9.1.2.

### 3.4.2 Choice of Regularization Parameters

Now we state our choice of regularization parameters $R_p$, $R_\star$ and their consequences. In particular, they affect the $\ell_\infty$ norm of the flow $\widetilde{\boldsymbol{f}}$ we obtain by solving the regularized objective, and the preconditioning guarantee (Lemma 3.4.7) which will be essential to obtain the correct trade-off between iteration complexity and demand perturbation.

**Definition 3.4.8.** *Given a weight vector $\boldsymbol{w}$, we will use the following values for the parameters $p$, $R_p$ and $R_\star$:*

$$\begin{aligned}p&=\min\left\{k\in 2\mathbb{Z}:k\geq(\log m)^{1/3}\right\}\,,\\ R_p&=p\cdot\left(10^6\cdot\delta^2\|\boldsymbol{w}\|_1\cdot\log\|\boldsymbol{w}\|_1\right)^{p+1}\,,\\ R_\star&=3\cdot\delta^2\|\boldsymbol{w}\|_1^2\,.\end{aligned}$$

This immediately yields the following useful corollaries.

**Corollary 3.4.9.** *For our specific choice of $p$, the regularized objective from (3.18) can be solved to high precision in $m^{1+o(1)}$ time. We can, furthermore, assume an exact solution.*

*Proof.* We use Theorem 9.1.5 which was proved in [103, 2]. The objective in (3.18) matches exactly the type handled there. For our choice of $p$, the running time is

$$2^{O(p^{3/2})} m^{1+O(1/\sqrt{p})} = 2^{O\sqrt{\log n}} m^{1+1/(\log^{1/6} n)} = m^{1+o(1)} .$$

Furthermore, the resulting solution has a quasi-polynomially small error $2^{-(\log m)^{O(1)}}$, which can be neglected in our analysis, per the discussion in Section 9.1.3. □

The proofs of the following two corollaries appear in Appendix 9.1.5 and 9.1.5, respectively.

**Corollary 3.4.10.** *Let $\widetilde{\boldsymbol{f}}_\star$ be the solution obtained by solving (3.18) and $\widetilde{\boldsymbol{f}}$ be its restriction to $G$. For our specific choice of regularization parameters we have*

$$\|\widetilde{\boldsymbol{f}}_\star\|_p \leq \frac{1}{10^6 \cdot \delta^2 \|\boldsymbol{w}\|_1 \cdot \log \|\boldsymbol{w}\|_1} .$$

**Corollary 3.4.11.** *For the choice of $\boldsymbol{h} = \delta \left( \frac{\boldsymbol{w}^+}{\boldsymbol{s}^+} - \frac{\boldsymbol{w}^-}{\boldsymbol{s}^-} \right)$, and as long as $\delta \leq \|\boldsymbol{w}\|_1^{-(1/4+o(1))}$ the $\hat{\gamma}$ in Lemma 3.4.7 can be upper bounded by*

$$\gamma = \delta^2 \|\boldsymbol{w}\|_1 \cdot 32\sqrt{6} \cdot \log \|\boldsymbol{w}\|_1 .$$

### 3.4.3 Weight Invariants

**Total weight invariant.** Our interior point method will inherently increase edge weights. However, the total increase has to remain bounded by $O(m)$ in order to be able to guarantee an upper bound on the energy $\mathcal{E}_{\boldsymbol{w},\boldsymbol{s}}(\boldsymbol{h})$. We state the following invariant that we intend to always enforce, in order to ensure that this is the case:

**Invariant 3.4.12.** *The sum of weights is bounded: $\|\boldsymbol{w}\|_1 \leq 3m$ .*

**Weight balancing.** Before proceeding with the description of the method, we define a notion that will be used by the algorithm to deal with severe edge weight imbalances. Such imbalances can limit the usability of Lemma 3.4.7 and lead to the augmenting flow $\widetilde{\boldsymbol{f}}$ being infeasible or expensive to correct.

**Definition 3.4.13** (Balanced edges)**.** *An edge $e \in E$ is called* balanced *if $\max\{w_e^+, w_e^-\} \leq \delta \|\boldsymbol{w}\|_1$ or $\min\{w_e^+, w_e^-\} \geq 96 \cdot \delta^4 \|\boldsymbol{w}\|_1^2$. Otherwise it is called* imbalanced*.*

Even though imbalanced edges can generally emerge, we are able to balance them by manually reducing the disparity between $w_e^+$ and $w_e^-$, while breaking the flow demand by a controllable amount. We will maintain the following invariant right before solving the regularized objective:

**Invariant 3.4.14.** *All edges are balanced.*

In Section 3.4.4 we will see that there is a way to enforce Invariant 3.4.14 while only increasing the weight and breaking the demand by a small amount. The main motivation behind keeping edges balanced is that it implies that each edge either has a favorable stretch property, or is not too congested:

**Lemma 3.4.15.** *Let $\widetilde{\boldsymbol{f}}$ be the restriction of the regularized problem (3.18) solution to $G$, with $\boldsymbol{h} = \delta \left( \frac{\boldsymbol{w}^+}{\boldsymbol{s}^+} - \frac{\boldsymbol{w}^-}{\boldsymbol{s}^-} \right)$ and congestion $\boldsymbol{\rho}$, where $\delta \leq \|\boldsymbol{w}\|_1^{-(1/4+o(1))}$. If $e \in E$ is balanced, then $\max\{|\rho_e^+|, |\rho_e^-|\} < C_\infty$ or $\left| \frac{w_e^+ \rho_e^+}{s_e^+} \right| + \left| \frac{w_e^- \rho_e^-}{s_e^-} \right| \leq 6\gamma$, where $C_\infty = \frac{1}{2\delta\sqrt{2\|\boldsymbol{w}\|_1}}$.*

The proof of this lemma appears in Appendix 9.1.5.

### 3.4.4 Executing the Interior Point Method

Having defined the regularized objective, we now show how to execute the interior point method using the solution returned by a high precision solver. Since the solution to this objective does not exactly match the guarantee required in (3.11-3.12), we will have to do some slight manual adjustments.

In the vanilla interior point method analysis that we saw earlier, we witnessed a very stringent requirement on the condition that we are able to correct a residual. Namely, we required that the energy required to route it decreases in every iteration of the correction step, which was guaranteed by the fact that after performing the first correction step the upper bound on energy $\sum w_i \rho_i^4$ is at most a small constant (i.e. $1/4$).

This requirement is too strong since, as a matter of fact, the most important obstacle handled by interior point methods is preserving slack feasibility. In our specific context this means that we want to perform updates to the current flow without violating capacity constraints, which is guaranteed by a weaker $\ell_\infty$ bound, i.e. $\|\boldsymbol{\rho}\|_\infty \leq 1/2$. While this condition is sufficient to preserve feasibility, it is not clear that after performing the corresponding update to the flow, the energy required to route the residual will be small, so the resulting residual can be reduced to $\mathbf{0}$. Instead we can enforce this property by canceling the components of the gradient which cause this energy to be large.

**Definition 3.4.16** (Perturbed residual correction)**.** *Consider a flow $\boldsymbol{f}$ with the corresponding slack vector $\boldsymbol{s} > \mathbf{0}$, weights $\boldsymbol{w}$ and parameter $\mu > 0$, with a corresponding residual $\nabla F_\mu^{\boldsymbol{w}}(\boldsymbol{x}) = -\boldsymbol{C}^\top \boldsymbol{h}$ where $\boldsymbol{h} = \delta \left( \frac{\boldsymbol{w}^+}{\boldsymbol{s}^+} - \frac{\boldsymbol{w}^-}{\boldsymbol{s}^-} \right)$ and $\delta \leq \|\boldsymbol{w}\|_1^{-1/4}/2$. The perturbed residual correction step is defined as an update to $\boldsymbol{f}$ via:*

$$\boldsymbol{f}' = \boldsymbol{f} + \widetilde{\boldsymbol{f}}\,, \tag{3.46}$$

$$(\boldsymbol{s}^-)' = \boldsymbol{s}^- + \widetilde{\boldsymbol{f}}\,, \tag{3.47}$$

$$(\boldsymbol{s}^+)' = \boldsymbol{s}^+ - \widetilde{\boldsymbol{f}}\,, \tag{3.48}$$

*where $\widetilde{\boldsymbol{f}}$ is the solution to the linear system*

$$\boldsymbol{\rho}^+ = \frac{\widetilde{\boldsymbol{f}}}{\boldsymbol{s}^+}\,, \tag{3.49}$$

$$\boldsymbol{\rho}^- = \frac{-\widetilde{\boldsymbol{f}}}{\boldsymbol{s}^-}\,, \tag{3.50}$$

$$\boldsymbol{C}^\top \left( \frac{\boldsymbol{w}^+ \boldsymbol{\rho}^+}{\boldsymbol{s}^+} - \frac{\boldsymbol{w}^- \boldsymbol{\rho}^-}{\boldsymbol{s}^-} \right) = \boldsymbol{C}^\top (\boldsymbol{h} + \Delta \boldsymbol{h})\,, \tag{3.51}$$

*such that*

$$\|\boldsymbol{\rho}\|_\infty \leq \frac{1}{2}\,, \tag{3.52}$$

*for some perturbation $\Delta \boldsymbol{h}$, followed by the updates to the $\boldsymbol{w}$ vector via:*

$$(w_e^+)' = \begin{cases} w_e^+ + \frac{(s_e^+)'}{(s_e^-)'} \cdot w_e^- (\rho_e^-)^2 & \text{if } |\rho_e^-| \geq C_\infty\,, \\ w_e^+ & \text{otherwise,} \end{cases} \tag{3.53}$$

*and*

$$(w_e^-)' = \begin{cases} w_e^- + \frac{(s_e^-)'}{(s_e^+)'} \cdot w_e^+(\rho_e^+)^2 & \text{if } |\rho_e^+| \geq C_\infty\,, \\ w_e^- & \text{otherwise,} \end{cases} \tag{3.54}$$

*where*

$$C_\infty = \frac{1}{2\delta\sqrt{2\left\|\boldsymbol{w}\right\|_1}}\,.$$

We also give a short lemma, which upper bounds the amount by which $\|\boldsymbol{w}\|_1$ gets increased when applying the perturbed residual correction.

**Lemma 3.4.17.** *Let $\boldsymbol{w}$ and $\boldsymbol{w}'$ be the old and new weights, respectively, as described in Definition 3.4.16. Then one has that*

$$\|\boldsymbol{w}' - \boldsymbol{w}\|_1 \leq 192\sqrt{2}\cdot\gamma\cdot\left(\delta^2\|\boldsymbol{w}\|_1\right)^{3/2}\,.$$

*Proof.* Let $S^+ = \{e \in E \,:\, |\rho_e^+| \geq C_\infty\}$ and $S^- = \{e \in E \,:\, |\rho_e^-| \geq C_\infty\}$. The total weight increase is equal to

$$\sum_{e\in S^+}\left((w_e^-)' - w_e^-\right) + \sum_{e\in S^-}\left((w_e^+)' - w_e^+\right)\,.$$

Let us upper bound the weight increase contributed by a single edge $e \in S^+$. The respective bound will follow for an edge in $S^-$ by symmetry. We have

$$(w_e^-)' - w_e^- = (s_e^-)'\cdot\frac{w_e^+(\rho_e^+)^2}{(s_e^+)'} \leq 4\cdot s_e^-\cdot\frac{w_e^+(\rho_e^+)^2}{s_e^+} = 4\cdot s_e^-\cdot|\rho_e^+|\cdot\left|\frac{w_e^+\rho_e^+}{s_e^+}\right|\,.$$

By Lemma 9.20, all the edges in $S^+\cup S^-$ have the low stretch property:

$$\left|\frac{w_e^+\rho_e^+}{s_e^+}\right| + \left|\frac{w_e^-\rho_e^-}{s_e^-}\right| \leq 6\gamma\,,$$

and so

$$(w_e^-)' - w_e^- \leq 4\cdot s_e^-\cdot|\rho_e^+|\cdot\left|\frac{w_e^+\rho_e^+}{s_e^+}\right| \leq 24\gamma|\rho_e^+|\,.$$

The total weight increase due to $S^+$ is thus:

$$\sum_{e\in S^+}\left((w_e^-)' - w_e^-\right) \leq 24\gamma\sum_{e\in S^+}\left|\rho_e^+\right|\,.$$

Symmetrically for $S^-$ we get

$$\sum_{e\in S^-}\left((w_e^+)' - w_e^+\right) \leq 24\gamma\sum_{e\in S^-}\left|\rho_e^-\right|\,,$$

and so

$$\|\boldsymbol{w}' - \boldsymbol{w}\|_1 \leq 24\gamma\cdot\|\boldsymbol{\rho}_{S^+\cup S^-}\|_1\,.$$

Now, since by Lemma 3.4.6 the energy to route the perturbed residual is bounded by the energy to

route the original residual,

$$\|\rho\|_2^2 \le \sum_{e\in E}\left(w_e^+(\rho_e^+)^2 + w_e^-(\rho_e^-)^2\right) = \sum_{e\in E}(\widetilde{f}_e)^2\left(\frac{w_e^+}{(s_e^+)^2} + \frac{w_e^-}{(s_e^-)^2}\right) \le 8\cdot\mathcal{E}^{\max}(\boldsymbol{h},\boldsymbol{w},\boldsymbol{s})\,,$$

we have that

$$\begin{aligned}\|\boldsymbol{\rho}_{S^+\cup S^-}\|_1 &\le \|\boldsymbol{\rho}\|_2^2\cdot\max\left\{\left\|1/\boldsymbol{\rho}_{S^+}^+\right\|_\infty, \left\|1/\boldsymbol{\rho}_{S^-}^-\right\|_\infty\right\}\\ &\le 8\cdot\mathcal{E}^{\max}(\boldsymbol{h},\boldsymbol{w},\boldsymbol{s})\cdot C_\infty^{-1}\\ &\le 4\cdot\delta^2\|\boldsymbol{w}\|_1\cdot 2\cdot\delta\sqrt{\|\boldsymbol{w}\|_1}\\ &= 8\sqrt{2}\cdot\left(\delta^2\|\boldsymbol{w}\|_1\right)^{3/2}.\end{aligned}$$

Thus we conclude that

$$\|\boldsymbol{w}'-\boldsymbol{w}\|_1 \le 192\sqrt{2}\cdot\gamma\cdot\left(\delta^2\|\boldsymbol{w}\|_1\right)^{3/2}.$$

□

The effect of the weight updates is to zero out the coordinates in the new residual that contribute a lot to energy. We formalize this intuition in the lemma below, whose proof appears in Appendix 9.1.5.

**Lemma 3.4.18.** *Suppose that we perform a perturbed residual correction step as described in Definition 3.4.16 and obtain a solution $\boldsymbol{x}'$ with weights $\boldsymbol{w}'$, residual $\nabla F_\mu^{\boldsymbol{w}'}(\boldsymbol{x}') = -\boldsymbol{C}^\top\boldsymbol{g}$, and let $\Delta\boldsymbol{h}$ be the perturbation of the residual. Then we have that*

$$\mathcal{E}_{\boldsymbol{w}',\boldsymbol{s}'}(\boldsymbol{g}+\Delta\boldsymbol{h}) \le 1/4\,. \tag{3.55}$$

The next short lemma is a straightforward application of the vanilla correction and perfect correction routines, which shows that while only very slightly perturbing weights, we can obtain an instance where $\nabla F_\mu^{\boldsymbol{w}''}(\boldsymbol{x}'') - \boldsymbol{C}^\top\Delta\boldsymbol{h} = 0$.

**Lemma 3.4.19.** *Suppose that $\nabla F_\mu^{\boldsymbol{w}'}(\boldsymbol{x}') = -\boldsymbol{C}^\top\boldsymbol{g}$ for some vector $\boldsymbol{g}$, and*

$$\mathcal{E}_{\boldsymbol{w}',\boldsymbol{s}'}(\boldsymbol{g}+\Delta\boldsymbol{h}) \le 1/4\,.$$

*Then using $O(\log\log m)$ iterations of a vanilla residual correction step, we can obtain a new instance with weights $\boldsymbol{w}''$ and $\mu' \le \mu(1+\frac{1}{2}\|\boldsymbol{w}\|_1^{-11})$ such that*

$$\nabla F_{\mu'}^{\boldsymbol{w}''}(\boldsymbol{x}'') = \boldsymbol{C}^\top\Delta\boldsymbol{h}\,.$$

*and $\|\boldsymbol{w}''-\boldsymbol{w}'\|_1 \le \|\boldsymbol{w}'\|_1^{-10}$.*

*Proof.* We apply Lemma 3.3.9 for the perturbed function $\hat{F}_\mu^{\boldsymbol{w}'}(\boldsymbol{x}) = F_\mu^{\boldsymbol{w}'}(\boldsymbol{x}) - \langle\Delta\boldsymbol{h},\boldsymbol{C}\boldsymbol{x}\rangle$ to obtain a new solution $\boldsymbol{f}'' = \boldsymbol{f}_0 + \boldsymbol{C}\boldsymbol{x}''$ and a new set of weights $\boldsymbol{w}'' \ge \boldsymbol{w}'$ such that $\boldsymbol{w}'' \le \boldsymbol{w}'(1-m^{-10})$ and $\nabla F_{\mu'}^{\boldsymbol{w}''}(\boldsymbol{x}'') - \boldsymbol{C}^\top\Delta\boldsymbol{h} = \nabla\hat{F}_{\mu'}^{\boldsymbol{w}''}(\boldsymbol{x}'') = \boldsymbol{0}$. □

**Lemma 3.4.20.** *Suppose we have an instance where $\nabla F_\mu^{\boldsymbol{w}}(\boldsymbol{x}) = \boldsymbol{C}^\top\Delta\boldsymbol{h}$. Then there exists a set of weights $\boldsymbol{w}' \ge \boldsymbol{w}$ such that $\nabla F_\mu^{\boldsymbol{w}'}(\boldsymbol{x}) = \boldsymbol{0}$ and $\|\boldsymbol{w}'-\boldsymbol{w}\|_1 \le \|\Delta\boldsymbol{h}\|_1$.*

*Proof.* By definition we have

$$\frac{\boldsymbol{C}^\top \boldsymbol{c}}{\mu} + \boldsymbol{C}^\top \left( \frac{\boldsymbol{w}^+}{\boldsymbol{s}^+} - \frac{\boldsymbol{w}^-}{\boldsymbol{s}^-} \right) = \boldsymbol{C}^\top \Delta \boldsymbol{h} \,. \tag{3.56}$$

We can restore exact centrality by perturbing the weights via the simple update:

$$(\boldsymbol{w}^+)' = \boldsymbol{w}^+ - \boldsymbol{s}^+ \cdot (\Delta \boldsymbol{h})_{\leq 0} \,, \tag{3.57}$$
$$(\boldsymbol{w}^-)' = \boldsymbol{w}^- + \boldsymbol{s}^- \cdot (\Delta \boldsymbol{h})_{\geq 0} \,. \tag{3.58}$$

Plugging into the above identity immediately yields the desired condition:

$$\begin{aligned}
\nabla F_\mu^{\boldsymbol{w}'}(\boldsymbol{x}) &= \frac{\boldsymbol{C}^\top \boldsymbol{c}}{\mu} + \boldsymbol{C}^\top \left( \frac{(\boldsymbol{w}^+)'}{\boldsymbol{s}^+} - \frac{(\boldsymbol{w}^-)'}{\boldsymbol{s}^-} \right) \\
&= \frac{\boldsymbol{C}^\top \boldsymbol{c}}{\mu} + \boldsymbol{C}^\top \left( \frac{\boldsymbol{w}^+}{\boldsymbol{s}^+} - \frac{\boldsymbol{w}^-}{\boldsymbol{s}^-} \right) + \boldsymbol{C}^\top \left( \frac{(\boldsymbol{w}^+)' - \boldsymbol{w}^+}{\boldsymbol{s}^+} - \frac{(\boldsymbol{w}^-)' - \boldsymbol{w}^-}{\boldsymbol{s}^-} \right) \\
&= \boldsymbol{C}^\top \Delta \boldsymbol{h} - \boldsymbol{C}^\top \Delta \boldsymbol{h} \\
&= \boldsymbol{0} \,.
\end{aligned}$$

Finally, we bound

$$\|\boldsymbol{w}' - \boldsymbol{w}\|_1 = -\langle \boldsymbol{s}^+, (\Delta \boldsymbol{h})_{\leq 0} \rangle + \langle \boldsymbol{s}^-, (\Delta \boldsymbol{h})_{\geq 0} \rangle \leq \|\boldsymbol{s}\|_\infty \cdot \|\Delta \boldsymbol{h}\|_1 \leq \|\Delta \boldsymbol{h}\|_1 \,.$$

For the final inequality we crucially used the fact that all slacks are at most 1. □

Combining Lemmas 3.4.18 and 3.4.20 together with the vanilla correction step (Corollary 3.3.7 and Lemma 3.3.8) we can derive an improved correction step based on the solution to the regularized objective. First we show that indeed this is possible, i.e. for a particular choice of regularization parameters we obtain a step with $\|\boldsymbol{\rho}\|_\infty \leq 1/2$.

**Lemma 3.4.21** (Feasibility lemma)**.** *Suppose we have an instance with weights $\boldsymbol{w}$ and slacks $\boldsymbol{s}$, residual $-\boldsymbol{C}^\top \boldsymbol{h}$ where $\boldsymbol{h} = \delta \left( \frac{\boldsymbol{w}^+}{\boldsymbol{s}^+} - \frac{\boldsymbol{w}^-}{\boldsymbol{s}^-} \right)$, and $\|\boldsymbol{w}\|_1^{-1/2} \leq \delta \leq \|\boldsymbol{w}\|_1^{-(1/4+o(1))}$. Then by solving the regularized objective we obtain a flow $\widetilde{\boldsymbol{f}}$ satisfying*

$$\begin{aligned}
\boldsymbol{\rho}^+ &= \frac{\widetilde{\boldsymbol{f}}}{\boldsymbol{s}^+} \,, \\
\boldsymbol{\rho}^- &= \frac{-\widetilde{\boldsymbol{f}}}{\boldsymbol{s}^-} \,, \\
\boldsymbol{C}^\top \left( \frac{\boldsymbol{w}^+ \boldsymbol{\rho}^+}{\boldsymbol{s}^+} - \frac{\boldsymbol{w}^- \boldsymbol{\rho}^-}{\boldsymbol{s}^-} \right) &= \boldsymbol{C}^\top (\boldsymbol{h} + \Delta \boldsymbol{h}) \,.
\end{aligned}$$

*such that:*

1. *the congestion satisfies*
$$\|\boldsymbol{\rho}\|_\infty \leq 1/2 \,,$$
2. *the perturbation $\Delta \boldsymbol{h}$ is bounded*
$$\|\Delta \boldsymbol{h}\|_1 \leq p \cdot \|\boldsymbol{w}\|_1^{1/p} \cdot (10^6 \cdot \delta^2 \|\boldsymbol{w}\|_1 \log \|\boldsymbol{w}\|_1)^2 \,,$$

*3. the flow $\widetilde{\boldsymbol{f}}$ routes a demand $\widetilde{\boldsymbol{d}}$ such that*

$$\|\widetilde{\boldsymbol{d}}\|_1 \leq 1\,.$$

*Proof.*

**1.** Let us first verify the crucial feature of $\widetilde{\boldsymbol{f}}$, namely that it ensures a low congestion $\|\boldsymbol{\rho}\|_\infty \leq 1/2$. For our specific choice of $\boldsymbol{h}$ we can upper bound, by applying Lemma 3.4.5:

$$\mathcal{E}^{\max}(\boldsymbol{h}, \boldsymbol{w}, \boldsymbol{s}) \leq \frac{1}{2}\delta^2 \|\boldsymbol{w}\|_1\,.$$

First we note that for this choice of parameters we can upper bound, using Corollary 3.4.10:

$$\|\widetilde{\boldsymbol{f}}\|_\infty \leq \|\widetilde{\boldsymbol{f}}\|_p \leq \frac{1}{10^6 \cdot \delta^2 \|\boldsymbol{w}\|_1 \cdot \log \|\boldsymbol{w}\|_1}\,. \tag{3.59}$$

By Lemma 3.4.15, for each edge $e \in E$ we either have

$$\max\left\{\left|\rho_e^+\right|, \left|\rho_e^-\right|\right\} < C_\infty = \frac{1}{2\delta\sqrt{2\,\|\boldsymbol{w}\|_1}} \leq 1/2$$

by our assumption on $\delta$, in which case we are done, or the low stretch property

$$\left|\frac{w_e^+ \rho_e^+}{s_e^+}\right| + \left|\frac{w_e^- \rho_e^-}{s_e^-}\right| \leq 6\gamma$$

holds, where by Corollary 3.4.11

$$\gamma = \delta^2 \|\boldsymbol{w}\|_1 \cdot 32\sqrt{6} \cdot \log \|\boldsymbol{w}\|_1\,.$$

This implies that

$$\left|\rho_e^+\right| = \left|\rho_e^+ s_e^+ \cdot \frac{\rho_e^+}{s_e^+}\right|^{1/2} = \left(\left|\widetilde{f}_e\right| \cdot \left|\frac{\rho_e^+}{s_e^+}\right|\right)^{1/2} \leq \left(\frac{1}{10^6 \cdot \delta^2\,\|\boldsymbol{w}\|_1 \cdot \log \|\boldsymbol{w}\|_1} \cdot 6\gamma\right)^{1/2} < \frac{1}{2}\,.$$

The argument for $\rho_e^-$ is symmetric. Therefore we have that $\|\boldsymbol{\rho}\|_\infty \leq 1/2$.

**2.** Now let us upper bound the $\ell_1$ norm of the perturbation $\Delta\boldsymbol{h}$. Per Lemma 3.4.6, our step produced by solving the regularized objective yields

$$\Delta\boldsymbol{h} = -R_p(\widetilde{\boldsymbol{f}})^{p-1}\,.$$

Therefore we can control

$$\|\Delta\boldsymbol{h}\|_1 = R_p \|\widetilde{\boldsymbol{f}}\|_{p-1}^{p-1} \leq R_p \left(\|\boldsymbol{w}\|_1^{\frac{1}{p-1}-\frac{1}{p}} \cdot \|\widetilde{\boldsymbol{f}}\|_p\right)^{p-1} = R_p \|\boldsymbol{w}\|_1^{1/p} \cdot \|\widetilde{\boldsymbol{f}}\|_p^{p-1}\,.$$

Using (3.59) we further upper bound this by

$$\begin{aligned}\|\Delta \boldsymbol{h}\|_1 &\leq p \cdot (10^6 \cdot \delta^2 \|\boldsymbol{w}\|_1 \cdot \log \|\boldsymbol{w}\|_1)^{p+1} \cdot \|\boldsymbol{w}\|_1^{1/p} \cdot \left(\frac{1}{10^6 \cdot \delta^2 \|\boldsymbol{w}\|_1 \cdot \log \|\boldsymbol{w}\|_1}\right)^{p-1} \\ &\leq p \cdot \|\boldsymbol{w}\|_1^{1/p} \cdot \left(10^6 \cdot \delta^2 \|\boldsymbol{w}\|_1 \cdot \log \|\boldsymbol{w}\|_1\right)^2 .\end{aligned}$$

**3.** Finally, let us upper bound the demand routed by $\widetilde{\boldsymbol{f}}$. For the demand perturbation, by Lemma 3.4.6 one has that after optimizing the regularized objective the resulting flow $\widetilde{\boldsymbol{f}}$ routes a demand $\widetilde{\boldsymbol{d}}$ such that

$$\|\widetilde{\boldsymbol{d}}\|_1 \leq \left(\frac{6\|\boldsymbol{w}\|_1 \cdot \mathcal{E}^{\max}(\boldsymbol{h}, \boldsymbol{w}, \boldsymbol{s})}{R_\star}\right)^{1/2} \leq \left(\frac{3\delta^2 \|\boldsymbol{w}\|_1^2}{R_\star}\right)^{1/2} = 1 ,$$

which gives us what we needed.

□

**Definition 3.4.22** (Weight balancing procedure)**.** *Given a flow $\boldsymbol{f}$ that is $\mu$-central with respect to weights $\boldsymbol{w}$ and with slacks $\boldsymbol{s}$, let $S \subseteq E$ be the set of edges that are not balanced (Invariant 3.4.14). The weight balancing procedure consists of computing new weights $\boldsymbol{w}'$ such that*

- *For each $e \in S$: If $w_e^+ \leq w_e^-$ then $w_e^{'+} = 96 \cdot \delta^4 \|\boldsymbol{w}\|_1^2$, $w_e^{'-} = w_e^-$, while if $w_e^+ > w_e^-$ then $w_e^{'+} = w_e^+$, $w_e^{'-} = 96 \cdot \delta^4 \|\boldsymbol{w}\|_1^2$.*
- *For each $e \notin S$ we set $w_e^{'+} = w_e^+$, $w_e^{'-} = w_e^-$.*

*Additionally, we compute a flow $\boldsymbol{f}'$ with slacks $\boldsymbol{s}' > \boldsymbol{0}$ such that*

$$\frac{\boldsymbol{w}^{'+}}{\boldsymbol{s}^{'+}} - \frac{\boldsymbol{w}^{'-}}{\boldsymbol{s}^{'-}} = \frac{\boldsymbol{w}^+}{\boldsymbol{s}^+} - \frac{\boldsymbol{w}^-}{\boldsymbol{s}^-} .$$

**Lemma 3.4.23** (Weight balancing lemma)**.** *Given $\boldsymbol{f}, \boldsymbol{w}, \boldsymbol{s}$, $S$ and $\mu$ as in Definition 3.4.22 after applying the weight balancing procedure we get a $\mu$-central flow $\boldsymbol{f}'$ with respect to weights $\boldsymbol{w}' \geq \boldsymbol{w}$ and with slacks $\boldsymbol{s}'$ such that all edges satisfy Invariant 3.4.14 with respect to $\boldsymbol{w}'$ and $\boldsymbol{s}'$. Additionally*

$$\left\|\boldsymbol{w}' - \boldsymbol{w}\right\|_1 \leq 96 \cdot |S| \cdot \delta^4 \|\boldsymbol{w}\|_1^2 ,$$

*and*

$$\left\|\boldsymbol{d}' - \boldsymbol{d}\right\|_1 \leq |S| ,$$

*and $\|d' - d\|_1 \leq |S|$ where $\boldsymbol{d}$ is the demand routed by $\boldsymbol{f}$ and $\boldsymbol{d}'$ the demand routed by $\boldsymbol{f}'$.*

*Proof.* Let $e \in S$ and without loss of generality $w_e^+ < 96 \cdot \delta^4 \|\boldsymbol{w}\|_1^2$ and $w_e^- > \delta \|\boldsymbol{w}\|_1$. First of all, we have $w_e^{'+} = 96 \cdot \delta^4 \|\boldsymbol{w}\|_1^2 > w_e^+$ and the weight increase $w_e^{'+} - w_e^+$ is at most $96 \cdot \delta^4 \|\boldsymbol{w}\|_1^2$. Furthermore, let us look at the function $g(t) = \frac{w_e^{'+}}{1-t} - \frac{w_e^{'-}}{t}$ for $t \in (0, 1)$. This is a continuous increasing function with $\lim_{t \to 0} g(t) = -\infty$ and $\lim_{t \to 1} g(t) = \infty$. Therefore there exists a unique $f_e' \in (0, 1)$ such that

$$\frac{w_e^{'+}}{1 - f_e'} - \frac{w_e^{'-}}{f_e'} = \frac{w_e^+}{1 - f_e} - \frac{w_e^-}{f_e} .$$

As $f_e, f'_e \in (0,1)$, the demand perturbation $|f'_e - f_e|$ on this edge is at most 1. Putting everything together, we get that $\|\boldsymbol{w}' - \boldsymbol{w}\|_1 \leq 96 \cdot |S| \cdot \delta^4 \|\boldsymbol{w}\|_1^2$ and $\left\|\boldsymbol{d}' - \boldsymbol{d}\right\|_1 \leq |S|$. □

**Lemma 3.4.24** (Progress lemma)**.** *Given a central instance with parameter $\mu$ and weights $\boldsymbol{w}$, we can obtain a new central instance with parameter $\mu/(1+\delta)$ and weights $\boldsymbol{w}''' + \Delta\boldsymbol{w} \geq \boldsymbol{w}$ with*

$$\delta \geq m^{-1/4}/10\,,$$

*such that*

$$\begin{aligned}\|\boldsymbol{w}''' - \boldsymbol{w}\|_1 \leq \|\Delta\boldsymbol{w}\|_1 &\\ + \left(\delta^2\|\boldsymbol{w} + \Delta\boldsymbol{w}\|_1\right)^{5/2} \cdot 6 \cdot 10^4 \cdot \log\|\boldsymbol{w} + \Delta\boldsymbol{w}\|_1 &\\ + m^{-10} &\\ + \left(10^6 \cdot \delta^2\|\boldsymbol{w} + \Delta\boldsymbol{w}\|_1 \cdot \log\|\boldsymbol{w} + \Delta\boldsymbol{w}\|_1\right)^2 \cdot p \cdot \|\boldsymbol{w} + \Delta\boldsymbol{w}\|_1^{1/p}\,.&\end{aligned}$$

*where $\Delta\boldsymbol{w} \geq \mathbf{0}$ is the weight increase caused by applying the procedure described in Definition 3.4.22 on weights $\boldsymbol{w}$. Furthermore, the demand perturbation is $\widetilde{\boldsymbol{d}} + \Delta\widetilde{\boldsymbol{d}}$, where*

$$\|\widetilde{\boldsymbol{d}}\|_1 \leq 1\,.$$

*and $\Delta\widetilde{\boldsymbol{d}}$ is the demand perturbation caused by applying the procedure described in Definition 3.4.22 on weights $\boldsymbol{w}$.*

*Proof.* We first apply the weight balancing procedure as described in Definition 3.4.22 to get new weights $\boldsymbol{w} + \Delta\boldsymbol{w} \geq \boldsymbol{w}$.

We will apply the residual correction step with the flow $\widetilde{\boldsymbol{f}}$ guaranteed by Lemma 3.4.21, which guarantees that the corresponding flow yields a congestion $\|\boldsymbol{\rho}\|_\infty \leq 1/2$. As this step is obtained for a perturbed residual $\nabla F_\mu^{\boldsymbol{w}+\Delta\boldsymbol{w}}(\boldsymbol{x}) - \boldsymbol{C}^\top\boldsymbol{h} = -\boldsymbol{C}^\top(\boldsymbol{h} + \Delta\boldsymbol{h})$, it means that after executing it we obtain an instance with weights $\boldsymbol{w}'$ and slacks $\boldsymbol{s}'$ such that

$$\mathcal{E}_{\boldsymbol{w}',\boldsymbol{s}'}(\boldsymbol{h}' + \Delta\boldsymbol{h}) \leq 1/4\,,$$

where $\nabla F_\mu^{\boldsymbol{w}'}(\boldsymbol{x}) = -\boldsymbol{C}^\top\boldsymbol{h}'$. This follows from applying Lemma 3.4.18. Therefore, using Lemma 3.4.19 we can obtain a new point where

$$\nabla F_\mu^{\boldsymbol{w}''}(\boldsymbol{x}'') = \boldsymbol{C}^\top\Delta\boldsymbol{h}\,,$$

for a slight change in weights from $\boldsymbol{w}'$ to $\boldsymbol{w}''$. Finally, applying Lemma 3.4.20 we can establish exact centrality

$$\nabla F_\mu^{\boldsymbol{w}'''}(\boldsymbol{x}'') = \mathbf{0}\,,$$

while slightly increasing weights from $\boldsymbol{w}''$ to $\boldsymbol{w}'''$.

Having described the method, let us first bound the weight change it causes. By Lemma 3.4.17 we have that the weight increase $\|\boldsymbol{w}' - (\boldsymbol{w} + \Delta\boldsymbol{w})\|_1$ caused by performing the perturbations described in Definition 3.4.16 is upper bounded by

$$192\sqrt{2} \cdot \gamma \cdot (\delta^2\|\boldsymbol{w} + \Delta\boldsymbol{w}\|_1)^{3/2}\,,$$

where

$$\gamma = \delta^2\|\boldsymbol{w} + \Delta\boldsymbol{w}\|_1 \cdot 32\sqrt{6} \cdot \log\|\boldsymbol{w} + \Delta\boldsymbol{w}\|_1\,.$$

Furthermore, restoring the condition that $\nabla F_\mu^{\boldsymbol{w}''}(\boldsymbol{x}'') = \boldsymbol{C}^\top\Delta\boldsymbol{h}$ is done while increasing the $\ell_1$ norm

of the weight vector by $\|\boldsymbol{w}'' - \boldsymbol{w}'\|_1 \leq m^{-10}$ and restoring exact centrality via Lemma 3.4.6 costs us a further weight increase $\|\boldsymbol{w}''' - \boldsymbol{w}''\|_1$ that is upper bounded by

$$\|\Delta \boldsymbol{h}\|_1 \leq p \cdot \|\boldsymbol{w} + \Delta \boldsymbol{w}\|_1^{1/p} \cdot (10^6 \cdot \delta^2 \|\boldsymbol{w} + \Delta \boldsymbol{w}\|_1 \log \|\boldsymbol{w} + \Delta \boldsymbol{w}\|_1)^2 \,,$$

where we used the guarantee from Lemma 3.4.21. Therefore we conclude that

$$\begin{aligned}
\|\boldsymbol{w}''' - \boldsymbol{w}\|_1 &= \|\Delta \boldsymbol{w}\|_1 + \|\boldsymbol{w}' - (\boldsymbol{w} + \Delta \boldsymbol{w})\|_1 + \|\boldsymbol{w}'' - \boldsymbol{w}'\|_1 + \|\boldsymbol{w}''' - \boldsymbol{w}''\|_1 \\
&\leq \|\Delta \boldsymbol{w}\|_1 \\
&\quad + \left(\delta^2 \|\boldsymbol{w} + \Delta \boldsymbol{w}\|_1\right)^{5/2} \cdot 6 \cdot 10^4 \cdot \log \|\boldsymbol{w} + \Delta \boldsymbol{w}\|_1 \\
&\quad + m^{-10} \\
&\quad + \left(10^6 \cdot \delta^2 \|\boldsymbol{w} + \Delta \boldsymbol{w}\|_1 \cdot \log \|\boldsymbol{w}\|_1\right)^2 \cdot p \cdot \|\boldsymbol{w} + \Delta \boldsymbol{w}\|_1^{1/p} \,.
\end{aligned}$$

Finally, the two steps in which the flow demand changes is the weight balancing procedure, in which the change in demand is $\Delta \widetilde{\boldsymbol{d}}$, and the step when we restrict the regularized problem solution to the non-augmented graph. In the latter case, the change in demand $\widetilde{\boldsymbol{d}}$ can be upper bounded by Lemma 3.4.21 by at most 1 in $\ell_1$ norm. Thus the demand will be perturbed by $\|\boldsymbol{d}' - \boldsymbol{d}\|_1 = \|\widetilde{\boldsymbol{d}} + \Delta \widetilde{\boldsymbol{d}}\|_1 \leq 1 + \|\Delta \widetilde{\boldsymbol{d}}\|_1$. This concludes the proof. □

Lemma 3.4.24 is the main workhorse of the improved algorithm. It shows that we can make large progress within the interior point method, while paying for some demand perturbation and for some slight increase in $\|\boldsymbol{w}\|_1$. In order to guarantee sufficient progress, all we are left to do is to ensure that we can set an appropriate $\delta$ such that the sum of weights never increases beyond $O(m)$. This is a mere consequence of the result given above.

**Lemma 3.4.25.** *Suppose we have a $\mu$-central instance with weights $\boldsymbol{w} \geq \mathbf{1}$, where $\|\boldsymbol{w}\|_1 \leq 2m + 1$ and $\mu = m^{O(1)}$. Let $\varepsilon = m^{-O(1)}$, and let $\delta = m^{-(3/8 + o(1))}$. In $\widetilde{O}(\delta^{-1})$ iterations of the procedure described in Lemma 3.4.24 we obtain an instance with duality gap at most $\varepsilon$ with a total demand perturbation of $\widetilde{O}(\delta^{-1})$.*

*Proof.* We perform a sequence of iterations as described in Lemma 3.4.24.

We will argue that within $T = \widetilde{O}(\delta^{-1} \log(m\mu\varepsilon^{-1}))$ iterations of the procedure described in Lemma 3.4.24 the barrier weights will always satisfy $\|\boldsymbol{w}\|_1 \leq 3m$. We need to take into account the total increase guaranteed by Lemma 3.4.24, together with the possible weight increases caused by the weight balancing procedure.

First let us bound the total number of balancing operations, since each of these can increase a single weight by a significant amount. First, from Definition 3.4.22 we see that once an edge gets balanced it will never become unbalanced again, as weights are monotonic.

Furthermore, we see that such an operation can only occur when the largest of the two paired weights is at least $\delta \|\boldsymbol{w}\|_1 \geq 2m\delta$, since we maintain $\boldsymbol{w} \geq \mathbf{1}$. Under the invariant that throughout the entire algorithm $\|\boldsymbol{w}\|_1 \leq 3m$, we therefore see that this can only happen at most $(3m)/(2m\delta) = 3/(2\delta)$ times. Invoking Lemma 3.4.23 we therefore get that the total weight change caused by these operations is at most

$$\frac{3}{2\delta} \cdot 96 \cdot \delta^4 \cdot (3m)^2 = 1296 \cdot \delta^3 m^2 \,.$$

In addition, we incur weight increases due to the progress steps; per Lemma 3.4.24, within each of the $T$ iterations, $\|\boldsymbol{w}\|_1$ increases by at most

$$(\delta^2(3m))^{5/2} \cdot 6 \cdot 10^4 \cdot \log 3m + m^{-10} + 10^{12} \cdot p\delta^4 (3m)^{2+1/p} \cdot \log^2(3m) \,.$$

Therefore the total weight increase over $T$ iterations is at most

$$\widetilde{O}\left(\left(\delta^4 m^{5/2} + p\delta^3 m^{2+1/p}\right)\log(m\mu\varepsilon^{-1})\right)$$

which is $o(m)$ as long as

$$\delta \leq 1/\max\left(m^{3/8+o(1)},\, p^{1/3}m^{(1+1/p)/3+o(1)}\right) = 1/m^{3/8+o(1)}\,.$$

Thus this specific choice of $\delta$ insures that the invariant $\|\boldsymbol{w}\|_1 \leq 3m$ is satisfied throughout the entire run of the algorithm.

Finally, we bound the total perturbation in demand suffered by the flow we maintain. From Lemma 3.4.18 each progress step perturbs the demand by at most 1 in $\ell_1$ norm. Furthermore, the weight balancing operations may perturb it by an additional $\frac{3}{2\delta}$ overall. Summing up we obtain the desired claim. □

Combining with the repairing procedure elaborated in [42], which we show how to adapt to our present setting in Section 3.5, we obtain the main theorem.

**Theorem 3.4.26.** *Given a directed graph $G(V, E, \boldsymbol{c})$ with $m$ arcs and $n$ vertices, such that $\|\boldsymbol{c}\|_\infty \leq W$, and a demand vector $\boldsymbol{d} \in \mathbb{Z}^n$, in $m^{11/8+o(1)}\log W$ time we can obtain a flow $\boldsymbol{f}$ which routes $\boldsymbol{d}$ in $G$ while satisfying the capacity constraints $\mathbf{0} \leq \boldsymbol{f} \leq \mathbf{1}$ and minimizing the cost $\sum_{e\in E} c_e f_e$, or certifies that no such flow exists.*

*Proof.* Before proceeding, we note that Lemma 3.4.25 assumes that the initial centrality parameter $\mu$ is bounded by a polynomial in $m$. This may not be exactly true as the initial centering (Lemma 9.1.2) is done while setting a parameter $\mu$ which is upper bounded by $m^{O(1)}W$. As Lemma 3.4.25 which bounds the number of iterations of our interior point method assumes that initially $\mu = m^{O(1)}$, we need to enforce the property that the method in the lemma is only called on instances where $W = m^{O(1)}$.

Similarly to [42], we enforce this property by employing the scaling technique of [71]. Thus, we reduce the problem to solving $O(\log W)$ instances of the problem, where costs are polynomially bounded. Without doing so, the running time of our algorithm would have depended super-linearly in $\log W$.

For each of the $O(\log W)$ instances, we proceed as follows. First, we apply Lemma 9.1.2 to obtain a $\mu$-central solution for $\mu = m^{O(1)}$. Then, we apply Lemma 3.4.25 with $\varepsilon = m^{-3}$ to obtain a flow $\boldsymbol{f}$ which routes a perturbed demand $\boldsymbol{d'}$ where $\|\boldsymbol{d'} - \boldsymbol{d}\|_1 \leq m^{3/8+o(1)}$, and is close to optimal, in the sense that it is accompanied by dual variables which certify a duality gap of $O(m^{-3})$. Note that each of the $m^{3/8+o(1)}$ iterations of the algorithm from Lemma 3.4.25 requires solving the regularized objective (3.18) to high precision, which due to our choice of parameter $p$ (Corollary 3.4.9) can be done in $m^{1+o(1)}$ time. Finally, we apply Lemma 3.5.4 to repair this flow in $m^{11/8+o(1)}$ time. Hence the total running time is $m^{11/8+o(1)}$.

We can certify infeasibility (the case where the demand can not be routed while satisfying the capacity constraints), by looking at the arcs used by the returned optimal solution. The initialization procedure assumes that a demand satisfying flow can be routed in the graph. This is done by adding to the graph $O(m)$ arcs with high costs (see Lemma 9.1.1). Using Lemma 9.1.1 we see that simply inspecting the optimal solution returned for this graph we can decide if the routing is infeasible in $G$. □

BalanceWeights$(G; \boldsymbol{w}, \boldsymbol{f})$

1. For each $e \in E$:
2. If $w_e^+ < 96 \cdot \delta^4 \|\boldsymbol{w}\|_1^2$ and $w_e^- > \delta \|\boldsymbol{w}\|_1$ then $w_e'^+ \leftarrow 96 \cdot \delta^4 \|\boldsymbol{w}\|_1^2$, $w_e'^- \leftarrow w_e^-$.
3. If $w_e^+ > \delta \|\boldsymbol{w}\|_1$ and $w_e^- < 96 \cdot \delta^4 \|\boldsymbol{w}\|_1^2$ then $w_e'^+ \leftarrow w_e^+$, $w_e'^- \leftarrow 96 \cdot \delta^4 \|\boldsymbol{w}\|_1^2$.
4. Set $\mathbf{0} < \boldsymbol{f}' < \mathbf{1}$ such that $\frac{\boldsymbol{w}'^+}{\mathbf{1}-\boldsymbol{f}'} - \frac{\boldsymbol{w}'^-}{\boldsymbol{f}'} = \frac{\boldsymbol{w}^+}{\mathbf{1}-\boldsymbol{f}} - \frac{\boldsymbol{w}^-}{\boldsymbol{f}}$.
5. Return $\boldsymbol{w}', \boldsymbol{f}'$.

Figure 3-4: Weight balancing procedure

## 3.5 Repairing the Flow

In this section we show that the flow produced in Section 3.4 can be repaired in such a way that we obtain an optimal flow for the original problem. This can be done by applying the combinatorial fixing techniques used in [42]. Let us first recall a few useful definitions concerning the bipartite perfect $\boldsymbol{b}$-matching problem.

**Bipartite Perfect $\boldsymbol{b}$-matching.** For a given weighted bipartite graph $G = (V, E)$ with $V = V_1 \cup V_2$ where $V_1$ and $V_2$ are the two sets of bipartition and a *demand vector* $\boldsymbol{b} \in \mathbb{R}_+^V$, a *perfect $\boldsymbol{b}$-matching* is a vector $\boldsymbol{x} \in \mathbb{R}_+^E$ such that $\sum_{e \in E(v)} x_e = b_v$ for all $v \in V$. A perfect $\boldsymbol{b}$-matching is a generalization of perfect matching; in the particular case where all $\boldsymbol{b}$'s equal 1, integer $\boldsymbol{b}$-matchings (or $\mathbf{1}$-matchings) are exactly perfect matchings. We require that $\boldsymbol{b}(V_1) = \boldsymbol{b}(V_2)$ as otherwise they trivially do not exist.

The *weighted bipartite perfect $\boldsymbol{b}$-matching problem* is defined as follows: given a weighted bipartite graph $G = (V, E, \boldsymbol{c})$, return a perfect $\boldsymbol{b}$-matching in $G$ that has minimum weight, or conclude that there is no perfect $\boldsymbol{b}$-matching in $G$. The dual problem to the weighted perfect bipartite $\boldsymbol{b}$-matching is a *$\boldsymbol{b}$-vertex packing problem* where we want to find a vector $\boldsymbol{y} \in \mathbb{R}^V$ satisfying the following LP

$$\begin{aligned} \max \quad & \sum_{v \in V} y_v b_v, \\ & y_u + y_v \leq w_{uv} \quad \forall u, v \in E. \end{aligned}$$

We employ the following result which follows from a direct application of Lemmas 35, 37, and Theorem 40 in [42].

**Theorem 3.5.1.** *Consider an instance $G = (V, E, \boldsymbol{c})$ of the weighted perfect bipartite $\boldsymbol{b}$-matching problem, where $\|\boldsymbol{b} - \widehat{\boldsymbol{b}}\|_1 \leq P$, and $\|\boldsymbol{b}\|_1 = O(m)$. Given a feasible primal-dual pair of variables $(\boldsymbol{x}, \boldsymbol{y})$ with duality gap at most $m^{-2}$, in $O(Pm + m \log n)$ time we can compute an optimal primal-dual pair $(\widehat{\boldsymbol{x}}, \widehat{\boldsymbol{y}})$ to the perfect $\widehat{\boldsymbol{b}}$-matching problem, or conclude that no such matching exists.*

In order to apply this theorem we need to convert our instance into a $\boldsymbol{b}$-matching instance with small duality gap. The output we receive from Lemma 3.4.25 is a flow $\boldsymbol{f}$ which routes a slightly different demand $\boldsymbol{d}$ from the one we originally intended. Furthermore $\boldsymbol{f}$ satisfies the centrality condition $\boldsymbol{C}^\top \left( \frac{\boldsymbol{w}^-}{\boldsymbol{f}} - \frac{\boldsymbol{w}^+}{\mathbf{1}-\boldsymbol{f}} \right) = \boldsymbol{C}^\top \frac{\boldsymbol{c}}{\mu}$ for a small value of $\mu$.

**From Flow to Bipartite $b$-Matching.** We convert this solution into a corresponding bipartite $b$-matching problem in a new bipartite graph $G'(V_1 \cup V_2, E', \boldsymbol{c'})$, defined as follows:

1. for each vertex $v \in V$, create the corresponding vertex in $V_1$;
2. for each arc $e \in E$, create a vertex $v_e \in V_2$;
3. for each arc $e = (u, v) \in E$ create edges $(u, v_e)$ with cost $c_e$ and $(v, v_e)$ with cost 0.

Let $\boldsymbol{d}$ be the demand routed by $\boldsymbol{f}$. For the bipartite graph $G'$, we define the $\boldsymbol{b}$ vector as follows:

1. for each $v \in V$ set for the corresponding vertex $v$ in $V_1$: $b_v = |E^+(v)| + d_v$.
2. for each $v_2 \in V_2$ set $b_{v_2} = 1$.

We can easily verify that an optimal primal-dual solution to the $\boldsymbol{b}$-matching problem in $G'$ maps to an optimal primal-dual solution to the minimum cost flow problem in $G$.

**Lemma 3.5.2.** *Let $(\boldsymbol{x}, \boldsymbol{y})$ be a pair of feasible primal-dual variables for the $\boldsymbol{b}$-matching problem in the graph $G'$, constructed according to the rules defined above. These can be mapped to a pair $(\boldsymbol{f}, \boldsymbol{z})$ of optimal feasible primal-dual variables for the minimum cost flow problem in $G$. Furthermore, this mapping can be constructed in linear time.*

*Proof.* Let $\boldsymbol{x}$ be the optimal $\boldsymbol{b}$-matching, and let $\boldsymbol{y}$ be its dual vector certifying optimality.

We construct $\boldsymbol{f}$ as follows: for each arc $e = (u, v) \in G$ we look at the corresponding gadget in $G'$ consisting of vertices $u, v \in V_1$ and $v_e \in V_2$ and set $f_e = x_{u,v_e}$.

First we verify that $\boldsymbol{f}$ is feasible. For each arc $e = (u, v) \in G$, the corresponding vertex $v_e \in V_2$ has $b_e = 1$, and therefore $x_{u,v_e} + x_{v,v_e} = 1$. As both are non-negative it means that in $G$, $0 \leq f_e \leq 1$.

Next we verify that $\boldsymbol{f}$ indeed routes the demand it is supposed to route, i.e. $\boldsymbol{d}$. By definition we have that for each $v \in V_1$,

$$\sum_{(v,v_e)\in E'} x_{v,v_e} = b_v = |E^+(v)| + d_v \,.$$

Therefore the demand at vertex $v$ can be written as

$$\begin{aligned}
\sum_{e=(u,v)\in E} f_e - \sum_{e'=(v,u)\in E} f_{e'} &= \sum_{e=(u,v)\in E} (1 - f_e) - |E^+(v)| + \sum_{e'=(v,u)\in E} f_{e'} \\
&= \sum_{(v,v_e)\in E'} x_{v,v_e} - |E^+(v)| \\
&= d_v \,.
\end{aligned}$$

Finally we certify optimality for $\boldsymbol{f}$ by exhibiting feasible dual variables which satisfy complementary slackness. For the LP formulation defined in Section 3.3.1 we require for each arc $e \in E$ two non-negative dual variables $z_e^-$ and $z_e^+$ such that summing up along each cycle

$$z_e^+ - z_e^- + c_e$$

with the appropriate sign depending on the orientation of the encountered arcs, we obtain exactly 0. For each $e = (u, v) \in E$, we define them as

$$\begin{aligned}
z_e^- &= c_e - y_u - y_{v_e} \,, \\
z_e^+ &= -y_v - y_{v_e} \,.
\end{aligned}$$

Feasibility is obvious since $z_e^+, z_e^- \geq 0$ because $\boldsymbol{y}$ is a feasible vector and hence for each edge the sum of the $y$'s of its vertices is upper bounded by the cost, and summing up $z_e^- - z_e^+$ along any cycle we obtain exactly the sum of the costs $c_e$ along that cycle, summed up with the appropriate sign.

We finally certify optimality for $\boldsymbol{f}$ by verifying that $(\boldsymbol{f}, (\boldsymbol{z}^-; \boldsymbol{z}^+))$ satisfy complementary slackness. We have that for each arc $(u, v) \in E$,

$$\begin{aligned} f_e z_e^- &= x_{uv_e}(c_e - y_u - y_{v_e}) = 0 \\ (1 - f_e) z_e^+ &= x_{vv_e}(-y_v - y_{v_e}) = 0\,, \end{aligned}$$

where we used the fact that $(\boldsymbol{x}, \boldsymbol{y})$ satisfy complementary slackness. Hence $\boldsymbol{f}$ is optimal in $G$. □

Our goal will be to obtain an optimal primal-dual solution for a $\widehat{\boldsymbol{b}}$-matching problem which carries over to an optimal solution to the minimum cost flow problem in $G$ with the original demand $\widehat{\boldsymbol{d}}$.

In order to do so, we first convert our flow instance with small duality gap to a $\boldsymbol{b}$-matching instance with the same duality gap. Afterwards, we fix the resulting $\boldsymbol{b}$-matching to optimality and we convert it to an optimal $\widehat{\boldsymbol{b}}$-matching which maps to a flow $\widehat{\boldsymbol{f}}$ routing the original demand $\widehat{\boldsymbol{d}}$ in $G$, via Theorem 3.5.1.

In order to do so, we use the $\mu$-central solution produced by the interior point method in Section 3.4, with $\mu = O(m^{-10})$. Recalling that our current solution satisfies

$$\boldsymbol{C}^\top \left( \frac{\mu \boldsymbol{w}^-}{\boldsymbol{f}} - \frac{\mu \boldsymbol{w}^+}{\mathbf{1} - \boldsymbol{f}} \right) = \boldsymbol{C}^\top \boldsymbol{c}\,,$$

we set primal-dual variables $(\boldsymbol{x}, \boldsymbol{y})$ for $G'$ as follows:

1. for each arc $e = (u, v) \in E$, set $x_{u,v_e} = f_e$ and $x_{v,v_e} = 1 - f_e$.
2. pick an arbitrary vertex $v_0 \in V$ and set its corresponding $y_{v_0} = 0$; then perform a breadth-first search in $G$ (ignoring orientations) starting from $v$, and for each vertex visited for the first time when traversing an arc $e = (u, v)$ (in any direction) such that $y_u - y_v = \frac{\mu w_e^+}{1 - f_e} - \frac{\mu w_e^-}{f_e} - c_e$;
3. for all $v_e \in V_2$ where $e = (u, v) \in E$, set $y_{v_e} = -y_u - \frac{\mu w_e^-}{f_e} + c_e = -y_v - \frac{\mu w_e^-}{1 - f_e}$.

Having defined $G'$ together with its demand vector $\boldsymbol{b}$ and corresponding primal-dual solution $(\boldsymbol{x}, \boldsymbol{y})$, let us show that indeed this is a feasible solution and it has low duality gap.

**Lemma 3.5.3.** *The primal-dual solution $(\boldsymbol{x}, \boldsymbol{y})$ constructed above is feasible and has duality gap $\mu \|\boldsymbol{w}\|_1$.*

*Proof.* First we check feasibility. Primal feasibility holds trivially by construction. For dual feasibility, consider for each arc $e = (u, v) \in E$ the two edges appearing in its corresponding gadget in $G'$, $(u, v_e)$ and $(v, v_e)$. For the former we have

$$y_u + y_{v_e} = -\frac{\mu w_e^-}{f_e} + c_e \leq c_{u,v_e}$$

and for the latter

$$y_v + y_{v_e} = -\frac{\mu w_e^+}{1 - f_e} \leq c_{v,v_e} = 0\,.$$

Therefore the dual constraints of the $\boldsymbol{b}$-matching problem are also satisfied. Finally we write the duality gap via

$$\begin{aligned}\sum_{e=(u,v)\in E'} x_e(c_e - y_u - y_v) &= \sum_{e=(u,v)\in E} (f_e(c_e - y_u - y_{v_e}) + (1-f_e)(0 - y_v - y_{v_e})) \\ &= \sum_{e=(u,v)\in E} \left( f_e \cdot \frac{\mu w_e^-}{f_e} + (1-f_e) \cdot \frac{\mu w_e^+}{1-f_e} \right) \\ &= \sum_{e=(u,v)\in E} \mu(w_e^- + w_e^+) \\ &= \mu \|\boldsymbol{w}\|_1 .\end{aligned}$$

□

Now we are ready to state the main result of this section.

**Lemma 3.5.4.** *Given a target integral demand $\widehat{\boldsymbol{d}}$, a flow $\boldsymbol{f}$ satisfying $\mu$-centrality with weights $\boldsymbol{w}$ such that $\|\boldsymbol{w}\|_1 = O(m)$, $\mu = O(m^{-10})$, and $\boldsymbol{f}$ routes a demand $\boldsymbol{d}$ we can obtain in time $\widetilde{O}(\|\boldsymbol{d} - \widehat{\boldsymbol{d}}\|_1 \cdot m)$ a flow $\widehat{\boldsymbol{f}}$ which is optimal among all flows routing $\widehat{\boldsymbol{d}}$.*

*Proof.* Using Lemma 3.5.3 we can convert the flow instance into a $\boldsymbol{b}$-matching instance with duality gap $O(m^{-2})$. Now we can invoke Theorem 3.5.1 to convert the matching into an optimal $\widehat{\boldsymbol{b}}$-matching where $\widehat{\boldsymbol{b}}$ corresponds to the demand $\widehat{\boldsymbol{d}}$ that the flow is supposed to route in $G$. We see that by construction $\|\boldsymbol{b} - \widehat{\boldsymbol{b}}\|_1 = \|\boldsymbol{d} - \widehat{\boldsymbol{d}}\|_1$. Therefore obtaining the corresponding optimal $\widehat{\boldsymbol{b}}$-matching is done in time $O((\|\boldsymbol{d} - \widehat{\boldsymbol{d}}\|_1 + \log n)m)$. Finally applying Lemma 3.5.2 we obtain the optimal flow $\widehat{\boldsymbol{f}}$ in $G$. □

## 3.6 Improving the Running Time to $m^{4/3+o(1)} \log W$

The running time of the algorithm we presented above hits a barrier of $m^{11/8+o(1)} \log W$. The key bottleneck there is the post-processing we do on the residual error of the (intermediate) solutions we obtain after performing each progress step. Indeed, while the length of our steps is dictated by the $\ell_\infty$-norm of our step size $\|\boldsymbol{\rho}\|_\infty$ (which is, in a sense optimal), it is unclear how to ensure that the energy required to route a flow that fixes the corresponding residual error is sufficiently small, without overly increasing the weights of the constraint barriers. In fact, the extent of weight perturbations necessary for this post-processing step are exactly what determines the $m^{11/8+o(1)} \log W$ running time. All the other weight perturbations, which are caused by the regularization terms, are much milder and would lead to the desired $m^{4/3+o(1)} \log W$ running time.

After the first version of [10] was posted, Liu and Sidford [114] published a preprint that obtains an improved running time for the unit-capacity maximum flow. The main technique introduced in that paper boils down exactly to avoiding the aforementioned bottleneck. Roughly, instead of advancing from a central point to the next one via a progress step followed by a sequence of residual correction steps, they instead directly solve the optimization problem which lands them at the next central point.

To do so, one must guarantee that this optimization problem is well-conditioned at all times, in the sense that the objective has a Hessian which always stays within a constant factor from the one at the origin. While such a Hessian stability condition is not true in general, Liu and Sidford [114] modify the logarithmic barriers they use by extending them with quadratics outside the region

where they would be naturally well-behaved. The resulting new objective function can be efficiently optimized by slightly extending the mixed $\ell_2$-$\ell_p$ solver of Kyng et al [103] (see Appendix 9.1.4).

Incorporating this idea in our framework yields a the desired running time improvement of our unit-capacity minimum cost flow algorithm as well. Most of the details, particularly those involving preconditioning and weight perturbations carry over from the previous sections. In fact, the only change to the algorithm needed is to replace the regularized Newton step (cf. line 3 in Figure 3-3) with solving a regularized problem which directly involves the logarithmic barrier. (Furthermore, lines 7 and 8 are no longer required.)

**Method Overview.** Let us specify the ideal optimization problem which we would solve in order to advance along the central path. Suppose we have a $\mu$-central instance i.e. we have a flow $\boldsymbol{f} = \boldsymbol{f}_0 + \boldsymbol{C}\boldsymbol{x}$ which satisfies

$$\boldsymbol{C}^\top \left( \frac{\boldsymbol{w}^+}{\mathbf{1} - \boldsymbol{f}_0 - \boldsymbol{C}\boldsymbol{x}} - \frac{\boldsymbol{w}^-}{\boldsymbol{f}_0 + \boldsymbol{C}\boldsymbol{x}} \right) = -\frac{\boldsymbol{C}^\top \boldsymbol{c}}{\mu} \,. \tag{3.60}$$

as it optimizes the convex objective

$$\min_{\boldsymbol{x}} F_\mu^{\boldsymbol{w}}(\boldsymbol{x}) = \min_{\boldsymbol{x}} \frac{1}{\mu} \langle \boldsymbol{c}, \boldsymbol{C}\boldsymbol{x} \rangle - \sum_{e \in E} \left( w_e^+ \log(\mathbf{1} - \boldsymbol{f}_0 - \boldsymbol{C}\boldsymbol{x})_e + w_e^- \log(\boldsymbol{f}_0 + \boldsymbol{C}\boldsymbol{x})_e \right) .$$

Our goal is to design an optimization procedure which enables us to augment $\boldsymbol{f}$ with a circulation $\boldsymbol{C}\widetilde{\boldsymbol{x}}$ in order to obtain an optimizer for

$$\begin{aligned} \min_{\boldsymbol{x}'} F^{\boldsymbol{w}}_{\mu/(1+\delta)}(\boldsymbol{x}') &= \min_{\boldsymbol{x}'} \frac{1+\delta}{\mu} \langle \boldsymbol{c}, \boldsymbol{C}\boldsymbol{x}' \rangle - \sum_{e \in E} \left( w_e^+ \log(\mathbf{1} - \boldsymbol{f}_0 - \boldsymbol{C}\boldsymbol{x}')_e + w_e^- \log(\boldsymbol{f}_0 + \boldsymbol{C}\boldsymbol{x}')_e \right) \\ &= \min_{\widetilde{\boldsymbol{x}}} \frac{1+\delta}{\mu} \langle \boldsymbol{c}, \boldsymbol{f} + \boldsymbol{C}\widetilde{\boldsymbol{x}} \rangle - \sum_{e \in E} \left( w_e^+ \log(\mathbf{1} - \boldsymbol{f} - \boldsymbol{C}\widetilde{\boldsymbol{x}})_e + w_e^- \log(\boldsymbol{f} + \boldsymbol{C}\widetilde{\boldsymbol{x}})_e \right) . \end{aligned}$$

We can equivalently rewrite this optimization procedure, after adding a constant term to it, as

$$\min_{\widetilde{\boldsymbol{x}}} \Psi^{\boldsymbol{w},\boldsymbol{f}}_{\mu/(1+\delta)}(\widetilde{\boldsymbol{x}}) \,,$$

where

$$\begin{aligned} \Psi^{\boldsymbol{w},\boldsymbol{f}}_{\mu/(1+\delta)}(\widetilde{\boldsymbol{x}}) :&= F^{\boldsymbol{w}}_{\mu/(1+\delta)}(\boldsymbol{x} + \widetilde{\boldsymbol{x}}) - \frac{1+\delta}{\mu} \langle \boldsymbol{c}, \boldsymbol{f} \rangle + \sum_{e \in E} \left( w_e^+ \log(1 - f_e) + w_e^- \log f_e \right) \\ &= \frac{1+\delta}{\mu} \langle \boldsymbol{c}, \boldsymbol{C}\widetilde{\boldsymbol{x}} \rangle - \sum_{e \in E} \left( w_e^+ \log \left( \mathbf{1} - \frac{\boldsymbol{C}\widetilde{\boldsymbol{x}}}{\mathbf{1} - \boldsymbol{f}} \right)_e + w_e^- \log \left( \mathbf{1} + \frac{\boldsymbol{C}\widetilde{\boldsymbol{x}}}{\boldsymbol{f}} \right)_e \right) . \end{aligned}$$

Optimizing this function is hard to do in general. However, assuming its optimal augmenting circulation $\boldsymbol{C}\widetilde{\boldsymbol{x}}$ satisfies

$$\left\| \frac{\boldsymbol{C}\widetilde{\boldsymbol{x}}}{\min\{\boldsymbol{f}, \mathbf{1} - \boldsymbol{f}\}} \right\|_\infty \leq \frac{1}{10} \,, \tag{3.61}$$

which in other words, says that augmenting $\boldsymbol{f}$ with $\boldsymbol{C}\widetilde{\boldsymbol{x}}$ will not come close to breaking the feasibility constraints, we can instead solve a well-conditioned objective obtained by replacing the log's with a better behaved function $\widetilde{\log}$ satisfying $\log(1+t) = \widetilde{\log}(1+t)$ whenever $|t| \leq 1/10$.

More precisely, we use the definition from [114] which we reproduce below:

$$\widetilde{\log}(1+t) = \begin{cases} \log(1+t), & \text{if } t \in [-\theta, \theta]\,, \\ \log(1+\theta) + (t-\theta)\cdot \log'(1+\theta) + (t-\theta)^2 \cdot \frac{1}{2}\log''(1+\theta), & \text{if } t > \theta\,, \\ \log(1-\theta) + (t+\theta)\cdot \log'(1-\theta) + (t+\theta)^2 \cdot \frac{1}{2}\log''(1-\theta), & \text{if } t < -\theta\,, \end{cases}$$

where we set $\theta = 1/10$. We can also easily verify that on the boundary of the interval $[-\theta, \theta]$ the first and second order derivatives exactly match those of $\log(1+t)$. The essential feature is that outside this range, the second derivative stays constant, whereas in the case of $\log(1+t)$ it changes very fast.

Using this, we can define the minimization problem

$$\min_{\widetilde{\boldsymbol{x}}} \widetilde{\Psi}^{\boldsymbol{w},\boldsymbol{f}}_{\mu/(1+\delta)}(\widetilde{\boldsymbol{x}})\,,$$

where

$$\widetilde{\Psi}^{\boldsymbol{w},\boldsymbol{f}}_{\mu/(1+\delta)}(\widetilde{\boldsymbol{x}}) := \frac{1+\delta}{\mu}\langle \boldsymbol{c}, \boldsymbol{C}\widetilde{\boldsymbol{x}}\rangle - \sum_{e\in E}\left( w_e^{+}\widetilde{\log}\left(\mathbf{1} - \frac{\boldsymbol{C}\widetilde{\boldsymbol{x}}}{\mathbf{1}-\boldsymbol{f}}\right)_e + w_e^{-}\widetilde{\log}\left(\mathbf{1} + \frac{\boldsymbol{C}\widetilde{\boldsymbol{x}}}{\boldsymbol{f}}\right)_e\right). \tag{3.62}$$

**Observation 3.6.1.** *If the minimizer $\widetilde{\boldsymbol{x}}$ of $\widetilde{\Psi}^{\boldsymbol{w},\boldsymbol{f}}_{\mu/(1+\delta)}(\widetilde{\boldsymbol{x}})$ satisfies the low-congestion condition from (3.61), then it also minimizes $\Psi^{\boldsymbol{w},\boldsymbol{f}}_{\mu/(1+\delta)}(\widetilde{\boldsymbol{x}})$.*

Due to the fact that the second order derivatives of $\widetilde{\log}$ are bounded, $\widetilde{\Psi}^{\boldsymbol{w},\boldsymbol{f}}_{\mu/(1+\delta)}(\widetilde{\boldsymbol{x}})$ is well-conditioned and, as a matter of fact can easily be minimized by using a small number of calls to a routine which minimizes quadratics over the set of circulations. As specified in [114] this can be easily done by using fast Laplacian system solvers. However, the main difficulty that arises is ensuring that the minimizer $\widetilde{\boldsymbol{x}}$ satisfies the confition from (3.61).

Enforcing this property is non-trivial, and requires regularizing the function $\widetilde{\Psi}^{\boldsymbol{w},\boldsymbol{f}}_{\mu/(1+\delta)}$ in an identical manner to the way we did it in the analysis from Section 3.4. Most of the results we used there carry over, after performing some minor modifications in the analysis.

**Regularizing the Objective.** In order to enforce the required property, we add two regularization terms to our optimization step. Just like in Section 3.4.1 we augment the graph $G$ to $G_\star$, which has a cycle basis $\boldsymbol{C}_\star$, and in this graph we write down the equivalent concave maximization problem for (3.62), which we regularize with two extra terms. We slightly abuse notation by making $\widetilde{\Psi}^{\boldsymbol{w},\boldsymbol{f}}_{\mu/(1+\delta)}$ act on an element in the circulation space of $G_\star$, with the meaning that the linear and logarithmic terms only act on edges in $E$:

$$\max_{\widetilde{\boldsymbol{f}}_\star = \boldsymbol{C}_\star\widetilde{\boldsymbol{x}}} -\widetilde{\Psi}^{\boldsymbol{w},\boldsymbol{f}}_{\mu/(1+\delta)}(\widetilde{\boldsymbol{x}}) - \frac{R_\star}{2}\sum_{e\in E'}(\widetilde{f}_\star)_e^2 - \frac{R_p}{p}\sum_{e\in E\cup E'}(\widetilde{f}_\star)_e^p\,. \tag{3.63}$$

Writing $\widetilde{\boldsymbol{f}}_\star = \widetilde{\boldsymbol{f}} + \widetilde{\boldsymbol{f}}'$ where $\widetilde{\boldsymbol{f}}$ is the restriction of $\widetilde{\boldsymbol{f}}_\star$ to the edges $E$ of $G$, and $\widetilde{\boldsymbol{f}}'$ is the restriction

to the augmenting edges $E'$, and using the centrality condition (3.60), we can further write this as

$$\max_{\widetilde{\boldsymbol{f}}_\star = \boldsymbol{C}_\star \widetilde{\boldsymbol{x}}} (1+\delta) \left\langle \frac{\boldsymbol{w}^+}{\mathbf{1}-\boldsymbol{f}} - \frac{\boldsymbol{w}^-}{\boldsymbol{f}}, \widetilde{\boldsymbol{f}} \right\rangle + \sum_{e \in E} \left( w_e^+ \widetilde{\log}\left(1 - \frac{\widetilde{f}_e}{1-f_e}\right) + w_e^- \widetilde{\log}\left(1 + \frac{\widetilde{f}_e}{f_e}\right)\right) - \frac{R_\star}{2} \sum_{e \in E'} (\widetilde{f}'_e)^2 - \frac{R_p}{p} \sum_{e \in E \cup E'} (\widetilde{f}_\star)_e^p \,. \tag{3.64}$$

Note that this optimization problem can be solved efficiently using Lemma 9.1.7. The key reason is that all the terms except for the one involving $p$ powers of the flow $\widetilde{f}$ have a second order derivative which is either constant, or which stays bounded within a small multiplicative factor from the one at 0. Similarly to before, the solver from Lemma 9.1.7 yields a high-accuracy, yet inexact solution. We can, however, assume we obtain an exact solution by performing minor perturbations to our problem, in a manner similar to the one discussed in Section 9.1.3.

We can now write the first order optimality condition for the objective in (3.63), thus providing an analogue of Lemma 3.4.6. Before doing so, we give the following helper lemma, which will be the main driver of the results in this section. It intuitively states that the optimal solution to (3.64) can be thought of as the solution to a regularized Newton step as the one in (3.18), but where the coefficients of the quadratic $\frac{1}{2} \sum_{e \in E} \widetilde{f}_e^2 \cdot \left( \frac{w_e^+}{(s_e^+)^2} + \frac{w_e^-}{(s_e^-)^2} \right)$ have been slightly perturbed by a small multiplicative constant.

**Lemma 3.6.2** (Optimality with average Hessian)**.** *Let $\widetilde{\boldsymbol{f}}_\star = \boldsymbol{C}_\star \widetilde{\boldsymbol{x}}_\star$ be the optimizer of (3.64), and let $\widetilde{\boldsymbol{f}}_\star = \widetilde{\boldsymbol{f}} + \widetilde{\boldsymbol{f}}'$, where the two components are supported on $E$ and $E'$, respectively. Then there exists a vector $\boldsymbol{\alpha} = (\boldsymbol{\alpha}^+; \boldsymbol{\alpha}^-)$, $(1+\theta)^{-2} \cdot \mathbf{1} \le \boldsymbol{\alpha} \le (1-\theta)^{-2} \cdot \mathbf{1}$, which can be explicitly computed, such that for any circulation $\boldsymbol{g}$ in $\boldsymbol{C}_\star$,*

$$\left\langle \boldsymbol{g}, \begin{bmatrix} \delta \left( \frac{\boldsymbol{w}^+}{\mathbf{1}-\boldsymbol{f}} - \frac{\boldsymbol{w}^-}{\boldsymbol{f}} \right) - \widetilde{\boldsymbol{f}} \left( \frac{\boldsymbol{\alpha}^+ \boldsymbol{w}^+}{(\mathbf{1}-\boldsymbol{f})^2} + \frac{\boldsymbol{\alpha}^- \boldsymbol{w}^-}{\boldsymbol{f}^2} \right) - R_p \cdot (\widetilde{\boldsymbol{f}})^{p-1} \\ -R_\star \cdot \widetilde{\boldsymbol{f}}' - R_p \cdot (\widetilde{\boldsymbol{f}}')^{p-1} \end{bmatrix} \right\rangle = 0 \,.$$

*Proof.* We write the first order optimality condition for (3.64). We have that for any circulation $\boldsymbol{g}$ in $G_\star$:

$$\left\langle \boldsymbol{g}, \begin{bmatrix} (1+\delta) \left( \frac{\boldsymbol{w}^+}{\mathbf{1}-\boldsymbol{f}} - \frac{\boldsymbol{w}^-}{\boldsymbol{f}} \right) + \boldsymbol{w}^+ \cdot \nabla_{\widetilde{\boldsymbol{f}}} \widetilde{\log}\left( \mathbf{1} - \frac{\widetilde{\boldsymbol{f}}}{\mathbf{1}-\boldsymbol{f}} \right) + \boldsymbol{w}^- \cdot \nabla_{\widetilde{\boldsymbol{f}}} \widetilde{\log}\left( \mathbf{1} + \frac{\widetilde{\boldsymbol{f}}}{\boldsymbol{f}} \right) - R_p \cdot (\widetilde{\boldsymbol{f}})^{p-1} \\ -R_\star \cdot \widetilde{\boldsymbol{f}}' - R_p \cdot (\widetilde{\boldsymbol{f}}')^{p-1} \end{bmatrix} \right\rangle = 0 \,.$$

Next we use the expansion

$$\frac{d}{dx} \widetilde{\log}\left(1 + \frac{x}{a}\right) = \frac{1}{a} \cdot \widetilde{\log}'\left(1 + \frac{x}{a}\right) = \frac{1}{a} \cdot \left(1 + \frac{x}{a} \int_0^1 \widetilde{\log}''\left(1 + t \cdot \frac{x}{a}\right) dt\right) \,.$$

From the definition of $\widetilde{\log}$ we know that its second order derivative is always stable which enables us to bound

$$-\frac{1}{(1-\theta)^2} \le \int_0^1 \widetilde{\log}''\left(1 + t \cdot \frac{x}{a}\right) dt \le -\frac{1}{(1+\theta)^2} \,,$$

hence for some $\alpha \in [(1+\theta)^{-2}, (1-\theta)^{-2}]$, one has

$$\frac{d}{dx} \widetilde{\log}\left(1 + \frac{x}{a}\right) = \frac{1}{a} \cdot \widetilde{\log}'\left(1 + \frac{x}{a}\right) = \frac{1}{a} - \alpha \cdot \frac{x}{a^2} \,.$$

Therefore there exists a vector $\boldsymbol{\alpha} = (\boldsymbol{\alpha}^+; \boldsymbol{\alpha}^-)$ satisfying $(1+\theta)^{-2} \cdot \mathbf{1} \leq \boldsymbol{\alpha} \leq (1-\theta)^{-2} \cdot \mathbf{1}$ such that

$$\left\langle \boldsymbol{g}, \begin{bmatrix} (1+\delta)\left(\frac{\boldsymbol{w}^+}{\mathbf{1}-\boldsymbol{f}} - \frac{\boldsymbol{w}^-}{\boldsymbol{f}}\right) + \boldsymbol{w}^+ \cdot \nabla_{\widetilde{\boldsymbol{f}}} \widetilde{\log}\left(\mathbf{1} - \frac{\widetilde{\boldsymbol{f}}}{\mathbf{1}-\boldsymbol{f}}\right) + \boldsymbol{w}^- \cdot \nabla_{\widetilde{\boldsymbol{f}}} \widetilde{\log}\left(\mathbf{1} + \frac{\widetilde{\boldsymbol{f}}}{\boldsymbol{f}}\right) - R_p \cdot (\widetilde{\boldsymbol{f}})^{p-1} \\ -R_\star \cdot \widetilde{\boldsymbol{f}}' - R_p \cdot (\widetilde{\boldsymbol{f}}')^{p-1} \end{bmatrix} \right\rangle$$

$$= \left\langle \boldsymbol{g}, \begin{bmatrix} \delta\left(\frac{\boldsymbol{w}^+}{\mathbf{1}-\boldsymbol{f}} - \frac{\boldsymbol{w}^-}{\boldsymbol{f}}\right) - \widetilde{\boldsymbol{f}}\left(\frac{\boldsymbol{\alpha}^+\boldsymbol{w}^+}{(\mathbf{1}-\boldsymbol{f})^2} + \frac{\boldsymbol{\alpha}^-\boldsymbol{w}^-}{\boldsymbol{f}^2}\right) - R_p \cdot (\widetilde{\boldsymbol{f}})^{p-1} \\ -R_\star \cdot \widetilde{\boldsymbol{f}}' - R_p \cdot (\widetilde{\boldsymbol{f}}')^{p-1} \end{bmatrix} \right\rangle = 0\,,$$

which is what we needed.

□

Lemma 3.6.2 enables us to use an optimality condition very similar to the one we had before in Section 3.4. As a matter of fact, all the remaining statements are nothing but "robust" versions of those we previously used. Essential here are new versions of Lemma 3.4.6 and Lemma 3.4.7 which accommodate the extra multiplicative factors on resistances. Roughly, our goal is to provide upper bounds on $\|\widetilde{\boldsymbol{f}}\|_\infty$ and $\|\widetilde{\boldsymbol{f}}/\min\{\mathbf{1}-\boldsymbol{f}, \boldsymbol{f}\}^2\|_\infty$, which together will imply that the condition from (3.61) is satisfied.

After proving that this is the case, we will show how to advance to the next point on the central path – the regularization terms on (3.64) will require us to increase the weights $\boldsymbol{w}$ in order to obtain optimality for the non-regularized objective i.e. $\nabla \widetilde{\Psi}^{\boldsymbol{w},\boldsymbol{f}}_{\mu/(1+\delta)}(\widetilde{\boldsymbol{x}}) = 0$. Nevertheless, this procedure is essentially identical to the one we previously used in Lemma 3.4.20.

### 3.6.1 Bounding Congestion

Here we show that the congestion condition from (3.61) is satisfied, and hence the minimizer of (3.64) also minimizes the expression after replacing $\widetilde{\log}$ with log. The proofs are almost identical to those from Section 3.4. For consistency we will use the slack notation

$$\boldsymbol{s}^- = \boldsymbol{f}\,, \quad \boldsymbol{s}^+ = \mathbf{1} - \boldsymbol{f}\,,$$

and use the shorthand notation for the residual

$$\boldsymbol{h} = \delta\left(\frac{\boldsymbol{w}^+}{\boldsymbol{s}^+} - \frac{\boldsymbol{w}^-}{\boldsymbol{s}^-}\right).$$

We first give a short lemma providing an optimality condition for the restriction of the flow $\widetilde{\boldsymbol{f}}_\star$ computed by (3.64) to the edges $E$ of the original graph $G$.

**Lemma 3.6.3** (Optimality in the non-augmented graph)**.** *Let $\widetilde{\boldsymbol{f}}_\star = \boldsymbol{C}_\star \widetilde{\boldsymbol{x}}_\star$ be the optimizer of (3.63) and let $\widetilde{\boldsymbol{f}}$ be its restriction to the edges of $G$. Let $\widetilde{\boldsymbol{d}}$ be the demand routed by $\widetilde{\boldsymbol{f}}$ in $G$. Then there exists a vector $\boldsymbol{\alpha} = (\boldsymbol{\alpha}^+; \boldsymbol{\alpha}^-) \in \mathbb{R}^{2m}$, $\frac{1}{(1+\theta)^2} \cdot \mathbf{1} \leq \boldsymbol{\alpha} \leq \frac{1}{(1-\theta)^2} \cdot \mathbf{1}$, which can be explicitly computed, such that*

$$\boldsymbol{C}^\top \cdot \left(\frac{\boldsymbol{\alpha}^+\boldsymbol{w}^+}{(\boldsymbol{s}^+)^2} + \frac{\boldsymbol{\alpha}^-\boldsymbol{w}^-}{(\boldsymbol{s}^-)^2}\right) \cdot \widetilde{\boldsymbol{f}} = \boldsymbol{C}^\top(\boldsymbol{h} + \Delta\boldsymbol{h}) \tag{3.65}$$

*where* $\Delta \boldsymbol{h} = -R_p(\widetilde{\boldsymbol{f}})^{p-1}$, *and*

$$\|\widetilde{\boldsymbol{d}}\|_1 \le 3\left(\frac{\|\boldsymbol{w}\|_1 \cdot \mathcal{E}^{\max}(\boldsymbol{h},\boldsymbol{w},\boldsymbol{s})}{R_\star}\right)^{1/2},$$
$$\|\widetilde{\boldsymbol{f}}_\star\|_p \le \left(\frac{p\cdot\frac{3}{2}\mathcal{E}^{\max}(\boldsymbol{h},\boldsymbol{w},\boldsymbol{s})}{R_p}\right)^{1/p}.$$

The proof can be found in Appendix 9.1.5.

Next we provide a guarantee enforced by the component of the regularizer involving augmenting edges.

**Lemma 3.6.4.** *Let* $\widetilde{\boldsymbol{f}}_\star$ *be the solution of the regularized objective and* $\widetilde{\boldsymbol{f}}$ *its restriction on* $G$*, and suppose that* $\|\boldsymbol{w}\|_1 \ge 3$*. Then there exists a vector* $\boldsymbol{\alpha} = (\boldsymbol{\alpha}^+;\boldsymbol{\alpha}^-) \in \mathbb{R}^{2m}$ *such that for all edges* $e \in E$*:*

$$\left|\left(\frac{\alpha_e^+ w_e^+}{(s_e^+)^2} + \frac{\alpha_e^- w_e^-}{(s_e^-)^2} + R_p\cdot \widetilde{f}_e^{p-2}\right)\widetilde{f}_e - h_e\right| \le \hat{\gamma}\,, \tag{3.66}$$

*where*

$$\hat{\gamma} = \left(R_\star + R_p\cdot\|\widetilde{\boldsymbol{f}}_\star\|_\infty^{p-2}\right)^{1/2}\cdot\left\|\frac{\boldsymbol{h}}{\sqrt{(\boldsymbol{w}^+ + \boldsymbol{w}^-)\left(\frac{\boldsymbol{\alpha}^+\boldsymbol{w}^+}{(\boldsymbol{s}^+)^2} + \frac{\boldsymbol{\alpha}^-\boldsymbol{w}^-}{(\boldsymbol{s}^-)^2}\right)}}\right\|_\infty \cdot 32\log\|\boldsymbol{w}\|_1\,.$$

*Furthermore, this implies that*

$$\left(\frac{\alpha_e^+ w_e^+}{(s_e^+)^2} + \frac{\alpha_e^- w_e^-}{(s_e^-)^2}\right)\cdot\left|\widetilde{f}_e\right| \le |h_e| + \hat{\gamma}\,. \tag{3.67}$$

The proof is essentially identical to that of Lemma 3.4.7. We discuss it at the end of Appendix 9.1.2.

Combining Lemmas 3.6.3 and 3.6.4 we can finally prove the main statement of this section. This will be true for as long as the condition we specified in Definition 3.4.13 holds. This condition will be later enforced by performing the balancing operation we also specified in Definition 3.4.22.

In order to obtain the desired congestion bound, we set our regularization parameters identically to Definition 3.4.8:

$$p = \min\left\{k\in 2\mathbb{Z} : k \ge (\log m)^{1/3}\right\},$$
$$R_p = p\cdot\left(10^6\cdot\delta^2\|\boldsymbol{w}\|_1\cdot\log\|\boldsymbol{w}\|_1\right)^{p+1},$$
$$R_\star = 3\cdot\delta^2\|\boldsymbol{w}\|_1^2\,.$$

**Lemma 3.6.5** (Congestion bound)**.** *Suppose that all edges* $e \in E$ *are balanced, per Definition 3.4.13, and* $\delta > 10\cdot\|\boldsymbol{w}\|_1^{-1/2}$*. Then the restriction* $\widetilde{\boldsymbol{f}}$ *of the flow* $\widetilde{\boldsymbol{f}}_\star$ *computed via (3.64) satisfies the low-congestion condition (3.61).*

*Proof.* Lemma 3.4.15 proves that if all edges are balanced, for any flow $\widehat{\boldsymbol{f}}$ in $G$ such that for all $e \in E$

$$\left(\frac{w_e^+}{(s_e^+)^2} + \frac{w_e^-}{(s_e^-)^2}\right)\cdot\left|\widehat{f}_e\right| \le |h_e| + \hat{\gamma}\,,$$

where $\boldsymbol{h} = \delta\left(\frac{\boldsymbol{w}^+}{\boldsymbol{s}^+} - \frac{\boldsymbol{w}^-}{\boldsymbol{s}^-}\right)$, then for each $e \in E$ it is true that $\max\{|\widehat{\rho}_e^+|, |\widehat{\rho}_e^-|\} < C_\infty$ or $\left|\frac{w_e^+\widehat{\rho}_e^+}{s_e^+}\right| + \left|\frac{w_e^-\widehat{\rho}_e^-}{s_e^-}\right| \leq 6\gamma$, where $\widehat{\rho}_e^+ = \frac{\widehat{f}_e}{s_e^+}$, $\widehat{\rho}_e^- = \frac{-\widehat{f}_e}{s_e^-}$, and $C_\infty = \frac{1}{2\delta\sqrt{2\|\boldsymbol{w}\|_1}}$. We apply this statement for

$$\left|\widehat{f}_e\right| := \frac{\frac{\alpha_e^+ w_e^+}{(s_e^+)^2} + \frac{\alpha_e^- w_e^-}{(s_e^-)^2}}{\frac{w_e^+}{(s_e^+)^2} + \frac{w_e^-}{(s_e^-)^2}} \left|\widetilde{f}_e\right| \geq \frac{1}{(1+\theta)^2}\left|\widetilde{f}_e\right|,$$

which also implies that $|\rho_e^+| = \left|\frac{\widetilde{f}_e}{s_e^+}\right| \leq (1+\theta)^2\,|\widehat{\rho}_e^+|$ , $|\rho_e^-| = \left|\frac{\widetilde{f}_e}{s_e^-}\right| \leq (1+\theta)^2\,|\widehat{\rho}_e^-|$. Note that we now have

$$\left(\frac{w_e^+}{(s_e^+)^2} + \frac{w_e^-}{(s_e^-)^2}\right)\left|\widehat{f}_e\right| = \left(\frac{\alpha_e^+ w_e^+}{(s_e^+)^2} + \frac{\alpha_e^- w_e^-}{(s_e^-)^2}\right) \cdot \left|\widetilde{f}_e\right| \leq |h_e| + \hat{\gamma},$$

and thus we get that for all $e \in E$ at least one of

$$\max\left\{\left|\rho_e^+\right|, \left|\rho_e^-\right|\right\} \leq (1+\theta)^2 \cdot \max\left\{\left|\widehat{\rho}_e^+\right|, \left|\widehat{\rho}_e^-\right|\right\} < (1+\theta)^2 \cdot C_\infty$$

and

$$\left|\frac{w_e^+\rho_e^+}{s_e^+}\right| + \left|\frac{w_e^-\rho_e^-}{s_e^-}\right| \leq (1+\theta)^2 \cdot \left(\left|\frac{w_e^+\widehat{\rho}_e^+}{s_e^+}\right| + \left|\frac{w_e^-\widehat{\rho}_e^-}{s_e^-}\right|\right) \leq 6 \cdot (1+\theta)^2 \cdot \gamma$$

is true. Using the fact that $\delta > 10 \cdot \|\boldsymbol{w}\|_1^{-1/2}$, the former becomes $\max\{|\rho_e^+|, |\rho_e^-|\} = \frac{(1+\theta)^2}{2\delta\sqrt{2\|\boldsymbol{w}\|_1}} < \frac{1}{20}$. For all the remaining edges, we have

$$\left|\frac{w_e^+\rho_e^+}{s_e^+}\right| + \left|\frac{w_e^-\rho_e^-}{s_e^-}\right| \leq 6 \cdot (1+\theta)^2 \cdot \gamma = 6 \cdot (1+\theta)^2 \cdot \delta^2\|\boldsymbol{w}\|_1 \cdot 32\sqrt{6} \cdot \log\|\boldsymbol{w}\|_1,$$

which, combined with

$$\begin{aligned}\left|\widetilde{f}_e\right| \leq \|\widetilde{\boldsymbol{f}}\|_p &\leq \left(\frac{p \cdot \frac{3}{2}\mathcal{E}^{\max}(\boldsymbol{h}, \boldsymbol{w}, \boldsymbol{s})}{R_p}\right)^{1/p} \\ &\leq \left(\frac{p \cdot \frac{3}{4}\delta^2\|\boldsymbol{w}\|_1}{p\,(10^6 \cdot \delta^2\|\boldsymbol{w}\|_1 \log\|\boldsymbol{w}\|_1)^{p+1}}\right)^{1/p} \\ &\leq \frac{1}{10^6 \cdot \delta^2\|\boldsymbol{w}\|_1 \log\|\boldsymbol{w}\|_1}\end{aligned}$$

gives

$$\begin{aligned}\left|\rho_e^+\right| &= \left(\left|\widetilde{f}_e\right| \cdot \left|\frac{\rho_e^+}{s_e^+}\right|\right)^{1/2} \\ &\leq \left(\frac{1}{10^6 \cdot \delta^2\|\boldsymbol{w}\|_1 \log\|\boldsymbol{w}\|_1} \cdot 6 \cdot (1+\theta)^2 \cdot \delta^2\|\boldsymbol{w}\|_1 \cdot 32\sqrt{6}\log\|\boldsymbol{w}\|_1\right)^{1/2} \\ &< \frac{1}{20}.\end{aligned}$$

Symmetrically, we get that $|\rho_e^-| < \frac{1}{20}$ and so we are done. □

### 3.6.2 Making Progress

Here we show how to use the flow obtained from optimizing (3.64) in order to achieve centrality for a new parameter $\mu/(1+\delta)$, at the expense of slightly increasing weights from $\boldsymbol{w}$ to some $\boldsymbol{w}'$.

**Lemma 3.6.6** (Almost-centrality after executing step)**.** *Let $\widetilde{\boldsymbol{f}}_\star$ be the optimizer of (3.64) and let $\widetilde{\boldsymbol{f}}$ be its restriction to the edges of $G$. Then*

$$\boldsymbol{C}^\top \left( (1+\delta) \left( \frac{\boldsymbol{w}^+}{\mathbf{1}-\boldsymbol{f}} - \frac{\boldsymbol{w}^-}{\boldsymbol{f}} \right) - \left( \frac{\boldsymbol{w}^+}{\mathbf{1}-\boldsymbol{f}-\widetilde{\boldsymbol{f}}} - \frac{\boldsymbol{w}^-}{\boldsymbol{f}+\widetilde{\boldsymbol{f}}} \right) \right) = \boldsymbol{C}^\top \cdot R_p \cdot (\widetilde{\boldsymbol{f}})^{p-1} .$$

*Proof.* Writing the optimality condition for (3.64), and using Lemma 3.6.5, we obtain

$$\boldsymbol{C}^\top \left( (1+\delta) \left( \frac{\boldsymbol{w}^+}{\mathbf{1}-\boldsymbol{f}} - \frac{\boldsymbol{w}^-}{\boldsymbol{f}} \right) + \boldsymbol{w}^+ \cdot \nabla_{\widetilde{\boldsymbol{f}}} \log \left( \mathbf{1} - \frac{\widetilde{\boldsymbol{f}}}{\mathbf{1}-\boldsymbol{f}} \right) + \boldsymbol{w}^- \cdot \nabla_{\widetilde{\boldsymbol{f}}} \log \left( \mathbf{1} + \frac{\widetilde{\boldsymbol{f}}}{\boldsymbol{f}} \right) - R_p \cdot (\widetilde{\boldsymbol{f}})^{p-1} \right) = 0 ,$$

which we can rewrite equivalently as

$$\boldsymbol{C}^\top \left( (1+\delta) \left( \frac{\boldsymbol{w}^+}{\mathbf{1}-\boldsymbol{f}} - \frac{\boldsymbol{w}^-}{\boldsymbol{f}} \right) - \left( \frac{\boldsymbol{w}^+}{\mathbf{1}-\boldsymbol{f}-\widetilde{\boldsymbol{f}}} - \frac{\boldsymbol{w}^-}{\boldsymbol{f}+\widetilde{\boldsymbol{f}}} \right) \right) = \boldsymbol{C}^\top \cdot R_p \cdot (\widetilde{\boldsymbol{f}})^{p-1} ,$$

which is what we needed.

□

As we can see, the regularization terms have two effects. One is that the update $\widetilde{\boldsymbol{f}}$ is not exactly a circulation, so this will account for some change in the routed demand. The other effect is that after augmenting the current flow $\boldsymbol{f}$ with $\widetilde{\boldsymbol{f}}$ we do not obtain a central solution, as shown in Lemma 3.6.6. We proceed to fix this manually by slightly increasing the weights $\boldsymbol{w}$. This follows from applying Lemma 3.4.20 to the residual $\Delta\boldsymbol{h} = -R_p(\widetilde{\boldsymbol{f}})^{p-1}$, which produces a central solution with a new set of weights $\boldsymbol{w}' \geq \boldsymbol{w}$ such that $\|\boldsymbol{w}' - \boldsymbol{w}\|_1 \leq R_p \cdot \sum_{e \in E} \left| \widetilde{f}_e \right|^{p-1}$.

We are now ready to characterize the amount of progress we make in a single iteration of the method previously described.

**Lemma 3.6.7** (Progress lemma)**.** *Given a $\mu$-central instance, i.e. a flow $\boldsymbol{f}$ and balanced weights $\boldsymbol{w}$ such that*

$$\boldsymbol{C}^\top \left( \frac{\boldsymbol{w}^+}{\mathbf{1}-\boldsymbol{f}} - \frac{\boldsymbol{w}^-}{\boldsymbol{f}} \right) = -\frac{\boldsymbol{C}^\top \boldsymbol{c}}{\mu} ,$$

*in the time require to solve (3.64) we can obtain a $\mu/(1+\delta)$-central instance, i.e. a flow $\boldsymbol{f}+\widetilde{\boldsymbol{f}}$ and weights $\boldsymbol{w}' \geq \boldsymbol{w}$, such that*

$$\boldsymbol{C}^\top \left( \frac{\boldsymbol{w}^+}{\mathbf{1}-\boldsymbol{f}-\widetilde{\boldsymbol{f}}} - \frac{\boldsymbol{w}^-}{\boldsymbol{f}+\widetilde{\boldsymbol{f}}} \right) = -(1+\delta)\frac{\boldsymbol{C}^\top \boldsymbol{c}}{\mu} ,$$

*where*

$$\|\boldsymbol{w}' - \boldsymbol{w}\|_1 \leq p \cdot 10^{12} \cdot \delta^4 \|\boldsymbol{w}\|_1^{2+1/p} \cdot \log^2 \|\boldsymbol{w}\|_1 ,$$

*and $\widetilde{\boldsymbol{f}}$ routes a demand $\widetilde{\boldsymbol{d}}$ such that*

$$\|\widetilde{\boldsymbol{d}}\|_1 \leq 3/2 .$$

*Proof.* Using Lemma 3.6.3 and Lemma 3.6.6, after solving (3.64) we obtain an augmenting flow $\widetilde{\boldsymbol{f}}$ routing a demand $\widetilde{\boldsymbol{d}}$ with

$$\|\widetilde{\boldsymbol{d}}\|_1 \leq 3\left(\frac{\|\boldsymbol{w}\|_1 \cdot \mathcal{E}^{\max}(\boldsymbol{h},\boldsymbol{w},\boldsymbol{s})}{R_\star}\right)^{1/2} \leq 3\left(\frac{\|\boldsymbol{w}\|_1 \cdot \frac{1}{2}\delta^2\|\boldsymbol{w}\|_1}{3\delta^2\|\boldsymbol{w}\|_1^2}\right)^{1/2} < 3/2\,,$$

such that

$$\boldsymbol{C}^\top\left((1+\delta)\left(\frac{\boldsymbol{w}^+}{\mathbf{1}-\boldsymbol{f}}-\frac{\boldsymbol{w}^-}{\boldsymbol{f}}\right)-\left(\frac{\boldsymbol{w}^+}{\mathbf{1}-\boldsymbol{f}-\widetilde{\boldsymbol{f}}}-\frac{\boldsymbol{w}^-}{\boldsymbol{f}+\widetilde{\boldsymbol{f}}}\right)\right) = \boldsymbol{C}^\top\cdot R_p\cdot(\widetilde{\boldsymbol{f}})^{p-1}\,.$$

We then increase the weights $\boldsymbol{w}$ to $\boldsymbol{w}'$ to make the right-hand side of this identity equal to $\mathbf{0}$. Per Lemma 3.4.20, this increases the weights to $\boldsymbol{w}'$ such that

$$\|\boldsymbol{w}'-\boldsymbol{w}\|_1 = R_p\cdot\sum_{e\in E}\left|\widetilde{f}_e\right|^{p-1} \leq R_p\left(m^{\frac{1}{p-1}-\frac{1}{p}}\cdot\|\widetilde{\boldsymbol{f}}\|_p\right)^{p-1} \leq R_p\cdot\|\boldsymbol{w}\|_1^{1/p}\cdot\|\widetilde{\boldsymbol{f}}\|_p^{p-1}\,,$$

where we used the fact that $\|\boldsymbol{w}\|_1 \geq m$. From Lemma 3.6.3, we have that for our specific choice of regularization parameters,

$$\|\widetilde{\boldsymbol{f}}\|_p \leq \left(\frac{p\cdot\frac{3}{2}\mathcal{E}^{\max}(\boldsymbol{h},\boldsymbol{w},\boldsymbol{s})}{R_p}\right)^{1/p} \leq \frac{1}{10^6\cdot\delta^2\|\boldsymbol{w}\|_1\log\|\boldsymbol{w}\|_1}\,.$$

Therefore

$$\begin{aligned}\|\boldsymbol{w}'-\boldsymbol{w}\|_1 &\leq \left(p\cdot\left(10^6\cdot\delta^2\|\boldsymbol{w}\|_1\cdot\log\|\boldsymbol{w}\|_1\right)^{p+1}\right)\cdot\|\boldsymbol{w}\|_1^{1/p}\cdot\left(\frac{1}{10^6\cdot\delta^2\|\boldsymbol{w}\|_1\log\|\boldsymbol{w}\|_1}\right)^{p-1}\\ &= p\cdot\left(10^6\cdot\delta^2\|\boldsymbol{w}\|_1\cdot\log\|\boldsymbol{w}\|_1\right)^2\cdot\|\boldsymbol{w}\|_1^{1/p} = p\cdot 10^{12}\cdot\delta^4\|\boldsymbol{w}\|_1^{2+1/p}\cdot\log^2\|\boldsymbol{w}\|_1\,.\end{aligned}$$

□

### 3.6.3 Wrapping Up

We can now give the main statement of this section, which follows from running the interior point method, based on the guarantee provided by Lemma 3.6.7.

**Lemma 3.6.8.** *Suppose we have a $\mu$-central instance with weights $\boldsymbol{w} \geq \mathbf{1}$, where $\|\boldsymbol{w}\|_1 \leq 2m+1$ and $\mu = m^{O(1)}$. Let $\varepsilon = m^{-O(1)}$, and let $\delta = m^{-(1/3+o(1))}$. In time dominated by $\widetilde{O}(\delta^{-1})$ iterations of the procedure described in Lemma 3.6.7 we obtain an instance with duality gap at most $\varepsilon$ with a total demand perturbation of $\widetilde{O}(\delta^{-1})$.*

*Proof.* We perform a sequence of iterations as described in Lemma 3.6.7. These are interspersed with calls to the weight balancing procedure (Lemma 3.4.23), required in order to maintain the invariant needed by Lemma 3.6.5.

We will argue that within $T = \widetilde{O}(\delta^{-1}\log(m\mu\varepsilon^{-1}))$ iterations the barrier weights will always satisfy $\|\boldsymbol{w}\|_1 \leq 3m$. We need to take into account the total increase guaranteed by Lemma 3.6.7 together with the possible weight increases caused by the weight balancing procedure.

First let us bound the total number of balancing operations, since each of these can increase a single weight by a significant amount. First, from Definition 3.4.22 we see that once an edge gets balanced it will never become unbalanced again, as weights are monotonic.

Furthermore, we see that such an operation can only occur when the largest of the two paired weights is at least $\delta\|\boldsymbol{w}\|_1 \geq 2m\delta$, since we maintain $\boldsymbol{w} \geq \mathbf{1}$. Under the invariant that throughout the entire algorithm $\|\boldsymbol{w}\|_1 \leq 3m$, we therefore see that this can only happen at most $(3m)/(2m\delta) = 3/(2\delta)$ times. Invoking Lemma 3.4.23 we therefore get that the total weight change caused by these operations is at most

$$\frac{3}{2\delta} \cdot 96 \cdot \delta^4 \cdot (3m)^2 = 1296 \cdot \delta^3 m^2 \,.$$

In addition, we incur weight increases due to the progress steps; per Lemma 3.6.7, within each of the $T$ iterations, $\|\boldsymbol{w}\|_1$ increases by at most

$$p \cdot 10^{12} \cdot \delta^4 (3m)^{2+1/p} \cdot \log^2(3m) \,.$$

Therefore the total weight increase over $T$ iterations is at most

$$\widetilde{O}\left(\left(p\delta^3 m^{2+1/p}\right)\log(m\mu\varepsilon^{-1})\right)$$

which is $o(m)$ as long as

$$\delta \leq 1/\left(p^{1/3} m^{(1+1/p)/3+o(1)}\right) = 1/m^{1/3+o(1)} \,.$$

Thus this specific choice of $\delta$ insures that the invariant $\|\boldsymbol{w}\|_1 \leq 3m$ was satisfied throughout the entire run of the algorithm.

Finally, we bound the total perturbation in demand suffered by the flow we maintain. Each of the $\widetilde{O}(\delta^{-1})$ iterations perturbs the demand by at most $3/2$ in $\ell_1$ norm, and the weight balancing operations may perturb the demand by an additional $\frac{3}{2\delta}$ overall. Summing up we obtain the desired claim. □

This enables us to obtain a running time of $m^{4/3+o(1)}$ for minimum cost flow in unit-capacity graphs. The proof is identical to that of Theorem 3.4.26, we use scaling to obtain a logarithmic dependence in $W$, and resort to the fixing procedure from Section 3.5 to repair the demand perturbation. The time required to implement each iteration of the interior point method is dominated by the time required by one call to the solver in Theorem 9.1.7, which is $m^{1+o(1)}$ by our choice of parameters.

**Theorem 3.6.9.** *Given a directed graph $G(V, E, \boldsymbol{c})$ with $m$ arcs and $n$ vertices, such that $\|\boldsymbol{c}\|_\infty \leq W$, and a demand vector $\boldsymbol{d} \in \mathbb{Z}^n$, in $m^{4/3+o(1)} \log W$ time we can obtain a flow $\boldsymbol{f}$ which routes $\boldsymbol{d}$ in $G$ while satisfying the capacity constraints $\mathbf{0} \leq \boldsymbol{f} \leq \mathbf{1}$ and minimizing the cost $\sum_{e\in E} c_e f_e$, or certifies that no such flow exists.*

# Chapter 4

# Faster Sparse Minimum Cost Flow by Electrical Flow Localization

## 4.1 Introduction

In the last decade, continuous optimization has proved to be an invaluable tool for designing graph algorithms, often leading to significant improvements over the best known combinatorial algorithms. This has been particularly true in the context of flow problems—arguably, some of the most prominent graph problems [44, 39, 105, 118, 152, 97, 106, 136, 117, 42, 151, 150, 153, 115, 114, 10, 26, 25]. Indeed, these developments have brought a host of remarkable improvements in a variety of regimes, such as when seeking only approximate solutions, or when the underlying graph is dense. However, most of these improvements did not fully address the challenge of seeking *exact* solutions in *sparse* graphs. Fortunately, the improvements for that regime eventually emerged [118, 117, 42, 115, 114, 10]. They still suffered though from an important shortcoming: they all had a polynomial running time dependency in the graph's capacities, and hence—in contrast to the classical combinatorial algorithms—they did not yield efficient algorithms in the presence of arbitrary capacities. Recently, Gao, Liu and Peng [73] have finally changed this state of affairs by providing the first *exact* maximum flow algorithm to break the $\widetilde{O}\left(m^{3/2}\log^{O(1)}U\right)$ barrier for sparse graphs with *general* capacities (bounded by $U$). Their approach, however, crucially relies on a preconditioning technique that is specific to the maximum flow problem and, in particular, having an $s$-$t$ demand—rather than a general one. As a result, the corresponding improvement held only for that particular problem.

In this chapter, we demonstrate how to circumvent these limitations and provide the first algorithm that breaks the $\widetilde{O}\left(m^{3/2}\log^{O(1)}(U+W)\right)$ barrier for the *minimum cost flow* problem in *sparse* graphs with general demands, capacities (bounded by $U$), and costs (bounded by $W$). This algorithm runs in time $\widetilde{O}\left(m^{3/2-1/762}\log(U+W)\right)$.

### 4.1.1 Previous work

In 2013, Mądry [118] presented the first running time improvement to the maximum flow problem since the $\widetilde{O}\left(m\sqrt{n}\log U\right)$ algorithm of [75] in the regime of sparse graphs with small capacities. To this end, he presented an algorithm that runs in time $\widetilde{O}\left(m^{10/7}\mathrm{poly}(U)\right)$, where $U$ is a bound on edge capacities, breaking past the $\widetilde{O}\left(m^{3/2}\right)$ running time barrier that has for decades resisted improvement attempts. The main idea in that work was to use an interior point method with an improved number of iterations guarantee that was delivered via use of an adaptive re-weighting of the central path and careful perturbations of the problem instance. Building on this framework,

a series of subsequent works [117, 115, 114] has brought the runtime of sparse max flow down to $\widetilde{O}\left(m^{4/3}\text{poly}(U)\right)$. (With the most recent of these works crucially relying on nearly-linear time $\ell_p$ flows [103].) In parallel [42, 10], the running time of the more general minimum cost flow problem was reduced to $\widetilde{O}\left(m^{4/3}\text{poly}(U)\log W\right)$, where $W$ is a bound on edge costs.

However, even though these algorithms offer a significant improvement when $U$ is relatively small, the question of whether there exists an algorithm faster than $\widetilde{O}\left(m^{3/2}\log^{O(1)}U\right)$ for sparse graphs with general capacities remained open. In fact, a polynomial dependence on capacities or costs seems inherent in the central path re-weighting technique used in all the aforementioned works. Recently, [73] finally made progress on this question by developing an algorithm for the maximum flow problem that runs in time $\widetilde{O}\left(m^{3/2-1/328}\log U\right)$. The source of improvement here was different from previous works, in the sense that it was not based on decreasing the number of iterations of the interior point method. Instead, it was based on devising a data structure to solve the dynamically changing Laplacian system required by the interior point method in sublinear time per iteration.

The new approach put forth by [73], despite being quite different to the prior ones, still leaned on the preconditioning approach of [117], as well as on other properties that are specific to the maximum flow problem. For this reason, this improvement did not extend to the minimum cost flow problem with general capacities, for which the fastest known runtime was still $\widetilde{O}\left(m\log(U+W)+n^{1.5}\log^2(U+W)\right)$ [25] and $\widetilde{O}(m^{3/2}\log^{O(1)}(U+W))$ [44] in the sparse regime.

### 4.1.2 Our result

In this work, we give an algorithm for the minimum cost flow problem with a running time of $\widetilde{O}\left(m^{3/2-1/762}\log(U+W)\right)$. This is the first improvement for sparse graphs with general capacities over [44], which runs in time $\widetilde{O}\left(m^{3/2}\log^{O(1)}(U+W)\right)$. Specifically, we prove that:

**Theorem 4.1.1.** *Given a graph $G(V,E)$ with edge costs $\boldsymbol{c}\in\mathbb{Z}^m_{[-W,W]}$, a demand $\boldsymbol{d}\in\mathbb{R}^n$, and capacities $\boldsymbol{u}\in\mathbb{Z}^m_{(0,U]}$, there exists an algorithm that with high probability runs in time $\widetilde{O}\left(m^{3/2-1/762}\log(U+W)\right)$ and returns a flow $\boldsymbol{f}\in[\boldsymbol{0},\boldsymbol{u}]$ in $G$ such that $\boldsymbol{f}$ routes the demand $\boldsymbol{d}$ and the cost $\langle\boldsymbol{c},\boldsymbol{f}\rangle$ is minimized.*

### 4.1.3 High level overview of our approach

As we build on the approach presented in [73], we first briefly overview some of the key ideas introduced there that will also be relevant for our discussion. The maximum flow interior point method by [117] works by, repeatedly over $\widetilde{O}\left(\sqrt{m}\right)$ steps, taking an electrical flow step that is a multiple of

$$\widetilde{\boldsymbol{f}}=\boldsymbol{R}^{-1}\boldsymbol{B}\boldsymbol{L}^{+}\boldsymbol{B}^{\top}\boldsymbol{1}_{st}\,,$$

where $\boldsymbol{L}=\boldsymbol{B}^{\top}\boldsymbol{R}^{-1}\boldsymbol{B}$ is a Laplacian matrix and $\boldsymbol{r}$ are resistances that change per step. However, $\widetilde{\boldsymbol{f}}$ has $m$ entries and takes $\widetilde{O}\left(m\right)$ to compute, which gives the standard $\widetilde{O}\left(m^{3/2}\right)$ bound. To go beyond this, [73] show that it suffices to compute $\widetilde{\boldsymbol{f}}$ for only a *sublinear* number of high-congestion entries of $\widetilde{\boldsymbol{f}}$, where congestion is defined as $\boldsymbol{\rho}=\sqrt{\boldsymbol{r}}\widetilde{\boldsymbol{f}}$. By known linear sketching results, these edges can be detected by computing the inner product $\langle\boldsymbol{q},\boldsymbol{\rho}\rangle$ for a small number of randomly chosen vectors $\boldsymbol{q}\in\mathbb{R}^m$. Crucially, given a vertex subset $C\subseteq V$ of sublinear size that contains $s$ and $t$, this inner product can be equivalently written as the following sublinear-sized inner product

$$\langle\boldsymbol{q},\boldsymbol{\rho}\rangle=\left\langle\boldsymbol{\pi}^{C}\left(\boldsymbol{B}^{\top}\frac{\boldsymbol{q}}{\sqrt{\boldsymbol{r}}}\right),SC^{+}\boldsymbol{d}\right\rangle\,,\tag{4.1}$$

where $SC := SC(G, C)$ is the *Schur complement* of $G$ onto $C$, $\boldsymbol{d}$ is equal to $\boldsymbol{B}^\top \mathbf{1}_{st}$, and $\boldsymbol{\pi}^C(\cdot)$ is a *demand projection* onto $C$. Therefore, the problem is reduced to maintaining two quantities: $\boldsymbol{\pi}^C\left(\boldsymbol{B}^\top \frac{\boldsymbol{q}}{\sqrt{\boldsymbol{r}}}\right)$ and $(SC(G, C))^+ \boldsymbol{d}$ in sublinear time per operation. The latter is computed by using the dynamic Schur complement data structure of [51], and the former can be maintained by a careful use of random walks.

We now describe our approach. Instead of using the interior point method formulation of [117] which only applies to the maximum flow problem, we use the one by [10] for the, more general, minimum cost flow problem.

There are now several obstacles to making this approach work by maintaining the quantity $\langle \boldsymbol{q}, \boldsymbol{\rho} \rangle$:

**Preconditioning** A significant difference between [117] and [10] is that while the former is able to guarantee that the magnitude of the electrical potentials computed in each step is inversely proportional to the duality gap, meaning that a large duality gap implies potential embeddings of low stretch, no such preconditioning method is known for minimum cost flow. In fact, [10] used demand perturbations to show that a *weaker* bound on the potentials can be achieved, which was still sufficient for their purposes. Unfortunately, this bound is not strong enough to be used in the analysis of [73].

In order to alleviate this issue, we completely remove preconditioning from the picture by only requiring a bound on the *energy* of the electrical potentials (instead of their magnitude). In particular, given an approximate demand projection $\widetilde{\pi}^C\left(\boldsymbol{B}^\top \frac{\boldsymbol{q}}{\sqrt{\boldsymbol{r}}}\right)$, identity (4.1) is used to detect congested edges. In [73], there is a uniform upper bound on the entries of the potential embedding $\boldsymbol{\phi} = SC^+ \boldsymbol{d}$ because of preconditioning, thus the error in (4.1) can be bounded by

$$\left\| \widetilde{\pi}^C\left(\boldsymbol{B}^\top \frac{\boldsymbol{q}}{\sqrt{\boldsymbol{r}}}\right) - \boldsymbol{\pi}^C\left(\boldsymbol{B}^\top \frac{\boldsymbol{q}}{\sqrt{\boldsymbol{r}}}\right) \right\|_1 \|\boldsymbol{\phi}\|_\infty .$$

As we do not have a good bound on $\|\boldsymbol{\phi}\|_\infty$, we instead use an alternative upper bound on the error:

$$\sqrt{\mathcal{E}_{\boldsymbol{r}}\left(\widetilde{\pi}^C\left(\boldsymbol{B}^\top \frac{\boldsymbol{q}}{\sqrt{\boldsymbol{r}}}\right) - \boldsymbol{\pi}^C\left(\boldsymbol{B}^\top \frac{\boldsymbol{q}}{\sqrt{\boldsymbol{r}}}\right)\right)} \sqrt{E_{\boldsymbol{r}}(\boldsymbol{\phi})},$$

where $\mathcal{E}_{\boldsymbol{r}}(\cdot)$ gives the energy to route a demand with resistances $\boldsymbol{r}$, and $E_{\boldsymbol{r}}(\cdot)$ gives the energy of a potential embedding with resistances $\boldsymbol{r}$. As the standard interior point method step satisfies $E_{\boldsymbol{r}}(\boldsymbol{\phi}) \leq 1$, all our efforts focus on ensuring that

$$\sqrt{\mathcal{E}_{\boldsymbol{r}}\left(\widetilde{\pi}^C\left(\boldsymbol{B}^\top \frac{\boldsymbol{q}}{\sqrt{\boldsymbol{r}}}\right) - \boldsymbol{\pi}^C\left(\boldsymbol{B}^\top \frac{\boldsymbol{q}}{\sqrt{\boldsymbol{r}}}\right)\right)} \leq \varepsilon \tag{4.2}$$

for some error parameter $\varepsilon$. One issue is the fact that the energy depends on the current resistances, therefore even if at some point the error of the demand projection is low, after a few iterations it might increase because of resistance changes. We deal with this issue by taking the stability of resistances along the central path into account. This allows us to upper bound how much this error increases after a number of iterations. The resistance stability lemma is a generalization of the one used in [73].

Unfortunately, even though (4.2) seems like the right type of guarantee, it is unclear how to ensure that it is always true. Specifically, it involves efficiently computing the hitting probabilities from some vertex $v$ to $C$ in an appropriate norm, which ends up being non-trivial. Instead, we show

that the following *weaker* error bound can be ensured with high probability:

$$\left|\left\langle \widetilde{\pi}^{C}\left(\boldsymbol{B}^{\top}\frac{\boldsymbol{q}}{\sqrt{\boldsymbol{r}}}\right)-\boldsymbol{\pi}^{C}\left(\boldsymbol{B}^{\top}\frac{\boldsymbol{q}}{\sqrt{\boldsymbol{r}}}\right),\boldsymbol{\phi}\right\rangle\right|\leq\varepsilon\,, \tag{4.3}$$

where $\boldsymbol{\phi}$ is a *fixed* potential vector with $E_{\boldsymbol{r}}(\boldsymbol{\phi})\leq 1$. Interestingly, this guarantee is still sufficient for our purposes.

**Costs and general demand** There is a fundamental obstacle to using the approach of [73] once edge costs are introduced. In particular, for the maximum flow problem, the demand pushed by the electrical flow in each iteration is an $s$-$t$ demand, so—up to scaling—it is always constant. In minimum cost flow on the other hand, the augmenting flow is a multiple of $\boldsymbol{c}-\boldsymbol{R}^{-1}\boldsymbol{B}\boldsymbol{L}^{+}\boldsymbol{B}^{\top}\boldsymbol{c}$. Here it is not possible to locate a sublinear number of congested edges just by looking at the electrical flow term $\boldsymbol{R}^{-1}\boldsymbol{B}\boldsymbol{L}^{+}\boldsymbol{B}^{\top}\boldsymbol{c}$, as there might be significant cancellations with $\boldsymbol{c}$. We instead use the following equivalent form: $\frac{\frac{1}{s^{+}}-\frac{1}{s^{-}}}{\boldsymbol{r}}-\boldsymbol{R}^{-1}\boldsymbol{B}\boldsymbol{L}^{+}\boldsymbol{B}^{\top}\frac{\frac{1}{s^{+}}-\frac{1}{s^{-}}}{\boldsymbol{r}}$, which allows us to ignore the first term because it is small and concentrate on the electrical flow term. One issue that arises is the fact that the demand vector $\boldsymbol{B}^{\top}\frac{\frac{1}{s^{+}}-\frac{1}{s^{-}}}{\boldsymbol{r}}$ now depends on slacks, and as a result changes throughout the interior point method. This issue can be handled relatively easily.

A more significant issue concerns the vertex sparsifier. In fact, the vertex sparsifier framework around which [73] is based only accepts demands that are supported on the vertex set $C$ of the sparsifier. As $|C|$ is sublinear in $n$, this only captures demands with sublinear support, one such example being max flow with support 2. However, our demand vector $\boldsymbol{B}^{\top}\frac{\frac{1}{s^{+}}-\frac{1}{s^{-}}}{\boldsymbol{r}}$ in general will be supported on $n$ vertices. Even though it might seem impossible to get around this issue, we show that the special structure of $C$ allows us to push the demand to a small number of vertices. More specifically, we show that if one projects all of the demand onto $C$, the flow induced by this new demand will not differ much from the one with the original demand. Concretely, given a Laplacian system $\boldsymbol{L}\boldsymbol{\phi}=\boldsymbol{d}$, we decompose it into two systems $\boldsymbol{L}\boldsymbol{\phi}^{(1)}=\boldsymbol{\pi}^{C}(\boldsymbol{d})$ and $\boldsymbol{L}\boldsymbol{\phi}^{(2)}=\boldsymbol{d}-\boldsymbol{\pi}^{C}(\boldsymbol{d})$, where $\boldsymbol{\pi}^{C}(\boldsymbol{d})$ is the projection of $\boldsymbol{d}$ onto $C$. Intuitively, the latter system computes the electrical flow to push all demands to $C$, and the former to serve this $C$-supported demand. We show that, as long as $C$ is a *congestion reduction subset* (as it is also the case in [73]), $\boldsymbol{\phi}^{(2)}$ has negligible contribution in the electrical flow, thus it can be ignored. More specifically, in Section 4.4.1 we prove the following lemma:

**Lemma 4.4.3.** *Consider a graph $G(V,E)$ with resistances $\boldsymbol{r}$ and Laplacian $\boldsymbol{L}$, a $\beta$-congestion reduction subset $C$, and a demand $\boldsymbol{d}=\delta\boldsymbol{B}^{\top}\frac{\boldsymbol{q}}{\sqrt{\boldsymbol{r}}}$ for some $\delta>0$ and $\boldsymbol{q}\in[-1,1]^{m}$. Then, the potential embedding defined as*

$$\boldsymbol{\phi}=\boldsymbol{L}^{+}\left(\boldsymbol{d}-\boldsymbol{\pi}^{C}\left(\boldsymbol{d}\right)\right)$$

*has congestion $\delta\cdot\widetilde{O}\left(1/\beta^{2}\right)$, i.e. $\left\|\frac{\boldsymbol{B}\boldsymbol{\phi}}{\sqrt{\boldsymbol{r}}}\right\|_{\infty}\leq\delta\cdot\widetilde{O}\left(1/\beta^{2}\right)$.*

Now, for computing $\boldsymbol{\phi}^{(1)}$, we need to get an approximate estimate of $\boldsymbol{\pi}^{C}(\boldsymbol{d})$. Even though the most natural approach would be to try to maintain $\boldsymbol{\pi}^{C}(\boldsymbol{d})$ under vertex insertions to $C$, this approach has issues related to the fact that our error guarantee is based on a *fixed* potential vector. In particular, if we used an estimate of $\boldsymbol{\pi}^{C}(\boldsymbol{d})$, then the potential vector in (4.3) would depend on the randomness of this estimate, and as a result the high probability guarantee would not work.

Instead, we show that it is not even neccessary to maintain $\boldsymbol{\pi}^{C}(\boldsymbol{d})$ very accurately. In fact, it suffices to *exactly* compute it only every few iterations of the algorithm, and use this estimate for the

calculation. What allows us to do this is the following lemma, which bounds the change of $\boldsymbol{\pi}^C(\boldsymbol{d})$ measured in energy, after a sequence of vertex insertions and resistance changes.

**Lemma 4.4.12.** *Consider a graph $G(V,E)$ with resistances $\boldsymbol{r}^0$, $\boldsymbol{q}^0 \in [-1,1]^m$, a $\beta$-congestion reduction subset $C^0$, and a fixed sequence of updates, where the $i$-th update $i \in \{0, T-1\}$ is of the following form:*

- AddTerminal*$(v^i)$: Set $C^{i+1} = C^i \cup \{v^i\}$ for some $v^i \in V \backslash C^i$, $q_e^{i+1} = q_e^i, r_e^{i+1} = r_e^i$*
- Update*$(e^i, \boldsymbol{q}, \boldsymbol{r})$: Set $C^{i+1} = C^i$, $q_e^{i+1} = q_e$ $r_e^{i+1} = r_e$, where $e^i \in E(C^i)$*

*Then, with high probability,*

$$\sqrt{\mathcal{E}_{\boldsymbol{r}^T}\left(\boldsymbol{\pi}^{C^0,\boldsymbol{r}^0}\left(\boldsymbol{B}^\top \frac{\boldsymbol{q}_S^0}{\sqrt{\boldsymbol{r}^0}}\right) - \boldsymbol{\pi}^{C^T,\boldsymbol{r}^T}\left(\boldsymbol{B}^\top \frac{\boldsymbol{q}_S^T}{\sqrt{\boldsymbol{r}^T}}\right)\right)} \leq \widetilde{O}\left(\max_{i\in\{0,\ldots,T-1\}} \left\|\frac{\boldsymbol{r}^T}{\boldsymbol{r}^i}\right\|_\infty^{1/2} \beta^{-2}\right) \cdot T\,.$$

If we call this demand projection estimate $\boldsymbol{\pi}_{old}$, the quantity that we would like to maintain (4.1) now becomes

$$\langle \boldsymbol{q}, \boldsymbol{\rho}\rangle \approx \left\langle \boldsymbol{\pi}^C\left(\boldsymbol{B}^\top \frac{\boldsymbol{q}}{\sqrt{\boldsymbol{r}}}\right), SC^+\boldsymbol{\pi}_{old}\right\rangle\,.$$

Therefore all that's left is to efficiently maintain approximations to demand projections of the form $\boldsymbol{\pi}^C\left(\boldsymbol{B}^\top \frac{\boldsymbol{q}}{\sqrt{\boldsymbol{r}}}\right)$.

**Bounding demand projections.** An important component for showing that demand projections can be updated efficiently is bounding the magnitude of an entry $\pi_v^{C\cup\{v\}}(\boldsymbol{B}^\top \frac{\mathbf{1}_e}{\sqrt{r_e}})$ of the projection, for some fixed edge $e = (u,w)$. This is apparent in the following identity which shows how a demand projection changes after inserting a vertex:

$$\boldsymbol{\pi}^{C\cup\{v\}}(\boldsymbol{d}) = \boldsymbol{\pi}^C(\boldsymbol{d}) + \pi_v^{C\cup\{v\}}(\boldsymbol{d}) \cdot \left(\mathbf{1}_v - \boldsymbol{\pi}^C(\mathbf{1}_v)\right)\,. \tag{4.4}$$

In [73] this projection entry is upper bounded by $(p_v^{C\cup\{v\}}(u) + p_v^{C\cup\{v\}}(w)) \cdot \frac{1}{\sqrt{r_e}}$, where $p_v^{C\cup\{v\}}(u)$ is the probability that a random walk starting at $u$ hits $v$ before $C$. This bound can be very bad as $r_e$ can be arbitrarily small, although in the particular case of max flow it is possible to show that such low-resistance edges cannot get congested and thus are not of interest.

In order to overcome this issue, we provide a different bound, which in contrast works best when $r_e$ is small.

**Lemma 4.4.6.** *Consider a graph $G(V,E)$ with resistances $\boldsymbol{r}$ and a subset of vertices $C \subseteq V$. For any vertex $v \in V\backslash C$ we have that*

$$\left|\pi_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top \frac{\mathbf{1}_e}{\sqrt{\boldsymbol{r}}}\right)\right| \leq (p_v^{C\cup\{v\}}(u) + p_v^{C\cup\{v\}}(w)) \cdot \frac{\sqrt{r_e}}{R_{eff}(v,e)}\,.$$

Here $R_{eff}(v,e)$ is the effective resistance between $v$ and $e$. In fact, together with the other upper bound mentioned above, this implies that

$$\left|\pi_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top \frac{\mathbf{1}_e}{\sqrt{\boldsymbol{r}}}\right)\right| \leq (p_v^{C\cup\{v\}}(u) + p_v^{C\cup\{v\}}(w)) \cdot \frac{1}{\sqrt{R_{eff}(v,e)}}\,,$$

which no longer depends on the value of the resistance $r_e$.

As we will see, it suffices to approximate $\pi_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top \frac{\mathbf{1}_e}{\sqrt{\boldsymbol{r}}}\right)$ up to additive accuracy roughly $\widehat{\varepsilon}\cdot(p_v^{C\cup\{v\}}(u)+p_v^{C\cup\{v\}}(w))/\sqrt{R_{eff}(C,v)}$ for some error parameter $\widehat{\varepsilon}>0$. Thus, Lemma 4.4.6 immediately implies that for any edge $e$ such that $R_{eff}(v,e)\boldsymbol{g}R_{eff}(C,v)$, this term is small enough to begin with, and thus can be ignored.

**Important edges.** In order to ensure that the demand projection can be updated efficiently, we focus only on the demand coming from a special set of edges, which we call *important*. These are the edges that are close (in effective resistance metric) to $C$ relative to their own resistance $r_e$. In fact, the farther an edge is from $C$ in this sense, the smaller its worst-case congestion, and so non-important edges do not influence the set of congested edges that we are looking for. At a high level, this is because parts of the graph that are very far in the potential embedding have minimal interactions with each other.

**Definition 4.1.2** (Important edges)**.** *An edge $e\in E$ is called* $\varepsilon$-important *(or just* important*) if* $R_{eff}(C,e)\le r_e/\varepsilon^2$.

Based on the above discussion, we seek to find congested edges *only* among important edges.

**Lemma 4.4.9** (Localization lemma)**.** *Let $\boldsymbol{\phi}^*$ be any solution of*

$$\boldsymbol{L}\boldsymbol{\phi}^*=\delta\cdot\boldsymbol{\pi}^C\left(\boldsymbol{B}^\top\frac{\boldsymbol{p}}{\sqrt{\boldsymbol{r}}}\right),$$

*where $\boldsymbol{r}$ are any resistances, $\boldsymbol{p}\in[-1,1]^m$, and $C\subseteq V$. Then, for any $e\in E$ that is not $\varepsilon$-important we have $\left|\frac{\boldsymbol{B}\boldsymbol{\phi}^*}{\sqrt{\boldsymbol{r}}}\right|_e\le 12\varepsilon$.*

One issue is that the set of important edges changes whenever $C$ changes. However, we show that, because of the stability of resistances along the central path, the set of important edges only needs to be updated once every few iterations.

## 4.2 Preliminaries

Given $x,y\in\mathbb{R}$ and $\alpha\in\mathbb{R}_{\ge 1}$, we say that $x$ and $y$ $\alpha$-approximate each other and write $x\approx_\alpha y$ if $\alpha^{-1}\le x/y\le\alpha$.

**Definition 4.2.1** (Schur complement)**.** *Given a graph $G(V,E)$ with Laplacian $\boldsymbol{L}\in\mathbb{R}^{n\times n}$ and a vertex subset $C\subseteq V$ as well as $F=V\backslash C$, $SC(G,C):=\boldsymbol{L}_{CC}-\boldsymbol{L}_{CF}\boldsymbol{L}_{FF}^{-1}\boldsymbol{L}_{FC}$ (or just SC) is called the* Schur complement of $G$ onto $C$.

**Fact 4.2.2** (Cholesky factorization)**.** *Given a matrix $\boldsymbol{L}\in\mathbb{R}^{n\times n}$, a subset $C\subseteq[n]$, and $F=[n]\backslash C$, we have*

$$\boldsymbol{L}^+=\begin{pmatrix}\boldsymbol{I} & -\boldsymbol{L}_{FF}^{-1}\boldsymbol{L}_{FC}\\ \mathbf{0} & \boldsymbol{I}\end{pmatrix}\begin{pmatrix}\boldsymbol{L}_{FF}^{-1} & \mathbf{0}\\ \mathbf{0} & SC(\boldsymbol{L},C)^+\end{pmatrix}\begin{pmatrix}\boldsymbol{I} & \mathbf{0}\\ -\boldsymbol{L}_{CF}\boldsymbol{L}_{FF}^{-1} & \boldsymbol{I}\end{pmatrix}.$$

### 4.2.1 Random walks

**Definition 4.2.3** (Hitting probabilities)**.** *Consider a graph $G(V,E)$ with resistances $\boldsymbol{r}$. For any $u,v\in V$, $C\subseteq V$, we denote by $p_v^{C,\boldsymbol{r}}(u)$ the probability that for random walk that starts from $u$*

*and uses edges with probability proportional to $\frac{1}{\boldsymbol{r}}$, the first vertex of $C$ to be visited is $v$. When not ambiguous, we will use the notation $p_v^C(u)$.*

**Definition 4.2.4** (Demand projection)**.** *Consider a graph $G(V,E)$ and a demand vector $\boldsymbol{d}$. For any $v \in V$, $C \subseteq V$, we define $\pi_v^{C,\boldsymbol{r}}(\boldsymbol{d}) = \sum_{u \in V} d_u p_v^{C,\boldsymbol{r}}(u)$ and call the resulting vector $\boldsymbol{\pi}^{C,\boldsymbol{r}}(\boldsymbol{d}) \in \mathbb{R}^n$ the* demand projection of $\boldsymbol{d}$ onto $C$*. When not ambiguous, we will use the notation $\boldsymbol{\pi}^C(\boldsymbol{d})$.*

For convenience, when we write $\boldsymbol{\pi}^C(\boldsymbol{d})$ we might also refer to the restriction of this vector to $C$. This will be clear from the context, and, as $\pi_v^C(\boldsymbol{d}) = 0$ for any $v \notin C$, no ambiguity is introduced.

**Fact 4.2.5** ([73])**.** *Given a graph $G(V,E)$ with Laplacian $\boldsymbol{L}$, a vertex subset $C \subseteq V$, and $\boldsymbol{d} \in \mathbb{R}^n$, we have*

$$\boldsymbol{\pi}^C(\boldsymbol{d}) = \boldsymbol{d}_C - \boldsymbol{L}_{CF}\boldsymbol{L}_{FF}^{-1}\boldsymbol{d}_F\,.$$

*Additionally,*

$$\left[\boldsymbol{L}^+\boldsymbol{d}\right]_C = SC^+\boldsymbol{\pi}^C(\boldsymbol{d})\,,$$

*where $SC$ is the Schur complement of $G$ onto $C$.*

An important property of the demand projection is that the energy required to route it is upper bounded by the energy required to route the original demand. The proof can be found in Section 9.2.2.

**Lemma 4.2.6.** *Let $\boldsymbol{d}$ be a demand vector, let $\boldsymbol{r}$ be resistances, and let $C \subseteq V$ be a subset of vertices. Then*

$$\mathcal{E}_{\boldsymbol{r}}\left(\boldsymbol{\pi}^C(\boldsymbol{d})\right) \leq \mathcal{E}_{\boldsymbol{r}}\left(\boldsymbol{d}\right)\,.$$

The following lemma relates the effective resistance between a vertex and a vertex set, to the energy to route a particular demand, based on a demand projection.

**Lemma 4.2.7** (Effective resistance and hitting probabilities)**.** *Given a graph $G(V,E)$ with resistances $\boldsymbol{r}$, any vertex set $A \subseteq V$ and vertex $u \in V \backslash A$, we have $R_{eff}(u,A) = \mathcal{E}_{\boldsymbol{r}}(\mathbf{1}_u - \boldsymbol{\pi}^A(\mathbf{1}_u))$.*

*Proof.* Let $\boldsymbol{L}$ be the Laplacian of $G$ with resistances $\boldsymbol{r}$ and $F = V \backslash A$. We first prove that

$$\mathcal{E}_{\boldsymbol{r}}(\mathbf{1}_u - \boldsymbol{\pi}^A(\mathbf{1}_u)) = \mathbf{1}_u^\top \boldsymbol{L}_{FF}^{-1}\mathbf{1}_u\,.$$

This is because

$$\begin{aligned}
&\mathcal{E}_{\boldsymbol{r}}(\mathbf{1}_u - \boldsymbol{\pi}^A(\mathbf{1}_u)) \\
&= \langle \mathbf{1}_u - \boldsymbol{\pi}^A(\mathbf{1}_u), \boldsymbol{L}^+(\mathbf{1}_u - \boldsymbol{\pi}^A(\mathbf{1}_u))\rangle \\
&= \left\langle \mathbf{1}_u - \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ -\boldsymbol{L}_{AF}\boldsymbol{L}_{FF}^{-1} & \boldsymbol{I} \end{pmatrix}\mathbf{1}_u, \boldsymbol{L}^+\left(\mathbf{1}_u - \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ -\boldsymbol{L}_{AF}\boldsymbol{L}_{FF}^{-1} & \boldsymbol{I} \end{pmatrix}\mathbf{1}_u\right)\right\rangle \\
&= \left\langle \begin{pmatrix} \boldsymbol{I} & \mathbf{0} \\ -\boldsymbol{L}_{AF}\boldsymbol{L}_{FF}^{-1} & \boldsymbol{I} \end{pmatrix}\left(\mathbf{1}_u + \begin{pmatrix} \mathbf{0} \\ \boldsymbol{L}_{AF}\boldsymbol{L}_{FF}^{-1}\mathbf{1}_u \end{pmatrix}\right), \right. \\
&\qquad \left. \begin{pmatrix} \boldsymbol{L}_{FF}^{-1} & \mathbf{0} \\ \mathbf{0} & SC(\boldsymbol{L},A)^+ \end{pmatrix}\begin{pmatrix} \boldsymbol{I} & \mathbf{0} \\ -\boldsymbol{L}_{AF}\boldsymbol{L}_{FF}^{-1} & \boldsymbol{I} \end{pmatrix}\left(\mathbf{1}_u + \begin{pmatrix} \mathbf{0} \\ \boldsymbol{L}_{AF}\boldsymbol{L}_{FF}^{-1}\mathbf{1}_u \end{pmatrix}\right)\right\rangle \\
&= \left\langle \mathbf{1}_u, \begin{pmatrix} \boldsymbol{L}_{FF}^{-1} & \mathbf{0} \\ \mathbf{0} & SC(\boldsymbol{L},A)^+ \end{pmatrix}\mathbf{1}_u \right\rangle \\
&= \langle \mathbf{1}_u, \boldsymbol{L}_{FF}^{-1}\mathbf{1}_u\rangle\,.
\end{aligned}$$

On the other hand, note that $R_{eff}(u, A) = \widehat{R}_{eff}(u, \widehat{a})$, where $\widehat{R}$ are the effective resistances in a graph $\widehat{G}$ that results after contracting $A$ to a new vertex $\widehat{a}$. It is easy to see that the Laplacian of this new graph is

$$\widehat{\boldsymbol{L}} = \begin{pmatrix} \boldsymbol{L}_{FF} & \boldsymbol{L}_{FA}\mathbf{1} \\ \mathbf{1}^\top \boldsymbol{L}_{AF} & \mathbf{1}^\top \boldsymbol{L}_{FA}\mathbf{1} \end{pmatrix}.$$

We look at the system $\widehat{\boldsymbol{L}} \begin{pmatrix} x \\ a \end{pmatrix} = \mathbf{1}_u - \mathbf{1}_{\widehat{a}}$, where $a$ is a scalar. The solution is given by

$$\boldsymbol{x} = \boldsymbol{L}_{FF}^{-1} \left( \mathbf{1}_u - a \cdot \boldsymbol{L}_{FA}\mathbf{1} \right).$$

However, as $\mathbf{1} \in \ker(\widehat{\boldsymbol{L}})$ by the fact that it is a Laplacian, we can assume that $a = 0$ by shifting. Therefore $\boldsymbol{x} = \boldsymbol{L}_{FF}^{-1}\mathbf{1}_u$, and so we can conclude that

$$\begin{aligned} R_{eff}(u, A) &= \langle \mathbf{1}_u - \mathbf{1}_{\widehat{a}}, \widehat{\boldsymbol{L}}^+ (\mathbf{1}_u - \mathbf{1}_{\widehat{a}}) \rangle \\ &= \langle \mathbf{1}_u, \boldsymbol{L}_{FF}^{-1}\mathbf{1}_u \rangle . \end{aligned}$$

So we have proved that $R_{eff}(u, A) = \mathcal{E}_{\boldsymbol{r}}(\mathbf{1}_u - \boldsymbol{\pi}^A(\mathbf{1}_u))$ and we are done. □

Finally, the following lemma relates the effective resistance between a vertex and a vertex set, to the effective resistance between vertices.

**Lemma 4.2.8.** *Given a graph $G(V, E)$ with resistances $\boldsymbol{r}$, any vertex set $A \subseteq V$ and vertex $u \in V \backslash A$, we have*

$$\frac{1}{|A|} \cdot \min_{v \in A} R_{eff}(u, v) \leq R_{eff}(u, A) \leq \min_{v \in A} R_{eff}(u, v).$$

*Proof.* Let $\boldsymbol{L}$ be the Laplacian of $G$ with resistances $\boldsymbol{r}$, and note that $R_{eff}(u, v) = \left\| \boldsymbol{L}^{+/2}(\mathbf{1}_u - \mathbf{1}_v) \right\|_2^2$ and, by Lemma 4.2.7, $R_{eff}(u, A) = \left\| \boldsymbol{L}^{+/2}(\mathbf{1}_u - \boldsymbol{\pi}^A(\mathbf{1}_u)) \right\|_2^2$. Expanding the latter, we have

$$\begin{aligned} R_{eff}(u, A) &= \left\| \sum_{v \in A} \pi_v^A(\mathbf{1}_u) \cdot \boldsymbol{L}^{+/2}(\mathbf{1}_u - \mathbf{1}_v) \right\|_2^2 \\ &= \sum_{v \in A} \left( \pi_v^A(\mathbf{1}_u) \right)^2 \left\| \boldsymbol{L}^{+/2}(\mathbf{1}_u - \mathbf{1}_v) \right\|_2^2 + \sum_{v \in A} \sum_{\substack{v' \in A \\ v' \neq v}} \pi_v^A(\mathbf{1}_u) \pi_{v'}^A(\mathbf{1}_u) \langle \mathbf{1}_u - \mathbf{1}_{v'}, \boldsymbol{L}^+(\mathbf{1}_u - \mathbf{1}_v) \rangle . \end{aligned}$$

Now, note that $\pi_v^A(\mathbf{1}_u), \pi_{v'}^A(\mathbf{1}_u) \geq 0$. Additionally, let $\boldsymbol{\phi} = \boldsymbol{L}^+(\mathbf{1}_u - \mathbf{1}_v)$ be the potential embedding that induces a 1-unit electrical flow from $v$ to $u$. As the potential embedding stretches between $\phi_v$

and $\phi_u$, we have that $\phi_{v'} \leq \phi_u$, so $\langle \mathbf{1}_u - \mathbf{1}_{v'}, \boldsymbol{L}^+(\mathbf{1}_u - \mathbf{1}_v)\rangle = \phi_u - \phi_{v'} \geq 0$. Therefore,

$$\begin{aligned} R_{eff}(u, A) &\geq \sum_{v \in A} \left(\pi_v^A(\mathbf{1}_u)\right)^2 \left\| \boldsymbol{L}^{+/2}(\mathbf{1}_u - \mathbf{1}_v) \right\|_2^2 \\ &\geq \frac{1}{|A|} \sum_{v \in A} \pi_v^A(\mathbf{1}_u) \cdot R_{eff}(u, v) \\ &\geq \frac{1}{|A|} \min_{v \in A} R_{eff}(u, v)\,, \end{aligned}$$

where we used the Cauchy-Schwarz inequality and the fact that $\sum_{v \in A} \pi_v^A(\mathbf{1}_u) = 1$. □

## 4.3 Interior Point Method with Dynamic Data Structures

The goal of this section is to show that, given a data structure for approximating electrical flows in sublinear time, we can execute the min cost flow interior point method with total runtime faster than $\widetilde{O}\left(m^{3/2}\right)$.

### 4.3.1 LP formulation and background

We present the interior point method setup that we will use, which is from [10]. Our goal is to solve the following minimum cost flow linear program:

$$\begin{aligned} &\min \langle \boldsymbol{c}, \boldsymbol{C}\boldsymbol{x} \rangle \\ &\mathbf{0} \leq \boldsymbol{f}^0 + \boldsymbol{C}\boldsymbol{x} \leq \boldsymbol{u}\,, \end{aligned}$$

where $\boldsymbol{f}^0$ is a flow with $\boldsymbol{B}^\top \boldsymbol{f}^0 = \boldsymbol{d}$ and $\boldsymbol{C}$ is an $m \times (m - n + 1)$ matrix whose image is the set of circulations in $G$.

In order to use an interior point method, the following log barrier objective is defined:

$$\min_{\boldsymbol{x}} F_\mu(\boldsymbol{x}) = \left\langle \frac{\boldsymbol{c}}{\mu}, \boldsymbol{C}\boldsymbol{x} \right\rangle - \sum_{e \in E} \left( \log\left(\boldsymbol{f}^0 + \boldsymbol{C}\boldsymbol{x}\right)_e + \log\left(\boldsymbol{u} - (\boldsymbol{f}^0 + \boldsymbol{C}\boldsymbol{x})\right)_e \right)\,. \tag{4.5}$$

For any parameter $\mu > 0$, the optimality condition of (4.5) is called the *centrality condition* and is given by

$$\boldsymbol{C}^\top \left( \frac{\boldsymbol{c}}{\mu} + \frac{1}{\boldsymbol{s}^+} - \frac{1}{\boldsymbol{s}^-} \right) = \mathbf{0}\,, \tag{4.6}$$

where $\boldsymbol{f} = \boldsymbol{f}^0 + \boldsymbol{C}\boldsymbol{x}$, and $\boldsymbol{s}^+ = \boldsymbol{u} - \boldsymbol{f}$, $\boldsymbol{s}^- = \boldsymbol{f}$ are called the *positive* and *negative* slacks of $\boldsymbol{f}$ respectively.

This leads us to the following definitions.

**Definition 4.3.1** ($\mu$-central flow)**.** *Given a minimum cost flow instance with costs $\boldsymbol{c}$, demands $\boldsymbol{d}$ and capacities $\boldsymbol{u}$, as well as a parameter $\mu > 0$, we will say that a flow $\boldsymbol{f}$ (and its corresponding slacks $\boldsymbol{s}$ and resistances $\boldsymbol{r}$) is $\mu$-central if $\boldsymbol{B}^\top \boldsymbol{f} = \boldsymbol{d}$, $\boldsymbol{s} > \mathbf{0}$, and it satisfies the centrality condition*

*(4.6), i.e.*

$$\boldsymbol{C}^\top \left( \frac{\boldsymbol{c}}{\mu} + \frac{1}{\boldsymbol{s}^+} - \frac{1}{\boldsymbol{s}^-} \right) = \mathbf{0} \,.$$

*Additionally, we will denote such flow by $\boldsymbol{f}(\mu)$ (and its corresponding slacks and resistances by $\boldsymbol{s}(\mu)$ and $\boldsymbol{r}(\mu)$, respectively).*

**Definition 4.3.2** ($(\mu,\alpha)$-central flow)**.** *Given parameters $\mu > 0$ and $\alpha \geq 1$, we will say that a flow $\boldsymbol{f}$ with resistances $\boldsymbol{r} > \mathbf{0}$ is $(\mu,\alpha)$-central if $\boldsymbol{r} \approx_\alpha \boldsymbol{r}(\mu)$. We will also call its corresponding slacks $\boldsymbol{s}$ and resistances $\boldsymbol{r}$ $(\mu,\alpha)$-central.*

Given a $\mu$-central flow $\boldsymbol{f}$ and some step size $\delta > 0$, the standard (Newton) step to obtain an approximately $\mu/(1+\delta)$-central flow $\boldsymbol{f}' = \boldsymbol{f} + \widetilde{\boldsymbol{f}}$ is given by

$$\begin{aligned}
\widetilde{\boldsymbol{f}} &= -\frac{\delta}{\mu}\frac{\boldsymbol{c}}{\boldsymbol{r}} + \frac{\delta}{\mu}\boldsymbol{R}^{-1}\boldsymbol{B}\boldsymbol{L}^+\boldsymbol{B}^\top\frac{\boldsymbol{c}}{\boldsymbol{r}} \\
&= \delta\frac{\frac{1}{\boldsymbol{s}^+} - \frac{1}{\boldsymbol{s}^-}}{\boldsymbol{r}} - \delta\boldsymbol{R}^{-1}\boldsymbol{B}\boldsymbol{L}^+\boldsymbol{B}^\top\frac{\frac{1}{\boldsymbol{s}^+} - \frac{1}{\boldsymbol{s}^-}}{\boldsymbol{r}} \\
&= \delta \cdot g(\boldsymbol{s}) - \delta\boldsymbol{R}^{-1}\boldsymbol{B}\boldsymbol{L}^+\boldsymbol{B}^\top g(\boldsymbol{s})
\end{aligned}$$

where $\boldsymbol{r} = \frac{1}{(\boldsymbol{s}^+)^2} + \frac{1}{(\boldsymbol{s}^-)^2}$ and we have denoted $g(\boldsymbol{s}) = \frac{\frac{1}{\boldsymbol{s}^+} - \frac{1}{\boldsymbol{s}^-}}{\boldsymbol{r}}$.

**Fact 4.3.3.** *Using known scaling arguments, can assume that costs and capacities are bounded by* $\mathrm{poly}(m)$*, while only incurring an extra logarithmic dependence in the largest network parameter [71].*

We also use the fact that the resistances in the interior point method are never too large, which is proved in Appendix 9.2.2.

**Fact 4.3.4.** *For any $\mu \in (1/\mathrm{poly}(m), \mathrm{poly}(m))$, we have $\|\boldsymbol{r}(\mu)\|_\infty \leq m^{\widetilde{O}(\log m)}$.*

### 4.3.2 Making progress with approximate electrical flows

The following lemma shows that we can make $k$ steps of the interior point method by computing $O(k^4)$ $(1 + O(k^{-6}))$-*approximate* electrical flows. The proof is essentially the same as in [73], but we provide it for completeness in Appendix 9.2.3.

**Lemma 4.3.5.** *Let $\boldsymbol{f}^1, \ldots, \boldsymbol{f}^{T+1}$ be flows with slacks $\boldsymbol{s}^t$ and resistances $\boldsymbol{r}^t$ for $t \in [T+1]$, where $T = \frac{k}{\varepsilon_{\mathrm{step}}}$ for some $k \leq \sqrt{m}/10$ and $\varepsilon_{\mathrm{step}} = 10^{-5}k^{-3}$, such that*

- *$\boldsymbol{f}^1$ is $(\mu, 1 + \varepsilon_{\mathrm{solve}}/8)$-central for $\varepsilon_{\mathrm{solve}} = 10^{-5}k^{-3}$*
- *For all $t \in [T]$ and $e \in E$, $f_e^{t+1} = \begin{cases} f_e(\mu) + \varepsilon_{\mathrm{step}} \sum_{i=1}^{t} \widetilde{f}_e^i & \text{if } \exists i \in [t] : \widetilde{f}_e^i \neq 0 \\ f_e^1 & \text{otherwise} \end{cases}$, where*

$$\widetilde{\boldsymbol{f}}^{*t} = \delta g(\boldsymbol{s}^t) - \delta(\boldsymbol{R}^t)^{-1}\boldsymbol{B}\left(\boldsymbol{B}^\top(\boldsymbol{R}^t)^{-1}\boldsymbol{B}\right)^+\boldsymbol{B}^\top g(\boldsymbol{s}^t)$$

*for $\delta = \frac{1}{\sqrt{m}}$ and*

$$\left\| \sqrt{\boldsymbol{r}^t}\left(\widetilde{\boldsymbol{f}}^{*t} - \widetilde{\boldsymbol{f}}^t\right) \right\|_\infty \leq \varepsilon$$

*for $\varepsilon = 10^{-6}k^{-6}$.*

*Then, setting* $\varepsilon_{\text{step}} = \varepsilon_{\text{solve}} = 10^{-5}k^{-3}$ *and* $\varepsilon = 10^{-6}k^{-6}$ *we get that* $\boldsymbol{s}^{T+1} \approx_{1.1} \boldsymbol{s}\left(\mu/(1+\varepsilon_{\text{step}}\delta)^{k\varepsilon_{\text{step}}^{-1}}\right)$.

From now and for the rest of Section 4.3 we fix the values of $\varepsilon_{\text{step}}, \varepsilon_{\text{solve}}, \varepsilon$ based on this lemma. Using this lemma together with the following recentering procedure also used in [73], we can exactly compute a $\left(\mu/(1+\varepsilon_{\text{step}}/\sqrt{m})^{k\varepsilon_{\text{step}}^{-1}}\right)$-central flow.

**Lemma 4.3.6.** *Given a flow* $\boldsymbol{f}$ *with slacks* $\boldsymbol{s}$ *such that* $\boldsymbol{s} \approx_{1.1} \boldsymbol{s}(\mu)$ *for some* $\mu > 0$*, we can compute* $\boldsymbol{f}(\mu)$ *in* $\widetilde{O}(m)$.

### 4.3.3 The Locator data structure

From the previous lemma it becomes obvious that the only thing left is to maintain in sublinear time an approximation to

$$\delta g(\boldsymbol{s}^t) - \delta(\boldsymbol{R}^t)^{-1}\boldsymbol{B}(\boldsymbol{B}^\top(\boldsymbol{R}^t)^{-1}\boldsymbol{B})^+\boldsymbol{B}^\top g(\boldsymbol{s}^t)\,.$$

for $\delta = 1/\sqrt{m}$. This is the job of the $(\alpha,\beta,\varepsilon)$-Locator data structure, which computes all the entries of this vector that have magnitude $\geq \varepsilon$. We note that the guarantees of this data structure are similar to the ones in [73], but our locator requires an extra parameter $\alpha$ which is a measure of how much resistances can deviate before a full recomputation has to be made.

**Definition 4.3.7** ($(\alpha,\beta,\varepsilon)$-Locator)**.** *An* $(\alpha,\beta,\varepsilon)$-Locator *is a data structure that maintains valid slacks* $\boldsymbol{s}$ *and resistances* $\boldsymbol{r}$*, and can support the following operations against oblivious adversaries with high probability:*

- Initialize($\boldsymbol{f}$)*: Set* $\boldsymbol{s}^+ = \boldsymbol{u} - \boldsymbol{f}$*,* $\boldsymbol{s}^- = \boldsymbol{f}$*,* $\boldsymbol{r} = \frac{1}{(\boldsymbol{s}^+)^2} + \frac{1}{(\boldsymbol{s}^-)^2}$.
- Update($e, \boldsymbol{f}$)*: Set* $s_e^+ = u_e - f_e$*,* $s_e^- = f_e$*,* $r_e = \frac{1}{(s_e^+)^2} + \frac{1}{(s_e^-)^2}$*. Works under the condition that*

$$r_e^{\max}/\alpha \leq r_e \leq \alpha \cdot r_e^{\min}\,,$$

  *where* $r_e^{\max}$ *and* $r_e^{\min}$ *are the maximum and minimum resistance values that edge* $e$ *has had since the last call to* BatchUpdate.

- BatchUpdate($Z, \boldsymbol{f}$)*: Set* $s_e^+ = u_e - f_e, s_e^- = f_e, r_e = \frac{1}{(s_e^+)^2} + \frac{1}{(s_e^-)^2}$ *for all* $e \in Z$.
- Solve()*: Let*

$$\widetilde{\boldsymbol{f}}^* = \delta g(\boldsymbol{s}) - \delta\boldsymbol{R}^{-1}\boldsymbol{B}(\boldsymbol{B}^\top\boldsymbol{R}^{-1}\boldsymbol{B})^+\boldsymbol{B}^\top g(\boldsymbol{s})\,, \tag{4.7}$$

  *where* $\delta = \frac{1}{\sqrt{m}}$*. Returns an edge set* $Z$ *of size* $\widetilde{O}\left(\varepsilon^{-2}\right)$ *that with high probability contains all* $e$ *such that* $\sqrt{r_e}\left|\widetilde{f}_e^*\right| \geq \varepsilon$.

*The data structure works as long as the total number of calls to* Update*, plus the sum of* $|Z|$ *for all calls to* BatchUpdate *is* $O(\beta m)$.

In Section 4.4 we will prove the following lemma, which constructs an $(\alpha,\beta,\varepsilon)$-Locator and outlines its runtime guarantees:

**Lemma 4.3.8** (Efficient $(\alpha, \beta, \varepsilon)$-LOCATOR)**.** *For any graph $G(V, E)$ and parameters $\alpha \geq 1$, $\beta \in (0, 1)$, $\varepsilon \geq \widetilde{\Omega}\left(\beta^{-2}m^{-1/2}\right)$, and $\widehat{\varepsilon} \in \left(\widetilde{\Omega}\left(\beta^{-2}m^{-1/2}\right), \varepsilon\right)$, there exists an $(\alpha, \beta, \varepsilon)$-LOCATOR for $G$ with the following runtimes per operation:*

- INITIALIZE($\boldsymbol{f}$)*:* $\widetilde{O}\left(m \cdot \left(\widehat{\varepsilon}^{-4}\beta^{-8} + \widehat{\varepsilon}^{-2}\varepsilon^{-2}\alpha^2\beta^{-4}\right)\right)$.
- UPDATE($e, \boldsymbol{f}$)*:* $\widetilde{O}\left(m \cdot \frac{\widehat{\varepsilon}\alpha^{1/2}}{\varepsilon^3} + \widehat{\varepsilon}^{-4}\varepsilon^{-2}\beta^{-8} + \widehat{\varepsilon}^{-2}\varepsilon^{-4}\alpha^2\beta^{-6}\right)$ *amortized.*
- BATCHUPDATE($Z, \boldsymbol{f}$)*:* $\widetilde{O}\left(m \cdot \frac{1}{\varepsilon^2} + |Z| \cdot \frac{1}{\varepsilon^2\beta^2}\right)$.
- SOLVE()*:* $\widetilde{O}\left(\beta m \cdot \frac{1}{\varepsilon^2}\right)$.

Note that even though a LOCATOR computes a set that contains all $\varepsilon$-congested edges, it does not return the actual flow values. The reason for that is that it only works against oblivious adversaries, and allowing (randomized) flow values to affect future updates constitutes an adaptive adversary. As in [73], we resolve this by sanitizing the outputs through a different data structure called CHECKER, which computes the flow values and works against semi-adaptive adversaries. As the definition and implementation of CHECKER is orthogonal to our contribution and also does not affect the final runtime, we defer the discussion to Appendix 9.2.6. To simplify the presentation in this section, we instead define the following idealized version of it, called PERFECTCHECKER.

**Definition 4.3.9** ($\varepsilon$-PERFECTCHECKER)**.** *For any error $\varepsilon > 0$, an $\varepsilon$-PERFECTCHECKER is an oracle that given a graph $G(V, E)$, slacks $\boldsymbol{s}$, resistances $\boldsymbol{r}$, supports the following operations:*

- UPDATE($e, \boldsymbol{f}$)*: Set $s_e^+ = u_e - f_e$, $s_e^- = f_e$, $r_e = \frac{1}{(s_e^+)^2} + \frac{1}{(s_e^-)^2}$.*
- CHECK($e$)*: Compute a flow value $\widetilde{f}_e$ such that $\sqrt{r_e}\left|\widetilde{f}_e - \widetilde{f}_e^*\right| \leq \varepsilon$, where*

$$\widetilde{\boldsymbol{f}}^* = \delta g(\boldsymbol{s}) - \delta \boldsymbol{R}^{-1}\boldsymbol{B}(\boldsymbol{B}^\top \boldsymbol{R}^{-1}\boldsymbol{B})^+ \boldsymbol{B}^\top g(\boldsymbol{s}),$$

*with $\delta = 1/\sqrt{m}$. If $\sqrt{r_e}\left|\widetilde{f}_e\right| < \varepsilon/2$ return 0, otherwise return $\widetilde{f}_e$.*

### 4.3.4 The minimum cost flow algorithm

Now, we will show how the data structure defined in Section 4.3.3 can be used to make progress along the central path. The main lemma that analyzes the performance of the minimum cost flow algorithm given access to an $(\alpha, \beta, \varepsilon)$-LOCATOR is Lemma 4.3.10. Also, the skeleton of the algorithm is described in Algorithm 1.

**Lemma 4.3.10** (MINCOSTFLOW)**.** *Let $\mathcal{L}$ be an $(\alpha, \beta, \varepsilon)$ − LOCATOR, $\boldsymbol{f}$ be a $\mu$-central flow where $\mu = \text{poly}(m)$, and $k \in \left[m^{1/316}\right]$, $\beta \geq \widetilde{\Omega}\left(k^3/m^{1/4}\right)$, $\widehat{T} \in \left[\widetilde{O}\left(m^{1/2}/k\right)\right]$ be some parameters. There is an algorithm that with high probability computes a $\mu'$-central flow $\boldsymbol{f}'$, where $\mu' \leq m^{-10}$. Additionally, the algorithm runs in time $\widetilde{O}\left(m^{3/2}/k\right)$, plus*

- $\widetilde{O}\left(k^3\beta^{-1/2}\right)$ *calls to* $\mathcal{L}$.INITIALIZE*,*
- $\widetilde{O}\left(m^{1/2}k^3\right)$ *calls to* $\mathcal{L}$.SOLVE*,*

**Algorithm 1** Minimum Cost Flow

**procedure** MinCostFlow($G, \boldsymbol{c}, \boldsymbol{d}, \boldsymbol{u}$)
  $\bar{\boldsymbol{f}}, \mu = \textsc{Initialize}(G, \boldsymbol{c}, \boldsymbol{d}, \boldsymbol{u})$ ▷ Lemma 4.3.12. $\bar{\boldsymbol{f}}$ is $\mu$-central at all times.
  $i = 0$
  **while** $\mu \geq m^{-10}$ **do**
    **if** $i$ is a multiple of $\lfloor \varepsilon_{\text{solve}} \sqrt{\beta m}/k \rfloor$ **then** ▷ Re-initialize when $|C|$ exceeds $O(\beta m)$.
      $\mathcal{L} = \textsc{Locator.Initialize}(\bar{\boldsymbol{f}})$ with error $\varepsilon/2$
    **if** $i$ is a multiple of $\lfloor \varepsilon_{\text{solve}} \sqrt{\beta_{\textsc{Checker}} m}/k \rfloor$ **then**
      $\mathcal{C}^i = \textsc{Checker.Initialize}(\bar{\boldsymbol{f}}, \varepsilon, \beta_{\textsc{Checker}})$ for $i \in [k\varepsilon_{\text{step}}^{-1}]$
    **if** $i$ is a multiple of $\lfloor 0.5\alpha^{1/4}/k - 1 \rfloor$ **then** ▷ Update important edges when $\mathcal{L}.\boldsymbol{r}^0$ expires
      $\mathcal{L}.\textsc{BatchUpdate}(\emptyset)$
    $\bar{\boldsymbol{f}}, \mu = \textsc{MultiStep}(\bar{\boldsymbol{f}}, \mu)$
    **if** $i$ is a multiple of $\widehat{T}$ **then**
      $Z = \emptyset$
      **for** $e \in E$ **do**
        $\bar{s}_e^+ = u_e - \bar{f}_e$, $\bar{s}_e^- = \bar{f}_e$
        **if** $\bar{s}_e^+ \not\approx_{\varepsilon_{\text{solve}}/16} \mathcal{L}.s_e^+$ or $\bar{s}_e^- \not\approx_{\varepsilon_{\text{solve}}/16} \mathcal{L}.s_e^-$ **then**
          $\mathcal{C}^i.\textsc{Update}(e, \bar{\boldsymbol{f}})$ for $i \in [k\varepsilon_{\text{step}}^{-1}]$
          $Z = Z \cup \{e\}$
      $\mathcal{L}.\textsc{BatchUpdate}(Z, \bar{\boldsymbol{f}})$
    **else**
      **for** $e \in E$ **do**
        $\bar{s}_e^+ = u_e - \bar{f}_e$, $\bar{s}_e^- = \bar{f}_e$
        **if** $\bar{s}_e^+ \not\approx_{\varepsilon_{\text{solve}}/8} \mathcal{L}.s_e^+$ or $\bar{s}_e^- \not\approx_{\varepsilon_{\text{solve}}/8} \mathcal{L}.s_e^-$ **then**
          $\mathcal{C}^i.\textsc{Update}(e, \bar{\boldsymbol{f}})$ for $i \in [k\varepsilon_{\text{step}}^{-1}]$
          $\mathcal{L}.\textsc{Update}(e, \bar{\boldsymbol{f}})$
    $i = i + 1$
  **return** $\textsc{Round}(G, \boldsymbol{c}, \boldsymbol{d}, \boldsymbol{u}, \bar{\boldsymbol{f}})$ ▷ Lemma 4.3.13

- $\widetilde{O}\left(m^{1/2}\left(k^6\widehat{T}+k^{15}\right)\right)$ *calls to* $\mathcal{L}$.UPDATE,
- $\widetilde{O}\left(m^{1/2}\alpha^{-1/4}\right)$ *calls to* $\mathcal{L}$.BATCHUPDATE($\emptyset$), *and*
- $\widetilde{O}\left(m^{1/2}k^{-1}\widehat{T}^{-1}\right)$ *calls to* $\mathcal{L}$.BATCHUPDATE($Z, \bar{\boldsymbol{f}}$) *for some* $Z \neq \emptyset, \bar{\boldsymbol{f}}$. *Additionally, the sum of* $|Z|$ *over all such calls is* $\widetilde{O}\left(mk^3\beta^{1/2}\right)$.

The proof appears in Appendix 9.2.3. Its main ingredient is the following lemma, which easily follows from Lemma 4.3.5 and essentially shows how $k$ steps of the interior point method can be performed in $\widetilde{O}(m)$ instead of $\widetilde{O}(mk)$. Its proof appears in Appendix 9.2.3.

**Lemma 4.3.11** (MULTISTEP)**.** *Let* $k \in \{1, \ldots, \sqrt{m}/10\}$. *We are given* $\boldsymbol{f}(\mu)$, *an* $(\alpha, \beta, \varepsilon/2)$-LOCATOR $\mathcal{L}$, *and an* $\varepsilon$-PERFECTCHECKER $\mathcal{C}$, *such that*

- $\mathcal{L}.\boldsymbol{r} = \mathcal{C}.\boldsymbol{r}$ *are* $(\mu, 1+\varepsilon_{\text{solve}}/8)$*-central resistances, and*
- $\mathcal{L}.\boldsymbol{r}^0$ *are* $(\mu^0, 1+\varepsilon_{\text{solve}}/8)$*-central resistances, where* $\mu^0 \leq \mu \cdot (1+\varepsilon_{\text{step}}/\sqrt{m})^{\widehat{T}}$ *and* $\widehat{T} = (0.5\alpha^{1/4}-k)\varepsilon_{\text{step}}^{-1}$. *Additionally, for any resistances* $\widehat{\boldsymbol{r}}$ *that* $\mathcal{L}$ *had at any point since the last call to* $\mathcal{L}$.BATCHUPDATE, $\widehat{\boldsymbol{r}}$ *are* $(\hat{\mu}, 1.1)$*-central for some* $\hat{\mu} \in [\mu, \mu^0]$.

*Then, there is an algorithm that with high probability computes* $\boldsymbol{f}(\mu')$, *where* $\mu' = \mu/(1+\varepsilon_{\text{step}}/\sqrt{m})^{k\varepsilon_{\text{step}}^{-1}}$. *The algorithm runs in time* $\widetilde{O}(m)$, *plus* $O(k^{16})$ *calls to* $\mathcal{L}$.UPDATE, $O(k^4)$ *calls to* $\mathcal{L}$.SOLVE, *and* $O(k^{16})$ *calls to* $\mathcal{C}$.UPDATE *and* $\mathcal{C}$.CHECK. *Additionally,* $\mathcal{L}.\boldsymbol{r}$ *and* $\mathcal{C}$ *are unmodified.*

**Algorithm 2** MultiStep

**procedure** MULTISTEP($\boldsymbol{f}, \mu$) ▷ Makes equivalent progress to $k$ interior point method steps
  $\widehat{\boldsymbol{r}} = \mathcal{L}.\boldsymbol{r}$ ▷ Save resistances to restore later
  **for** $i = 1, \ldots, k\varepsilon_{\text{step}}^{-1}$ **do**
    $Z = \mathcal{L}$.SOLVE()
    **for** $e \in Z$ **do** ▷ $Z$: Set of edges with sufficiently changed flow
      $\widetilde{f}_e = \mathcal{C}^i$.CHECK($e$)
      **if** $\widetilde{f}_e \neq 0$ **then**
        $f_e = f_e + \varepsilon_{\text{step}}\widetilde{f}_e$
        $\mathcal{L}$.UPDATE($e, \boldsymbol{f}$)
        $\mathcal{C}^j$.TEMPORARYUPDATE($e, \boldsymbol{f}$) for $j \in \left[i+1, k\varepsilon_{\text{step}}^{-1}\right]$
  $\mu = \mu/(1+\varepsilon_{\text{step}}/\sqrt{m})^{k\varepsilon_{\text{step}}^{-1}}$
  $\boldsymbol{f}$ = RECENTER($\boldsymbol{f}, \mu$) ▷ Lemma 4.3.6
  **for** $e \in E$ **do**
    **if** $\mathcal{L}.r_e \neq \widehat{r}_e$ **then**
      $\mathcal{L}$.UPDATE($e, \widehat{\boldsymbol{r}}$) ▷ Return LOCATOR resistances to their original state
  Call $\mathcal{C}^i$.ROLLBACK() to undo all TEMPORARYUPDATEs for all $\mathcal{C}^i$
  **return** $\boldsymbol{f}, \mu$

### 4.3.5 Proof of Theorem 4.1.1

**Correctness.** First of all, we apply capacity and cost scaling [71] to make sure that $\|\boldsymbol{c}\|_\infty, \|\boldsymbol{u}\|_\infty = \text{poly}(m)$. These incur an extra factor of $\log(U+W)$ in the runtime.

We first get an initial solution to the interior point method by using the following lemma:

**Lemma 4.3.12** (Interior point method initialization, Appendix A in [10])**.** *Given a min cost flow instance $\mathcal{I} = (G(V,E), \boldsymbol{c}, \boldsymbol{d}, \boldsymbol{u})$, there exists an algorithm that runs in time $O(m)$ and produces a new min cost flow instance $\mathcal{I}' = \left(G'(V',E'), \boldsymbol{c}', \boldsymbol{d}', \boldsymbol{u}'\right)$, where $|V'| = O(|V|)$ and $|E'| = O(|E|)$, as well as a flow $\boldsymbol{f}$ such that*

- *$\boldsymbol{f}$ is $\mu$-central for $\mathcal{I}'$ for some $\mu = \Theta\left(\|\boldsymbol{c}\|_2\right)$*
- *Given an optimal solution for $\mathcal{I}'$, an optimal minimum cost flow solution for $\mathcal{I}$ can be computed in $O(m)$*

Therefore we now have a $\text{poly}(m)$-central solution for an instance $\mathcal{I}$. We can now apply Lemma 4.3.10 to get a $\mu'$-central solution with $\mu' \leq m^{-10}$. Then we can apply the following lemma to round the solution, which follows from Lemma 5.4 in [10].

**Lemma 4.3.13** (Interior point method rounding)**.** *Given a min cost flow instance $\mathcal{I}$ and a $\mu$-central flow $\boldsymbol{f}$ for $\mu \leq m^{-10}$, there is an algorithm that runs in time $\widetilde{O}(m)$ and returns an optimal integral flow.*

By Lemma 4.3.12, this solution can be turned into an exact solution for the original instance. As Lemma 4.3.10 succeeds with high probability, the whole algorithm does too.

**Runtime.** To determine the final runtime, we analyze each operation in Algorithm 1 separately.

The Initialize (Lemma 4.3.12) and Round (Lemma 4.3.13) operations take time $\widetilde{O}\left(m\right)$. Now, the runtime of Lemma 4.3.10 is $\widetilde{O}\left(m^{3/2}/k\right)$ plus the runtime incurred because of calls to the locator $\mathcal{L}$. We will use the runtimes per operation from Lemma 4.3.8.

$\mathcal{L}$.Solve: This operation is run $\widetilde{O}\left(m^{1/2}k^3\right)$ times, and each of these costs $\widetilde{O}\left(\frac{\beta m}{\varepsilon^2}\right) = \widetilde{O}\left(mk^{12}\beta\right)$. Therefore in total $\widetilde{O}\left(m^{3/2}k^{15}\beta\right)$.

We pick $\beta$ by $m^{3/2}k^{15}\beta \leq m^{3/2}/k$ as $\beta = k^{-16}$, so the runtime is $\widetilde{O}\left(m^{3/2}/k\right)$. Note that this satisfies the constraint $\beta \geq \widetilde{\Omega}\left(k^3/m^{1/4}\right)$ as long as $k \leq \widetilde{\Omega}\left(m^{1/76}\right)$.

$\mathcal{L}$.BatchUpdate: This is run $\widetilde{O}\left(m^{1/2}/\alpha^{1/4}\right)$ times with empty arguments, each of which takes time $\widetilde{O}\left(m/\varepsilon^2\right) = \widetilde{O}\left(mk^{12}\right)$. The total runtime because of these is $\widetilde{O}\left(m^{3/2}k^{12}\alpha^{-1/4}\right)$. As we need this to be below $\widetilde{O}\left(m^{3/2}/k\right)$, we set $\alpha = k^{52}$.

This operation is also run $\widetilde{O}\left(m^{1/2}k^{-1}\widehat{T}^{-1}\right)$ times with some non-empty argument $Z$, each of which takes time $\widetilde{O}\left(m/\varepsilon^2 + |Z|/(\varepsilon^2\beta^2)\right) = \widetilde{O}\left(mk^{12} + k^{44}|Z|\right)$. As by Lemma 4.3.10 the total sum of $|Z|$ over all calls is $\widetilde{O}\left(mk^3\beta^{1/2}\right) = \widetilde{O}\left(mk^{-5}\right)$, we get a runtime of

$$\widetilde{O}\left(m^{1/2}k^{-1}\widehat{T}^{-1} \cdot mk^{12} + k^{44} \cdot mk^{-5}\right) = \widetilde{O}\left(m^{3/2}k^{11}\widehat{T}^{-1} + mk^{39}\right).$$

In order to set the first term to be at most $\widetilde{O}\left(m^{3/2}/k\right)$, we set $\widehat{T} = k^{12}$.

Therefore the total runtime of this operation is $\widetilde{O}\left(m^{3/2}/k + mk^{39}\right)$.

$\mathcal{L}$.UPDATE: This is run $\widetilde{O}\left(m^{1/2}\left(k^6\widehat{T}+k^{15}\right)\right)=\widetilde{O}\left(m^{1/2}k^{18}\right)$ times and the amortized cost per operation is

$$\widetilde{O}\left(m\cdot\frac{\widehat{\varepsilon}\alpha^{1/2}}{\varepsilon^3}+\widehat{\varepsilon}^{-4}\varepsilon^{-2}\beta^{-8}+\widehat{\varepsilon}^{-2}\varepsilon^{-4}\alpha^2\beta^{-6}\right)$$
$$=\widetilde{O}\left(m\cdot k^{44}\widehat{\varepsilon}+k^{140}\widehat{\varepsilon}^{-4}+k^{224}\widehat{\varepsilon}^{-2}\right),$$

so in total

$$m^{3/2}k^{62}\widehat{\varepsilon}+m^{1/2}k^{158}\widehat{\varepsilon}^{-4}+m^{1/2}k^{242}\widehat{\varepsilon}^{-2}.$$

As we need the first term to be $\widetilde{O}\left(m^{3/2}/k\right)$, we set $\widehat{\varepsilon}=k^{-63}$. Therefore the total runtime is

$$\widetilde{O}\left(m^{3/2}/k+m^{1/2}k^{410}+m^{1/2}k^{368}\right)=\widetilde{O}\left(m^{3/2}/k+m^{1/2}k^{410}\right).$$

$\mathcal{L}$.INITIALIZE: This is run $\widetilde{O}\left(k^3\beta^{-1/2}\right)=k^{11}$ times in total, and the runtime for each run is

$$\widetilde{O}\left(m\cdot\left(\widehat{\varepsilon}^{-4}\beta^{-8}+\widehat{\varepsilon}^{-2}\varepsilon^{-2}\alpha^2\beta^{-4}\right)\right)=\widetilde{O}\left(m\cdot\left(k^{380}+k^{306}\right)\right)=\widetilde{O}\left(m\cdot k^{380}\right),$$

so in total $\widetilde{O}\left(mk^{380}\right)$.

Therefore, for the whole algorithm, we get $\widetilde{O}\left(m^{3/2}/k+m^{1/2}k^{410}+mk^{380}\right)$ which after balancing gives $k=m^{1/762}$.

## 4.4 An Efficient $(\alpha,\beta,\varepsilon)$-LOCATOR

In this section we will show how to implement an $(\alpha,\beta,\varepsilon)$-LOCATOR, as defined in Definition 4.3.7. In order to maintain the approximate electrical flow $\widetilde{\boldsymbol{f}}$ required by Lemma 4.3.8 we will keep a vertex sparsifier in the form of a sparsified Schur complement onto some vertex set $C$. As in [73], we choose $C$ to be a *congestion reduction subset*.

**Definition 4.4.1** (Congestion reduction subset [73])**.** *Given a graph $G(V,E)$ with resistances $\boldsymbol{r}$ and any parameter $\beta\in(0,1)$, a vertex subset $C\subseteq V$ is called a* $\beta$-congestion reduction subset *(or just* congestion reduction subset*) if:*

- $|C|\leq O(\beta m)$
- *For any $u\in V$, a random walk starting from $u$ that visits $\widetilde{\Omega}\left(\beta^{-1}\log n\right)$ distinct vertices hits $C$ with high probability*
- *If we generate $\deg(u)$ random walks from each $u\in V\backslash C$, the expected number of these that hit some fixed $v\in V\backslash C$ before $C$ is $\widetilde{O}\left(1/\beta^2\right)$. Concretely:*

$$\sum_{u\in V}\deg(u)\cdot p_v^{C\cup\{v\}}(u)\leq\widetilde{O}\left(1/\beta^2\right). \tag{4.8}$$

The following lemma shows that such a vertex subset can be constructed efficiently:

**Lemma 4.4.2** (Construction of congestion reduction subset [73])**.** *Given a graph $G(V,E)$ with resistances $\boldsymbol{r}$ and a parameter $\beta\in(0,1)$, there is an algorithm that generates a $\beta$-congestion reduction subset in time $\widetilde{O}\left(m/\beta^2\right)$.*

Intuitively, (4.8) says that "not too many" random walks go through a given vertex before reaching $C$. This property is crucial for ensuring that when inserting a new vertex into $C$, the data structure will not have to change too much. As we will see in Section 4.4.1, this property plays an even more central role when general demands are introduced, as it allows us to show that the demands outside $C$ can be pushed to $C$. Additionally, in Section 4.4.2 we will use it to show that edges that are too far from $C$ in effective resistance metric are not *important*, in the sense that neither can they get congested, nor can their demand congest anything else.

### 4.4.1 Moving demands to the sparsifier

The goal of this section is to show that if $C$ is a congestion reduction subset, then any demand of the form $\boldsymbol{d} = \boldsymbol{B}^\top \frac{\boldsymbol{q}}{\sqrt{\boldsymbol{r}}}$ for some $\boldsymbol{q} \in [-1,1]^m$ can be approximated by $\boldsymbol{\pi}^C(\boldsymbol{d})$, i.e. its demand projection onto $C$ (Definition 4.2.4). This allows us to move all demands to the sublinear-sized $C$ and thus enables us to work with the Schur complement of $G$ onto $C$.

**Lemma 4.4.3.** *Consider a graph $G(V,E)$ with resistances $\boldsymbol{r}$ and Laplacian $\boldsymbol{L}$, a $\beta$-congestion reduction subset $C$, and a demand $\boldsymbol{d} = \delta \boldsymbol{B}^\top \frac{\boldsymbol{q}}{\sqrt{\boldsymbol{r}}}$ for some $\delta > 0$ and $\boldsymbol{q} \in [-1,1]^m$. Then, the potential embedding defined as*

$$\boldsymbol{\phi} = \boldsymbol{L}^+ \left( \boldsymbol{d} - \boldsymbol{\pi}^C(\boldsymbol{d}) \right)$$

*has congestion $\delta \cdot \widetilde{O}\left(1/\beta^2\right)$, i.e. $\left\| \frac{\boldsymbol{B}\boldsymbol{\phi}}{\sqrt{\boldsymbol{r}}} \right\|_\infty \leq \delta \cdot \widetilde{O}\left(1/\beta^2\right)$.*

We first prove a restricted version of the lemma where $\boldsymbol{d}$ is an $s-t$ demand. Then, Lemma 4.4.3 follows trivially by applying (4.8).

**Lemma 4.4.4.** *Consider a graph $G(V,E)$ with resistances $\boldsymbol{r}$ and Laplacian $\boldsymbol{L}$, a $\beta$-congestion reduction subset $C$, and a demand $\boldsymbol{d} = \delta \boldsymbol{B}^\top \frac{\mathbf{1}_{st}}{\sqrt{\boldsymbol{r}}}$ for some $\delta > 0$ and $(s,t) \in E \backslash E(C)$. Then, for the potential embedding defined as*

$$\boldsymbol{\phi} = \boldsymbol{L}^+ \left( \boldsymbol{d} - \boldsymbol{\pi}^C(\boldsymbol{d}) \right)$$

*it follows that for any $e = (u,v) \in E$ we have*

$$\left| \frac{(\boldsymbol{B}\boldsymbol{\phi})_e}{\sqrt{r_e}} \right| \leq 2\delta \cdot \left( p_u^{C \cup \{u\}}(s) + p_v^{C \cup \{v\}}(s) + p_u^{C \cup \{u\}}(t) + p_v^{C \cup \{v\}}(t) \right).$$

The proof of Lemma 4.4.4 appears in Appendix 9.2.4.

### 4.4.2 $\varepsilon$-Important edges

In this section we will show that the effect of edges that are "far" from the congestion reduction subset $C$ is negligible, as both their congestion and the congestion incurred because of their demands are small. More specifically, given a demand $\boldsymbol{d}$ supported on $C$ with energy $\leq 1$, i.e. $\mathcal{E}_{\boldsymbol{r}}(\boldsymbol{d}) \leq 1$, the

congestion $\boldsymbol{\rho} = \boldsymbol{R}^{-1/2}\boldsymbol{B}\boldsymbol{L}^{+}\boldsymbol{d}_C$ that it induces satisfies:

$$\begin{aligned}
|\rho_e| &= \left|\left\langle \mathbf{1}_e, \boldsymbol{R}^{-1/2}\boldsymbol{B}\boldsymbol{L}^{+}\boldsymbol{d}\right\rangle\right| \\
&= \left|\left\langle \boldsymbol{B}^{\top}\frac{\mathbf{1}_e}{\sqrt{\boldsymbol{r}}}, \boldsymbol{L}^{+}\boldsymbol{d}\right\rangle\right| \\
&= \left|\left\langle \boldsymbol{\pi}^{C}\left(\boldsymbol{B}^{\top}\frac{\mathbf{1}_e}{\sqrt{\boldsymbol{r}}}\right), SC^{+}\boldsymbol{d}_C\right\rangle\right| \\
&\leq \sqrt{\mathcal{E}_{\boldsymbol{r}}\left(\boldsymbol{\pi}^{C}\left(\boldsymbol{B}^{\top}\frac{\mathbf{1}_e}{\sqrt{\boldsymbol{r}}}\right)\right)\mathcal{E}_{\boldsymbol{r}}\left(\boldsymbol{d}\right)} \\
&\leq \sqrt{\mathcal{E}_{\boldsymbol{r}}\left(\boldsymbol{\pi}^{C}\left(\boldsymbol{B}^{\top}\frac{\mathbf{1}_e}{\sqrt{\boldsymbol{r}}}\right)\right)}\,.
\end{aligned}$$

For the last equality we used Fact 4.2.5, for the first inequality we applied Cauchy-Schwarz, and for the second one we used the upper bound on the energy required to route $\boldsymbol{d}$. Therefore, if we bound the energy of the projection of $\boldsymbol{B}^{\top}\frac{\mathbf{1}_e}{\sqrt{\boldsymbol{r}}}$ onto $C$, we can also bound the congestion of $e$. This is done in the following lemma, whose proof appears in Appendix 9.2.4.

**Lemma 4.4.5.** *Consider a graph $G(V,E)$ with resistances $\boldsymbol{r}$ and $C \subseteq V$. Then, for all $e \in E\backslash E(C)$ we have*

$$\sqrt{\mathcal{E}_r\left(\boldsymbol{\pi}^{C}\left(\boldsymbol{B}^{\top}\frac{\mathbf{1}_e}{\sqrt{r_e}}\right)\right)} \leq 6\cdot\sqrt{\frac{r_e}{R_{eff}(C,e)}}\,.$$

This is the consequence of the following lemma, which bounds the magnitude of the projection on a specific vertex, based on its effective resistance distance from $e$, as well as hitting probabilities from $e$ to $C$. The proof appears in Appendix 9.2.5.

**Lemma 4.4.6.** *Consider a graph $G(V,E)$ with resistances $\boldsymbol{r}$ and a subset of vertices $C \subseteq V$. For any vertex $v \in V\backslash C$ we have that*

$$\left|\pi_v^{C\cup\{v\}}\left(\boldsymbol{B}^{\top}\frac{\mathbf{1}_e}{\sqrt{\boldsymbol{r}}}\right)\right| \leq (p_v^{C\cup\{v\}}(u) + p_v^{C\cup\{v\}}(w))\cdot\frac{\sqrt{r_e}}{R_{eff}(v,e)}\,.$$

This lemma is complementary to the more immediate property

$$\left|\pi_v^{C\cup\{v\}}\left(\boldsymbol{B}^{\top}\frac{\mathbf{1}_e}{\sqrt{\boldsymbol{r}}}\right)\right| \leq (p_v^{C\cup\{v\}}(u) + p_v^{C\cup\{v\}}(w))\cdot\frac{1}{\sqrt{r_e}}\,,$$

and they are both used in Section 4.5 in order to estimate demand projections. In fact, the just by multiplying these two, we get the following lemma, which is nice because it doesn't depend on $r_e$:

**Lemma 4.4.7.** *Consider a graph $G(V,E)$ with resistances $\boldsymbol{r}$ and a subset of vertices $C \subseteq V$. For any vertex $v \in V\backslash C$ we have that*

$$\left|\pi_v^{C\cup\{v\}}\left(\boldsymbol{B}^{\top}\frac{\mathbf{1}_e}{\sqrt{\boldsymbol{r}}}\right)\right| \leq (p_v^{C\cup\{v\}}(u) + p_v^{C\cup\{v\}}(w))\cdot\frac{1}{\sqrt{R_{eff}(v,e)}}\,.$$

By the previous discussion, Lemma 4.4.6 implies that if $|\rho_e| \geq \varepsilon$, then $R_{eff}(C, e) \leq r_e \cdot \frac{36}{\varepsilon^2}$. This motivates the following definition of *$\varepsilon$-important edges*.

**Definition 4.4.8** (Important edges)**.** *An edge $e \in E$ is called* $\varepsilon$-important *(or just* important*) if $R_{eff}(C, e) \leq r_e/\varepsilon^2$.*

Now it is time for the main lemma of this section, which uses Lemma 4.4.5 to show that if our goal is to detect edges with congestion $\geq \varepsilon$, it is sufficient to restrict to computing demand projections of $\Omega(\varepsilon)$-important edges. Its proof appears in Appendix 9.2.4.

**Lemma 4.4.9** (Localization lemma)**.** *Let $\boldsymbol{\phi}^*$ be any solution of*

$$\boldsymbol{L}\boldsymbol{\phi}^* = \delta \cdot \boldsymbol{\pi}^C \left( \boldsymbol{B}^\top \frac{\boldsymbol{p}}{\sqrt{\boldsymbol{r}}} \right),$$

*where $\boldsymbol{r}$ are any resistances, $\boldsymbol{p} \in [-1, 1]^m$, and $C \subseteq V$. Then, for any $e \in E$ that is not $\varepsilon$-important we have $\left| \frac{\boldsymbol{B}\boldsymbol{\phi}^*}{\sqrt{\boldsymbol{r}}} \right|_e \leq 6\varepsilon$.*

### 4.4.3 Proving Lemma 4.3.8

Before moving to the description of how LOCATOR works and its proof, we will provide a lemma which bounds how fast a demand projection changes.

We will use the following observation, which states that if our congestion reduction subset $C$ contains an $\beta m$-sized uniformly random edge subset, then with high probability, effective resistance neighborhoods that are disjoint from $C$ only have $\widetilde{O}\left(\beta^{-1}\right)$ edges. Note that this is will be true throughout the algorithm as long as the resistances do not depend on the randomness of $C$. This is true, as resistance updates are only ever given as inputs to LOCATOR.

**Lemma 4.4.10** (Few edges in a small neighborhood)**.** *Let $\beta \in (0, 1)$ be a parameter and $C$ be a vertex set which contains a subset of $\beta m$ edges sampled at random. Then with high probability, for any $v \in V \backslash C$ we have that $|N_E(v, R_{eff}(C, v)/2)| \leq 10\beta^{-1} \ln m$, where*

$$N_E(v, R) := \{e \in E \mid R_{eff}(e, v) \leq R\}.$$

*Proof.* Suppose that for some vertex $v \in V \backslash C$, $|N_E(u, R_{eff}(C, v)/2)| \geq 10\beta^{-1} \ln m$. Since by construction $C$ contains a random edge subset of size $\beta m$, with high probability $N_E(u, R_{eff}(C, u)/2) \cap C \neq \emptyset$, so there exists $u \in C$ such that $R_{eff}(u, v) \leq R_{eff}(C, v)/2$. This is a contradiction since $u \in C$ implies $R_{eff}(C, v) \leq R_{eff}(u, v)$. Union bounding over all $v$ yields the claim. □

Using this fact, we can now show that the change of the demand projection (measured in energy) is quite mild. The proof of the following lemma can be found in Appendix 9.2.4.

**Lemma 4.4.11** (Projection change)**.** *Consider a graph $G(V, E)$ with resistances $\boldsymbol{r}$, $\boldsymbol{q} \in [-1, 1]^m$, and a $\beta$-congestion reduction subset $C$. Then, with high probability,*

$$\sqrt{\mathcal{E}_{\boldsymbol{r}} \left( \boldsymbol{\pi}^{C \cup \{v\}} \left( \boldsymbol{B}^\top \frac{\boldsymbol{q}}{\sqrt{\boldsymbol{r}}} \right) - \boldsymbol{\pi}^C \left( \boldsymbol{B}^\top \frac{\boldsymbol{q}}{\sqrt{\boldsymbol{r}}} \right) \right)} \leq \widetilde{O}\left(\beta^{-2}\right).$$

The above lemma can be applied over multiple vertex insertions and resistance changes, to bound the overall energy change. This is shown in the following lemma, which is proved in Appendix 9.2.4:

**Lemma 4.4.12.** *Consider a graph $G(V,E)$ with resistances $\boldsymbol{r}^0$, $\boldsymbol{q}^0 \in [-1,1]^m$, a $\beta$-congestion reduction subset $C^0$, and a fixed sequence of updates, where the $i$-th update $i \in \{0,\ldots,T-1\}$ is of the following form:*

- AddTerminal*$(v^i)$: Set $C^{i+1} = C^i \cup \{v^i\}$ for some $v^i \in V \backslash C^i$, $q_e^{i+1} = q_e^i, r_e^{i+1} = r_e^i$*
- Update*$(e^i, \boldsymbol{q}, \boldsymbol{r})$ Set $C^{i+1} = C^i$, $q_e^{i+1} = q_e$ $r_e^{i+1} = r_e$, where $e^i \in E(C^i)$*

*Then, with high probability,*

$$\sqrt{\mathcal{E}_{\boldsymbol{r}^T}\left(\boldsymbol{\pi}^{C^0,\boldsymbol{r}^0}\left(\boldsymbol{B}^\top \frac{\boldsymbol{q}_S^0}{\sqrt{\boldsymbol{r}^0}}\right) - \boldsymbol{\pi}^{C^T,\boldsymbol{r}^T}\left(\boldsymbol{B}^\top \frac{\boldsymbol{q}_S^T}{\sqrt{\boldsymbol{r}^T}}\right)\right)} \leq \widetilde{O}\left(\max_{i\in\{0,\ldots,T-1\}} \left\|\frac{\boldsymbol{r}^T}{\boldsymbol{r}^i}\right\|_\infty^{1/2} \beta^{-2}\right) \cdot T\,.$$

We are now ready to describe the Locator data structure. We will give an outline here, and defer the full proof to Appendix 9.2.4. The goal of an $(\alpha,\beta,\varepsilon)$-Locator is, given some flow $\boldsymbol{f}$ with slacks $\boldsymbol{s}$ and resistances $\boldsymbol{r}$, to compute all $e \in E$ such that $\sqrt{r_e}\left|\widetilde{f}_e^*\right| \geq \varepsilon$, where

$$\widetilde{\boldsymbol{f}}^* = \delta g(\boldsymbol{s}) - \delta \boldsymbol{R}^{-1}\boldsymbol{B}\boldsymbol{L}^+\boldsymbol{B}^\top g(\boldsymbol{s})$$

($\boldsymbol{L} = \boldsymbol{B}^\top\boldsymbol{R}^{-1}\boldsymbol{B}$), where $\delta = 1/\sqrt{m}$.

If we set $\rho_e^* = \sqrt{r_e}\widetilde{f}_e^*$, we can equivalently write

$$\boldsymbol{\rho}^* = \delta\sqrt{\boldsymbol{r}}g(\boldsymbol{s}) - \delta\boldsymbol{R}^{-1/2}\boldsymbol{B}\boldsymbol{L}^+\boldsymbol{B}^\top g(\boldsymbol{s})\,,$$

and require to find all the entries of $\boldsymbol{\rho}^*$ with magnitude at least $\varepsilon$. As $\delta\left\|\sqrt{\boldsymbol{r}}g(\boldsymbol{s})\right\|_\infty \leq \delta \leq \varepsilon/100$, we can concentrate on the second term, and denote

$$\boldsymbol{\rho}'^* = \delta\boldsymbol{R}^{-1/2}\boldsymbol{B}\boldsymbol{L}^+\boldsymbol{B}^\top g(\boldsymbol{s})$$

for convenience.

First, we use Lemma 4.4.3 to show that

$$\delta\left\|\boldsymbol{R}^{-1/2}\boldsymbol{B}\boldsymbol{L}^+\left(g(\boldsymbol{s}) - \boldsymbol{\pi}^C(g(\boldsymbol{s}))\right)\right\|_\infty \leq \delta\cdot\widetilde{O}\left(\beta^{-2}\right) \leq \varepsilon/100\,.$$

Now, let's set $\boldsymbol{\pi}_{old} = \boldsymbol{\pi}^{C^0}\left(g(\boldsymbol{s}^0)\right)$, where $C^0$ was the vertex set of the sparsifier and $\boldsymbol{s}^0$ the slacks after the last call to BatchUpdate. As we will be calling BatchUpdate at least every $T$ calls to Update for some $T \geq 1$, Lemma 4.4.12 implies that

$$\delta\left\|\boldsymbol{R}^{-1/2}\boldsymbol{B}\boldsymbol{L}^+\left(\boldsymbol{\pi}^C(g(\boldsymbol{s})) - \boldsymbol{\pi}_{old}\right)\right\|_\infty \leq \delta\cdot\widetilde{O}\left(\alpha\beta^{-2}\right)T \leq \varepsilon/100\,,$$

as long as $T \leq \varepsilon\sqrt{m}/\widetilde{O}\left(\alpha\beta^{-2}\right)$.

Importantly, we will never be *removing* vertices from $C$, so $C^0 \subseteq C$. This implies that it suffices to find the large entries of

$$\delta\boldsymbol{R}^{-1/2}\boldsymbol{B}\boldsymbol{L}^+\boldsymbol{\pi}_{old}\,.$$

Now, note that for any edge $e$ that was *not* $\varepsilon/(100\alpha)$-important for $C^0$ and corresponding

resistances $\boldsymbol{r}^0$, we have

$$\begin{aligned}
&\delta \left| \boldsymbol{R}^{-1/2} \boldsymbol{B} \boldsymbol{L}^{+} \boldsymbol{\pi}_{old} \right|_e \\
&\leq \delta \sqrt{\mathcal{E}_{\boldsymbol{r}}(\boldsymbol{\pi}^{C^0}(\boldsymbol{B}^\top \frac{\mathbf{1}_e}{\sqrt{\boldsymbol{r}}}))} \sqrt{\mathcal{E}_{\boldsymbol{r}}(\boldsymbol{\pi}_{old})} \\
&\leq \delta \cdot \sqrt{\alpha} \frac{\varepsilon}{100\alpha} \cdot \sqrt{2\alpha m} \\
&= \varepsilon/50 \,,
\end{aligned}$$

where we used Lemma 4.4.5 and the fact that $\mathcal{E}_{\boldsymbol{r}}(\boldsymbol{\pi}^{C^0}(g(\boldsymbol{s}^0))) \leq 2\mathcal{E}_{\boldsymbol{r}}(g(\boldsymbol{s}^0)) \leq 2\alpha m$. Therefore it suffices to approximate

$$\delta \boldsymbol{I}_S \boldsymbol{R}^{-1/2} \boldsymbol{B} \boldsymbol{L}^{+} \boldsymbol{\pi}_{old} \,,$$

where $S$ was the set of $\frac{\varepsilon}{100\alpha}$-important edges last computed during the last call to BatchUpdate.

Now, we will use the sketching lemma from (Lemma 5.1, [73] v2), which shows that in order to find all $\Omega(\varepsilon)$ large entries of this vector, it suffices to compute the inner products

$$\delta \left\langle \boldsymbol{\pi}^C \left( \boldsymbol{B}^\top \frac{\boldsymbol{q}_S^i}{\sqrt{\boldsymbol{r}}} \right), SC^{+} \boldsymbol{\pi}_{old} \right\rangle$$

for $i \in [\widetilde{O}\left(\varepsilon^{-2}\right)]$ up to $O(\varepsilon)$ accuracy. Here $SC$ is the Schur complement onto $C$.

Based on this, there are two types of quantities that we will maintain:

- $\widetilde{O}\left(1/\varepsilon^2\right)$ approximate demand projections $\widetilde{\pi}^C \left( \boldsymbol{B}^\top \frac{\boldsymbol{q}_S^i}{\sqrt{\boldsymbol{r}}} \right)$, and
- an approximate Schur complement $\widetilde{SC}$ of $G$ onto $C$.

For the latter, we will directly use the dynamic Schur complement data structure DynamicSC that was also used by [73] and is based on [51]. For completeness, we present this data structure in Appendix 9.2.1.

For the former, we will need $\widetilde{O}\left(1/\varepsilon^2\right)$ data structures for maintaining demand projections onto $C$, under vertex insertions to $C$. The guarantees of each such a data structure, that we call an $(\alpha, \beta, \varepsilon)$-DemandProjector, are as follows.

**Definition 4.4.13** ($(\alpha, \beta, \varepsilon)$-DemandProjector)**.** *Let $\widehat{\varepsilon} \in (0, \varepsilon)$ be a tradeoff parameter. Given a graph $G(V, E)$, resistances $\boldsymbol{r}$, and a vector $\boldsymbol{q} \in [-1, 1]^m$, an $(\alpha, \beta, \varepsilon)$-DemandProjector is a data structure that maintains a vertex subset $C \subseteq V$ and an approximation to the demand projection $\boldsymbol{\pi}^C \left( \boldsymbol{B}^\top \frac{\boldsymbol{q}}{\sqrt{\boldsymbol{r}}} \right)$, with high probability under oblivious adversaries. The following operations are supported:*

- Initialize$(C, \boldsymbol{r}, \boldsymbol{q}, S, \mathcal{P})$*: Initialize the data structure in order to maintain an approximation of $\boldsymbol{\pi}^C \left( \boldsymbol{B}^\top \frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}} \right)$, where $C \subseteq V$ is a $\beta$-congestion reduction subset, $\boldsymbol{r}$ are resistances, $\boldsymbol{q} \in [-1, 1]^m$, and $S \subseteq E$ is a subset of $\gamma$-important edges. $\mathcal{P} = \{\mathcal{P}^{u,e,i} \mid u \in V, e \in E, u \in e, i \in [h]\}$ for some $h \in \mathbb{Z}_{\geq 1}$, is a set of independent random walks from $u$ to $C$ for any $u$.*
- AddTerminal$(v, \widetilde{R}_{eff}(C, v))$*: Insert $v$ into $C$. Also, $\widetilde{R}_{eff}(C, v)$ is an estimate of $R_{eff}(C, v)$ such that $\widetilde{R}_{eff}(C, v) \approx_2 R_{eff}(C, v)$. Returns an estimate*

$$\widetilde{\pi}_v^{C \cup \{v\}} \left( \boldsymbol{B}^\top \frac{\boldsymbol{q}}{\sqrt{\boldsymbol{r}}} \right)$$

*for the demand projection of* $\boldsymbol{q}$ *onto* $C \cup \{v\}$ *at coordinate* $v$ *such that*

$$\left|\widetilde{\pi}_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top \frac{\boldsymbol{q}}{\sqrt{\boldsymbol{r}}}\right) - \pi_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top \frac{\boldsymbol{q}}{\sqrt{\boldsymbol{r}}}\right)\right| \leq \frac{\widehat{\varepsilon}}{\sqrt{R_{eff}(v,C)}}\,.$$

- UPDATE$(e, \boldsymbol{r}', \boldsymbol{q}')$*: Set* $r_e = r'_e$ *and* $q_e = q'_e$*, where* $e \in E(C)$*, and* $q'_e \in [-1,1]$*. Furthermore,* $r'_e$ *satisfies the inequality* $r_e^{\max}/\alpha \leq r'_e \leq \alpha \cdot r_e^{\min}$*, where* $r_e^{\min}$ *and* $r_e^{\max}$ *represent the minimum, respectively the maximum values that the resistance of* $e$ *has had since the last call to* INITIALIZE*.*
- OUTPUT()*: Output* $\widetilde{\pi}^C\left(\boldsymbol{B}^\top \frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right)$ *such that such that after* $T \leq n^{O(1)}$ *calls to* ADDTERMINAL*, for any fixed vector* $\boldsymbol{\phi}$*,* $E_{\boldsymbol{r}}(\boldsymbol{\phi}) \leq 1$*, with high probability*

$$\left|\left\langle \widetilde{\pi}^C\left(\boldsymbol{B}^\top \frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right) - \boldsymbol{\pi}^C\left(\boldsymbol{B}^\top \frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right), \boldsymbol{\phi}\right\rangle\right| \leq \widehat{\varepsilon}\cdot\sqrt{\alpha}\cdot T\,.$$

We will implement such a data structure in Section 4.5, where we will prove the following lemma:

**Lemma 4.4.14** (Demand projection data structure)**.** *For any graph* $G(V,E)$ *and parameters* $\widehat{\varepsilon} \in (0,\varepsilon)$*,* $\beta \in (0,1)$*, there exists an* $(\alpha,\beta,\varepsilon)$-DEMANDPROJECTOR *for* $G$ *which, given access to* $h = \widetilde{\Theta}(\widehat{\varepsilon}^{-4}\beta^{-6} + \widehat{\varepsilon}^{-2}\beta^{-2}\gamma^{-2})$ *precomputed independent random walks from* $u$ *to* $C$ *for each* $e \in E$*,* $u \in e$*, has the following runtimes per operation:*

- INITIALIZE*:* $\widetilde{O}(m)$*.*
- ADDTERMINAL*:* $\widetilde{O}\left(\widehat{\varepsilon}^{-4}\beta^{-8} + \widehat{\varepsilon}^{-2}\beta^{-6}\gamma^{-2}\right)$*.*
- UPDATE*:* $O(1)$*.*
- OUTPUT*:* $O(\beta m + T)$*, where* $T$ *is the number of calls made to* ADDTERMINAL *after the last call to* INITIALIZE*.*

Now we describe the way we will use the DEMANDPROJECTORs and DYNAMICSC to get an $(\alpha,\beta,\varepsilon)$-LOCATOR $\mathcal{L}$.

**Algorithm 3** LOCATOR $\mathcal{L}$.INITIALIZE

**procedure** $\mathcal{L}$.INITIALIZE($\boldsymbol{f}$)
  $\boldsymbol{s}^+ = \boldsymbol{u} - \boldsymbol{f}$, $\boldsymbol{s}^- = \boldsymbol{f}$, $\boldsymbol{r} = \frac{1}{(\boldsymbol{s}^+)^2} + \frac{1}{(\boldsymbol{s}^-)^2}$
  $\boldsymbol{Q}$ = Sketching matrix produced by (Lemma 5.1, [73] v2)
  DYNAMICSC = DYNAMICSC.INITIALIZE($\boldsymbol{G}, \emptyset, \boldsymbol{r}, \varepsilon, \beta$)
  $C$ = DYNAMICSC.$C$ ▷ $\beta$-congestion reduction subset
  Estimate $\widetilde{R}_{eff}(C,e) \approx_4 R_{eff}(C,e)$ using Lemma 9.2.3
  $S = \left\{e \in E \mid \widetilde{R}_{eff}(C,e) \leq r_e \cdot \left(\frac{100\alpha}{\varepsilon}\right)^2\right\}$
  $h = \widetilde{\Theta}\left(\widehat{\varepsilon}^{-4}\beta^{-6} + \widehat{\varepsilon}^{-2}\varepsilon^{-2}\alpha^2\beta^{-2}\right)$
  Sample walks $\mathcal{P}^{u,e,i}$ from $u$ to $C$ for $e \in E \setminus E(C)$, $u \in e$, $i \in [h]$ (Lemma 5.15, [73] v2)
  $\mathrm{DP}^i$ = DEMANDPROJECTOR.INITIALIZE($C, \boldsymbol{r}, \boldsymbol{q}^i, S, \mathcal{P}$) for all rows $\boldsymbol{q}^i$ of $\boldsymbol{Q}$
  $\mathcal{L}$.BATCHUPDATE($\emptyset$)

**$\mathcal{L}$.INITIALIZE**: Every time $\mathcal{L}$.INITIALIZE is called, we first generate a $\beta$-congestion reduction subset $C$ based on Lemma 4.4.2 (takes time $\widetilde{O}\left(m/\beta^2\right)$), then a sketching matrix $\boldsymbol{Q}$ and its rows $\boldsymbol{q}^i$ for

$i \in \left[\widetilde{O}\left(1/\varepsilon^2\right)\right]$ as in (Lemma 5.1, [73] v2) (takes time $\widetilde{O}\left(m/\varepsilon^2\right)$), and finally random walks $\mathcal{P}^{u,e,i}$ from $u$ to $C$ for each $u \in V$, $e \in E \backslash E(C)$ with $u \in e$, and $i \in [h]$, where $h = \widetilde{O}\left(\widehat{\varepsilon}^{-4}\beta^{-6} + \widehat{\varepsilon}^{-2}\varepsilon^{-2}\alpha^2\beta^{-2}\right)$ as in (Lemma 5.15, [73] v2) (takes time $\widetilde{O}\left(h/\beta^2\right)$ for each $(u, e)$).

We also compute $\widetilde{R}_{eff}(C, u) \approx_2 R_{eff}(C, u)$ for all $u \in V$ as described in Lemma 9.2.3 so that we can let $S$ be a subset of $\varepsilon/(100\alpha)$-important edges that contains all $\varepsilon/(200\alpha)$-important edges. This takes time $\widetilde{O}(m)$. Then, we call DynamicSC.Initialize$(G, C, \boldsymbol{r}, O(\varepsilon), \beta)$ (from Appendix 9.2.1) to initialize the dynamic Schur complement onto $C$, with error tolerance $O(\varepsilon)$, which takes time $\widetilde{O}\left(m \cdot \frac{1}{\varepsilon^4\beta^4}\right)$, as well as DemandProjector.Initialize$(C, \boldsymbol{r}, \boldsymbol{q}, S, \mathcal{P})$ for the $\widetilde{O}\left(1/\varepsilon^2\right)$ DemandProjectors, i.e. one for each $\boldsymbol{q} \in \{\boldsymbol{q}^i \mid i \in [\widetilde{O}\left(1/\varepsilon^2\right)]\}$. Also, we compute

$$\boldsymbol{\pi}^{old} = \boldsymbol{\pi}^C\left(\boldsymbol{B}^\top g(\boldsymbol{s})\right),$$

which takes $\widetilde{O}(m)$ as in DemandProjector.Initialize. All of this takes $\widetilde{O}\left(m \cdot \left(\frac{1}{\widehat{\varepsilon}^4\beta^8} + \frac{\alpha^2}{\widehat{\varepsilon}^2\varepsilon^2\beta^4}\right)\right)$.

**Algorithm 4** Locator $\mathcal{L}$.Update and $\mathcal{L}$.BatchUpdate

**procedure** Update$(e = (u, w), \boldsymbol{f})$
  $s_e^+ = u_e - f_e$, $s_e^- = f_e$, $r_e = \frac{1}{(s_e^+)^2} + \frac{1}{(s_e^-)^2}$
  $\widetilde{R}_{eff}(C, u) =$ DynamicSC.AddTerminal$(u)$
  $\widetilde{R}_{eff}(C \cup \{u\}, w) =$ DynamicSC.AddTerminal$(w)$
  $C = C \cup \{u, w\}$
  **for** $i = 1, \ldots, \widetilde{O}\left(1/\varepsilon^2\right)$ **do**
    DP$^i$.AddTerminal$(u, \widetilde{R}_{eff}(C, u))$
    DP$^i$.AddTerminal$(w, \widetilde{R}_{eff}(C \cup \{u\}, w))$
  DynamicSC.Update$(e, r_e)$
  **for** $i = 1 \ldots \widetilde{O}\left(1/\varepsilon^2\right)$ **do**
    DP$^i$.Update$(e, \boldsymbol{r}, \boldsymbol{q}^i)$
**procedure** BatchUpdate$(Z, \boldsymbol{f})$
  $\boldsymbol{s}^+ = \boldsymbol{u} - \boldsymbol{f}$, $\boldsymbol{s}^- = \boldsymbol{f}$, $\boldsymbol{r} = \frac{1}{(\boldsymbol{s}^+)^2} + \frac{1}{(\boldsymbol{s}^-)^2}$
  Estimate $\widetilde{R}_{eff}(C, e) \approx_4 R_{eff}(C, e)$ using Lemma 9.2.3
  $S = \left\{e \in E \mid \widetilde{R}_{eff}(C, e) \leq r_e \cdot \left(\frac{100\alpha}{\varepsilon}\right)^2\right\}$ ▷ $\frac{\varepsilon}{100\alpha}$-important edges
  **for** $e = (u, w) \in Z$ **do**
    DynamicSC.AddTerminal$(u)$
    DynamicSC.AddTerminal$(w)$
    $C = C \cup \{u, w\}$
    DynamicSC.Update$(e, r_e)$
  **for** $i = \left[\widetilde{O}\left(1/\varepsilon^2\right)\right]$ **do**
    DP$^i$.Initialize$(C, \boldsymbol{r}, \boldsymbol{q}^i, S, \mathcal{P})$
  $\boldsymbol{\pi}_{old} = \frac{1}{\sqrt{m}} \cdot \boldsymbol{\pi}^C\left(\boldsymbol{B}^\top \frac{\frac{1}{s^+} - \frac{1}{s^-}}{\boldsymbol{r}}\right)$ ▷ Compute exactly using Laplacian solve

**$\mathcal{L}$.Update**: Now, whenever $\mathcal{L}$.Update is called on an edge $e$, either $e \in E(C)$ or $e \notin E(C)$. In the first case we simply call Update on DynamicSC and all DemandProjectors.

In the second case, we first call DynamicSC.AddTerminal on one endpoint $v$ of $e$. After

doing this we can also get an estimate $\widetilde{R}_{eff}(C,v) \approx_2 R_{eff}(C,v)$ by looking at the edges between $C$ and $v$ in the sparsified Schur complement. By the guarantees of the expander decomposition used inside DynamicSC [73], the number of expanders containing $v$, amortized over all calls to DynamicSC.AddTerminal, is $O(\text{poly}\log(n))$. As the sparsified Schur complement contains $\widetilde{O}\left(1/\varepsilon^2\right)$ neighbors of $v$ from each expander, the amortized number of neighbors of $v$ in the sparsified Schur complement is $\widetilde{O}\left(1/\varepsilon^2\right)$, and the amortized runtime to generate them (by random sampling) is $\widetilde{O}\left(1/\varepsilon^2\right)$.

Given the resistances $r_1,\ldots,r_l$ of these edges, setting $\widetilde{R}_{eff}(C,v) = \left(\sum_{i=1}^{l} r_i^{-1}\right)^{-1}$ we guarantee that $\widetilde{R}_{eff}(C,v) \approx_{1+O(\varepsilon)} R_{eff}(C,v)$, by the fact that DynamicSC maintains an $(1+O(\varepsilon))$-approximate sparsifier of the Schur complement. Then, we call AddTerminal$(v, \widetilde{R}_{eff}(C,v))$ on all DemandProjectors.

After repeating the same process for the other endpoint of $e$, we finally call Update on DynamicSC and all DemandProjectors. This takes time $\widetilde{O}\left(\frac{1}{\widehat{\varepsilon}^2\beta^2}\right)$ because of the Schur complement and amortized $\widetilde{O}\left(m\cdot\frac{\widehat{\varepsilon}\alpha^{1/2}}{\varepsilon}+\frac{1}{\widehat{\varepsilon}^4\beta^8}+\frac{\alpha^2}{\widehat{\varepsilon}^2\varepsilon^2\beta^{-6}}\right)$ for each of the demand projectors, so the total amortized runtime is $\widetilde{O}\left(m\cdot\frac{\widehat{\varepsilon}\alpha^{1/2}}{\varepsilon^3}+\frac{1}{\widehat{\varepsilon}^4\varepsilon^2\beta^8}+\frac{\alpha^2}{\widehat{\varepsilon}^2\varepsilon^4\beta^6}\right)$.

**$\mathcal{L}$.BatchUpdate**: When $\mathcal{L}$.BatchUpdate is called on a set of edges $Z$, we add them one by one in the DynamicSC data structure following the same process as in $\mathcal{L}$.Update. For the demand projectors, we first manually insert the endpoints of these edges into $C$ and then re-initialize all DemandProjectors, by calling Initialize with a new subset $S$ of $\frac{\varepsilon}{200\alpha}$-important edges that contains all $\frac{\varepsilon}{100\alpha}$-important edges. Such a set can be computed by estimating $R_{eff}(C,u)$ for all $u\in V\backslash C$ up to a constant factor and, by Lemma 9.2.3, takes time $\widetilde{O}(m)$. Also, we compute

$$\boldsymbol{\pi}^{old} = \boldsymbol{\pi}^C\left(\boldsymbol{B}^\top g(\boldsymbol{s})\right),$$

which takes $\widetilde{O}(m)$ as in DemandProjector.Initialize. The total runtime of this is $\widetilde{O}\left(m/\varepsilon^2+|Z|/(\beta^2\varepsilon^2)\right)$.

**Algorithm 5** Locator $\mathcal{L}$.Solve

```
1: procedure Solve()
2:     SC~ = DynamicSC.SC~()
3:     φ_old = SC~^+ π_old
4:     v = 0
5:     for i = 1,...,O~(1/ε^2) do
6:         π~^i = DP^i.Output()
7:         v_i = ⟨π~^i, φ_old⟩
8:     Z = Recover(v, ε/100)        ▷ Recovers all ε/2-congested edges (Lemma 5.1, [73] v2)
9:     return Z
```

**$\mathcal{L}$.Solve**: When $\mathcal{L}$.Solve is called, we set $\widetilde{SC}$ = DynamicSC.$SC()$, call Output on all DemandProjectors to obtain vectors $\widetilde{\pi}^i$ which are estimators for $\boldsymbol{\pi}^C(\boldsymbol{B}^\top\frac{\boldsymbol{g}_S^i}{\sqrt{\boldsymbol{r}}})$ in the sense of Definition 4.4.13. Then we compute $v_i = \langle\widetilde{\pi}^i, \widetilde{SC}^+\boldsymbol{\pi}_{old}\rangle$ where $\boldsymbol{\pi}_{old}$ is the demand projection that was computed exactly the last time BatchUpdate was called. These computed terms represent an approximation to the update in $(\boldsymbol{Q}\boldsymbol{\rho})_i$ between two consecutive calls of $\mathcal{L}$.Solve. As we will show in the appendix, $\langle\widetilde{\pi}^i, \widetilde{SC}^+\boldsymbol{\pi}_{old}\rangle$ is an $\varepsilon$-additive approximation of $\langle\boldsymbol{\pi}^C(\boldsymbol{B}^\top\frac{\boldsymbol{q}^i}{\sqrt{\boldsymbol{r}}}), \boldsymbol{L}^+\boldsymbol{\pi}^C(\boldsymbol{B}^\top g(\boldsymbol{s}))\rangle$ for

all $i \in \left[\widetilde{O}\left(1/\varepsilon^2\right)\right]$. The key fact that makes this approximation feasible is that although updates to the demand projection are hard to approximate with few samples, when hitting them with the deterministic vector $\boldsymbol{\pi}_{old}$, the resulting inner products strongly concentrate. The runtime of this is $\widetilde{O}\left(\beta m/\varepsilon^2\right)$.

Using these computed values with the $\ell_2$ heavy hitter data structure (Lemma 5.1, [73] v2) we get all edges with congestion more than $\varepsilon$. The total runtime is $\widetilde{O}\left(\beta m/\varepsilon^2\right)$.

## 4.5 The Demand Projection Data Structure

The main goal of this section is to construct an $(\alpha, \beta, \varepsilon)$-DemandProjector, as defined in Definition 4.4.13, and thus prove Lemma 4.4.14. The most important operation that needs to be implemented in order to prove Lemma 4.4.14 is to maintain the demand projection after inserting a vertex $v \in V \backslash C$ to $C$. In order to do this, we use the following identity from [73]:

$$\boldsymbol{\pi}^{C\cup\{v\}}\left(\boldsymbol{d}\right) = \boldsymbol{\pi}^{C}\left(\boldsymbol{d}\right) + \pi_v^{C\cup\{v\}}\left(\boldsymbol{d}\right) \cdot \left(\mathbf{1}_v - \boldsymbol{\pi}^{C}(\mathbf{1}_v)\right) , \tag{4.9}$$

where $\boldsymbol{d}$ is any demand (in our case, we have $\boldsymbol{d} = \boldsymbol{B}^\top \frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}$ for some $S \subseteq E$). For this, we need to compute approximations to $\pi_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top \frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right)$ and $\boldsymbol{\pi}^{C}\left(\mathbf{1}_v\right)$.

In Section 4.5.1, we will show that if $S$ is a subset of $\gamma$-important edges, we can efficiently estimate $\pi_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top \frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right)$ up to additive accuracy $\frac{\widehat{\varepsilon}}{R_{eff}(C,v)}$ by sampling random walks to $C$ starting only from edges with relatively high resistance. For the remaining edges, the $\gamma$-importance property will imply that we are not losing much by ignoring them.

Then, in Section 4.5.2 we will show how to approximate $\mathbf{1}_v - \boldsymbol{\pi}^{C}(\mathbf{1}_v)$. This is equivalent to estimating the hitting probabilities from $v$ to $C$. The guarantee that we would ideally like to get is on the error to route

$$\mathcal{E}_{\boldsymbol{r}}\left(\widetilde{\pi}^{C}(\mathbf{1}_v) - \boldsymbol{\pi}^{C}(\mathbf{1}_v)\right) \leq \widehat{\varepsilon}^2 R_{eff}(C, v)\,. \tag{4.10}$$

Note that this is not possible to do efficiently for general $C$. For example, suppose that the hitting distribution is uniform. In this case, $\Omega(|C|)$ random walks are required to get a bound similar to (4.10). However, it might still be possible to guarantee it by using the structure of $C$, and this would simplify some parts of our analysis. Instead, we are going to work with the following weaker approximation bound: For any fixed potential vector $\boldsymbol{\phi} \in \mathbb{R}^n$ with $E_{\boldsymbol{r}}(\boldsymbol{\phi}) \leq 1$, we have w.h.p.

$$\left|\left\langle \widetilde{\pi}^{C}(\mathbf{1}_v) - \boldsymbol{\pi}^{C}(\mathbf{1}_v), \boldsymbol{\phi}\right\rangle\right| \leq \widehat{\varepsilon}\sqrt{R_{eff}(C, v)}\,. \tag{4.11}$$

Now, using these estimation lemmas, we will bound how our demand projection degrades when inserting a new vertex into $C$. This is stated in the following lemma and proved in Appendix 9.2.5.

**Lemma 4.5.1** (Inserting a new vertex to $C$)**.** *Consider a graph $G(V, E)$ with resistances $\boldsymbol{r}$, $\boldsymbol{q} \in [-1, 1]^m$, a $\beta$-congestion reduction subset $C$, and $v \in V\backslash C$. We also suppose that we have an estimate of the $C-v$ effective resistance such that $\widetilde{R}_{eff}(C, v) \approx_2 R_{eff}(C, v)$, as well as to independent random walks $\mathcal{P}^{u,e,i}$ for each $u \in V\backslash C$, $e \in E\backslash E(C)$ with $u \in e$, $i \in [h]$, where each random walk starts from $u$ and ends at $C$.*

*If we let $S$ be a subset of $\gamma$-important edges for $\gamma > 0$, then for any error parameter $\widehat{\varepsilon} > 0$ we*

*can compute* $\widetilde{\pi}_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right)\in\mathbb{R}$ *and* $\widetilde{\pi}^{C\cup\{v\}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right)\in\mathbb{R}\in\mathbb{R}^n$ *such that with high probability*

$$\left|\widetilde{\pi}_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right)-\pi_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right)\right|\leq\frac{\widehat{\varepsilon}}{\sqrt{R_{eff}(C,v)}}\,,$$

*as long as* $h=\widetilde{\Omega}\left(\widehat{\varepsilon}^{-4}\beta^{-6}+\widehat{\varepsilon}^{-2}\beta^{-2}\gamma^{-2}\right)$*. Furthermore, for any fixed* $\boldsymbol{\phi}$*,* $E_{\boldsymbol{r}}(\boldsymbol{\phi})\leq 1$*, after* $T$ *insertions after the last call to* INITIALIZE*, with high probability*

$$\left|\left\langle\widetilde{\pi}^{C\cup\{v\}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right)-\boldsymbol{\pi}^{C\cup\{v\}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right),\boldsymbol{\phi}\right\rangle\right|\leq\widehat{\varepsilon}T\,,$$

*as long as* $h=\widetilde{\Omega}\left(\widehat{\varepsilon}^{-2}\beta^{-4}\gamma^{-2}\right)$*.*

**Algorithm 6** DEMANDPROJECTOR DP.ADDTERMINAL

**procedure** DP.ADDTERMINAL($v,\widetilde{R}_{eff}(C,v)$)
  **if** $v\in C$ **then**
    **return**
  $t=t+1$
  $\widetilde{\pi}_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right)=0$
  **for** $u,e\in S,i$ such that $\mathcal{P}^{u,e,i}\ni v$ and $\widetilde{R}_{eff}(C,v)\leq\frac{1}{\left(\min\{\widehat{\varepsilon}/\widetilde{O}(\beta^{-2}),\gamma/4\}\right)^2}r_e$ **do**
    **if** $e=(u,*)$ **then**
      $\widetilde{\pi}_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right)=\widetilde{\pi}_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right)+\frac{1}{h}\frac{q_e}{\sqrt{r_e}}$
    **else**
      $\widetilde{\pi}_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right)=\widetilde{\pi}_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right)-\frac{1}{h}\frac{q_e}{\sqrt{r_e}}$
    Shortcut $\mathcal{P}^{u,e,i}$ at $v$
  $h'=\widetilde{O}\left(\widehat{\varepsilon}^{-2}\beta^{-4}\gamma^{-2}\right)$
  $\widetilde{\pi}^C\left(\mathbf{1}_v\right)=\mathbf{0}$
  **for** $i=1,\dots h'$ **do**
    Run random walk from $v$ to $C$ with probabilities prop. to $\boldsymbol{r}^{-1}$, let $u$ be the last vertex
    $\widetilde{\pi}_u^C\left(\mathbf{1}_v\right)=\widetilde{\pi}_u^C\left(\mathbf{1}_v\right)+\frac{1}{h'}$
  $\widetilde{\pi}^{C\cup\{v\}}(\boldsymbol{B}^\top\frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}})=\widetilde{\pi}^C(\boldsymbol{B}^\top\frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}})+\widetilde{\pi}_v^{C\cup\{v\}}(\boldsymbol{B}^\top\frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}})\cdot(\mathbf{1}_v-\widetilde{\pi}^C(\mathbf{1}_v))$
  $C=C\cup\{v\}$, $F=F\backslash\{v\}$

### 4.5.1 Estimating $\pi_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right)$

There is a straightforward algorithm to estimate $\pi_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right)$. For each edge $e=(u,w)\in E\backslash E(C)$, sample a number of random walks from $u$ and $w$ until they hit $C\cup\{v\}$. Then, add to the estimate $\frac{q_e}{\sqrt{r_e}}$ times the fraction of the random walks starting from $u$ that contain $v$, minus $\frac{q_e}{\sqrt{r_e}}$ times the fraction of the random walks starting from $w$ that contain $v$. [73] uses this sampling method together with the following concentration bound, to get a good estimate if the resistances of all congested edges are sufficiently large.

**Lemma 4.5.2** (Concentration inequality 1 [73])**.** *Let* $S=X_1+\cdots+X_n$ *be the sum of* $n$ *independent random variables. The range of* $X_i$ *is* $\{0,a_i\}$ *for* $a_i\in[-M,M]$*. Let* $t,E$ *be positive numbers such*

*that* $t \leq E$ *and* $\sum_{i=1}^{n} |\mathbb{E}[X_i]| \leq E$. *Then*

$$\Pr\left[|S - \mathbb{E}[S]| > t\right] \leq 2\exp\left(-\frac{t^2}{6EM}\right).$$

Unfortunately, in our setting there is no reason to expect these resistances to be large, so the variance of this estimate might be too high. We have already introduced the concept of important edges in order to alleviate this problem, and proved that we only need to look at important edges. Even if all edges of which the demand projection is estimated are important (i.e. close to $C$), however, $v$ can still be far from $C$. This is an issue, since we don't directly estimate projections onto $C$, but instead estimate the projection onto $C \cup \{v\}$ and then from $v$ onto $C$.

Intuitively, however, if $v$ is far from $C$, it should also be far from the set of important edges, so the insertion of $v$ should not affect their demand projection too much. As the distance upper bound between an important edge and $C$ is relative to the scale of the resistance of that edge, this statement needs be more fine-grained in order to take the resistances of important edges into account.

More concretely, in the following lemma, which is proved in Appendix 9.2.5, we show that if we only compute demand projection estimates for edges $e$ such that $r_e \geq c^2 R_{eff}(C, v)$ for some appropriately chosen $c > 0$, then we can guarantee a good bound on the number of random walks we need to sample.

For the remaining edges, we will show that the energy of their contributions to the projection is negligible, so that we can reach to our desired statement in Lemma 4.5.4.

**Lemma 4.5.3.** *Consider a graph* $G(V, E)$ *with resistances* $\boldsymbol{r}$, $\boldsymbol{q} \in [-1, 1]^n$, *a* $\beta$*-congestion reduction subset* $C$, *as well as* $v \in V \backslash C$. *If for some* $c > 0$ *we are given a set of edges*

$$S' \subseteq \left\{ e \in E \backslash E(C) \mid R_{eff}(C, v) \leq \frac{1}{c^2} r_e \right\},$$

*then for any* $\delta_1' > 0$ *we can compute* $\widetilde{\pi}_v^{C \cup \{v\}} \left( \boldsymbol{B}^\top \frac{\boldsymbol{q}_{S'}}{\sqrt{\boldsymbol{r}}} \right) \in \mathbb{R}$ *such that with high probability*

$$\left| \widetilde{\pi}_v^{C \cup \{v\}} \left( \boldsymbol{B}^\top \frac{\boldsymbol{q}_{S'}}{\sqrt{\boldsymbol{r}}} \right) - \pi_v^{C \cup \{v\}} \left( \boldsymbol{B}^\top \frac{\boldsymbol{q}_{S'}}{\sqrt{\boldsymbol{r}}} \right) \right| \leq \frac{\delta_1'}{\beta c \sqrt{R_{eff}(C, v)}}.$$

*The algorithm requires access to* $\widetilde{O}\left(\delta_1'^{-2} \log n \log \frac{1}{\beta}\right)$ *independent random walks from* $u$ *to* $C$ *for each* $u \in V \backslash C$ *and* $e \in E \backslash E(C)$ *with* $u \in e$.

This leads us to the desired statement for this section, whose proof appears in Appendix 9.2.5.

**Lemma 4.5.4** (Estimating $\pi_v^{C \cup \{v\}} \left( \boldsymbol{B}^\top \frac{\boldsymbol{q}}{\sqrt{\boldsymbol{r}}} \right)$)**.** *Consider a graph* $G(V, E)$ *with resistances* $\boldsymbol{r}$, $\boldsymbol{q} \in [-1, 1]^n$, *a* $\beta$*-congestion reduction subset* $C$, *as well as* $v \in V \backslash C$. *If we are given a set* $S$ *of* $\gamma$*-important edges for some* $\gamma \in (0, 1)$ *and an estimate* $\widetilde{R}_{eff}(C, v) \approx_2 R_{eff}(C, v)$, *then for any* $\delta_1 \in (0, 1)$ *we can compute* $\widetilde{\pi}_v^{C \cup \{v\}} \left( \boldsymbol{B}^\top \frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}} \right) \in \mathbb{R}$ *such that with high probability*

$$\left| \widetilde{\pi}_v^{C \cup \{v\}} \left( \boldsymbol{B}^\top \frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}} \right) - \pi_v^{C \cup \{v\}} \left( \boldsymbol{B}^\top \frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}} \right) \right| \leq \frac{\delta_1}{\sqrt{R_{eff}(C, v)}}. \tag{4.12}$$

*The algorithm requires* $\widetilde{O}\left(\delta_1^{-4}\beta^{-6}+\delta_1^{-2}\beta^{-2}\gamma^{-2}\right)$ *independent random walks from* $u$ *to* $C$ *for each* $u\in V\backslash C$ *and* $e\in E\backslash E(C)$ *with* $u\in e$.

*Additionally, we have*

$$\left|\pi_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right)\right|\leq\frac{1}{\gamma\sqrt{R_{eff}(C,v)}}\cdot\widetilde{O}\left(\frac{1}{\beta^2}\right).$$

### 4.5.2 Estimating $\boldsymbol{\pi}^C(\mathbf{1}_v)$

In contrast to the quantity $\pi_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}}{\sqrt{\boldsymbol{r}}}\right)$, where there are cancellations between its two components $\pi_v^{C\cup\{v\}}\left(\sum_{e=(u,w)\in E}\frac{q_e}{\sqrt{r_e}}\mathbf{1}_u\right)$ and $\pi_v^{C\cup\{v\}}\left(\sum_{e=(u,w)\in E}-\frac{q_e}{\sqrt{r_e}}\mathbf{1}_w\right)$ (as $\boldsymbol{B}^\top\frac{\boldsymbol{q}}{\sqrt{\boldsymbol{r}}}$ sums up to $\mathbf{0}$), in $\boldsymbol{\pi}^C(\mathbf{1}_v)$ there are no cancellations. The goal is to simply estimate the hitting probabilities from $v$ to the vertices of $C$, which can be done by sampling a number of random walks from $v$ to $C$.

As discussed before, even though ideally we would like to have an error bound of the form $\sqrt{\mathcal{E}_{\boldsymbol{r}}(\widetilde{\pi}^C(\mathbf{1}_v)-\boldsymbol{\pi}^C(\mathbf{1}_v))}\leq\delta_2\sqrt{R_{eff}(C,v)}$, our analysis is only able to guarantee that for any fixed potential vector $\boldsymbol{\phi}$ with $E_{\boldsymbol{r}}(\boldsymbol{\phi})\leq 1$, with high probability $\left|\langle\boldsymbol{\phi},\widetilde{\pi}^C(\mathbf{1}_v)-\boldsymbol{\pi}^C(\mathbf{1}_v)\rangle\right|\leq\delta_2\sqrt{R_{eff}(C,v)}$. However, this is still sufficient for our purposes.

In Appendix 9.2.5 we prove the following general concentration inequality, which basically states that we can estimate the desired hitting probabilities as long as we have a bound on the $\ell_2$ norm of the potentials $\boldsymbol{\phi}$ weighted by the hitting probabilities.

**Lemma 4.5.5** (Concentration inequality 2)**.** *Let* $\boldsymbol{\pi}$ *be a probability distribution over* $[n]$ *and* $\widetilde{\pi}$ *an empirical distribution of* $Z$ *samples from* $\boldsymbol{\pi}$*. For any* $\bar{\boldsymbol{\phi}}\in\mathbb{R}^n$ *with* $\left\|\bar{\boldsymbol{\phi}}\right\|_{\boldsymbol{\pi},2}^2\leq\mathcal{V}$*, we have*

$$\Pr\left[\left|\langle\widetilde{\pi}-\boldsymbol{\pi},\bar{\boldsymbol{\phi}}\rangle\right|>t\right]\leq\frac{1}{n^{100}}+\widetilde{O}\left(\log\left(n\cdot\mathcal{V}/t\right)\right)\exp\left(-\frac{Zt^2}{\widetilde{O}\left(\mathcal{V}\log^2 n\right)}\right).$$

We will apply it for $\bar{\boldsymbol{\phi}}=\boldsymbol{\phi}-\phi_v\cdot\mathbf{1}$, and it is important to note that $\mathcal{E}_{\boldsymbol{r}}(\bar{\boldsymbol{\phi}})=\mathcal{E}_{\boldsymbol{r}}(\boldsymbol{\phi})$. In order to get a bound on $\left\|\bar{\boldsymbol{\phi}}\right\|_{\boldsymbol{\pi}^C(\mathbf{1}_v),2}^2$, we use the following lemma, which is proved in Appendix 9.2.5.

**Lemma 4.5.6** (Bounding the second moment of potentials)**.** *For any graph* $G$*, resistances* $\boldsymbol{r}$*, potentials* $\boldsymbol{\phi}$ *with* $E_{\boldsymbol{r}}(\boldsymbol{\phi})\leq 1$*,* $C\subseteq V$ *and* $v\in V\backslash C$ *we have* $\left\|\boldsymbol{\phi}-\phi_v\mathbf{1}\right\|_{\boldsymbol{\pi}^C(\mathbf{1}_v),2}^2\leq 8\cdot R_{eff}(C,v)$*.*

To give some intuition on this, consider the case when $V=C\cup\{v\}=\{1,\ldots,k\}\cup\{v\}$, and there are edges $e_1,\ldots,e_k$ between $C$ and $v$, one for each vertex of $C$. Then, we have $\pi_i^C(\mathbf{1}_v)=(r_{e_i})^{-1}/\sum_{i=1}^k(r_{e_i})^{-1}$, and so

$$\left\|\bar{\boldsymbol{\phi}}\right\|_{\boldsymbol{\pi}^C(\mathbf{1}_v),2}^2=\sum_{i=1}^k\frac{(\phi_i-\phi_v)^2}{r_{e_i}}\cdot\left(\sum_{i=1}^k(r_{e_i})^{-1}\right)^{-1}\leq\mathcal{E}_{\boldsymbol{r}}(\bar{\boldsymbol{\phi}})\cdot R_{eff}(C,v)\leq R_{eff}(C,v).$$

We finally arrive at the desired statement about estimating $\boldsymbol{\pi}^C(\mathbf{1}_v)$.

**Lemma 4.5.7** (Estimating $\boldsymbol{\pi}^C(\mathbf{1}_v)$)**.** *Consider a graph* $G(V,E)$ *with resistances* $\boldsymbol{r}$*, a* $\beta$*-congestion reduction subset* $C$*, as well as* $v\in V\backslash C$*. Then, for any* $\delta_2>0$*, we can compute* $\widetilde{\pi}^C(\mathbf{1}_v)\in\mathbb{R}^n$ *such*

*that with high probability*

$$\left|\langle \boldsymbol{\phi}, \widetilde{\pi}^{C}\left(\mathbf{1}_v\right) - \boldsymbol{\pi}^{C}\left(\mathbf{1}_v\right)\rangle\right| \leq \delta_2 \cdot \sqrt{R_{eff}(C,v)}\,, \tag{4.13}$$

*where $\boldsymbol{\phi} \in \mathbb{R}^n$ is a fixed vector with $E_{\boldsymbol{r}}(\boldsymbol{\phi}) \leq 1$. The algorithm computes $\widetilde{O}\left(\frac{\log n}{\delta_2^2}\right)$ random walks from $v$ to $C$.*

*Proof.* Because both $\widetilde{\pi}^{C}(\mathbf{1}_v)$ and $\boldsymbol{\pi}^{C}(\mathbf{1}_v)$ are probability distributions, the quantity (4.13) doesn't change when a multiple of $\mathbf{1}$ is added to $\boldsymbol{\phi}$, and so we can replace it by $\bar{\boldsymbol{\phi}} = \boldsymbol{\phi} - \phi_v \mathbf{1}$.

Now, $\widetilde{\pi}^{C}\left(\mathbf{1}_v\right)$ will be defined as the empirical hitting distribution that results from sampling $Z$ random walks from $v$ to $C$. Directly applying the concentration bound in Lemma 4.5.5 and setting $Z = \widetilde{O}\left(\frac{\log n}{\delta_2^2}\right)$, together with the fact that $\left\|\bar{\boldsymbol{\phi}}\right\|^2_{\boldsymbol{\pi}^{C}(\mathbf{1}_v),2} \leq 8 \cdot R_{eff}(C,v)$ by Lemma 4.5.6 and $\log\log R_{eff}(C,v) \leq O(\log\log n)$, we get

$$\Pr\left[\left|\langle \widetilde{\pi}^{C}(\mathbf{1}_v) - \boldsymbol{\pi}^{C}(\mathbf{1}_v), \bar{\boldsymbol{\phi}}\rangle\right| > \delta_2 \cdot \sqrt{R_{eff}(C,v)}\right] < \frac{1}{n^{10}}\,.$$

□

### 4.5.3 Proof of Lemma 4.4.14

We are now ready for the proof of Lemma 4.4.14.

*Proof of Lemma 4.4.14.* Let DP be a demand projection data structure. We analyze its operations one by one.

**Algorithm 7** DemandProjector DP.Initialize

```
1: procedure DP.Initialize(C, r, q, S, P)
2:     Initialize C, r, q, S, P
3:     F = V \ C
4:     h = Õ(ε̂^{-4} β^{-6} + ε̂^{-2} β^{-4} γ^{-2})      ▷ #random walks for each pair u ∈ V, e ∈ E with u ∈ e
5:     t = 0                                              ▷ #calls to AddTerminal since last call to UpdateFull
6:     φ = L_FF^+ [B^⊤ q_S / √r]_F
7:     π̃^C (B^⊤ q_S / √r) = [B^⊤ q_S / √r]_C − L_CF φ
```

**DP.Initialize**$(C, \boldsymbol{r}, \boldsymbol{q}, S, \mathcal{P})$**:** We initialize the values of $C, \boldsymbol{r}, \boldsymbol{q}, S, \mathcal{P}$. Then we exactly compute the demand projection, i.e. $\widetilde{\pi}^{C}\left(\boldsymbol{B}^\top \frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right) = \boldsymbol{\pi}^{C}\left(\boldsymbol{B}^\top \frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right)$, which takes time $\widetilde{O}\left(m\right)$ as shown in [73]. More specifically, we have $\boldsymbol{\pi}^{C}\left(\boldsymbol{B}^\top \frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right) = \begin{pmatrix}\boldsymbol{I} & \boldsymbol{L}_{CF}\boldsymbol{L}_{FF}^{-1}\end{pmatrix}\boldsymbol{B}^\top \frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}$ which only requires applying the operators $\boldsymbol{L}_{FF}^{-1}$ and $\boldsymbol{L}_{CF}$.

**DP.AddTerminal**$(v,\ \widetilde{R}_{eff}(C,v))$**:** We will serve this operation by applying Lemma 4.5.1. It is important to note that the error guarantee for the Output procedure increases with every call to AddTerminal, so in general we have a bounded budget for the number of calls to thus procedure before having to call again Initialize.

We apply Lemma 4.5.1 to obtain $\widetilde{\pi}_v^{C\cup\{v\}}(\boldsymbol{B}^\top \frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}})$, and update the estimate $\widetilde{\pi}_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top \frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right)$. The former can be achieved with $h = \widetilde{O}\left(\widehat{\varepsilon}^{-4}\beta^{-6} + \widehat{\varepsilon}^{-2}\beta^{-2}\gamma^{-2}\right)$ random walks. Note that these random

walks are already stored in $\mathcal{P}$, so accessing each of them takes time $\widetilde{O}(1)$. Using the congestion reduction property of $C$, we see that the running time of the procedure, which is dominated by shortcutting the random walks is $\widetilde{O}\left(h\beta^{-2}\right)$, which gives the claimed bound. The latter can be achieved with $h' = \widetilde{O}\left(\widehat{\varepsilon}^{-2}\beta^{-4}\gamma^{-2}\right)$ fresh random walks. Due to the congestion reduction property, simulating each of these requires $\widetilde{O}\left(\beta^{-2}\right)$ time.

**Algorithm 8** DemandProjector DP.Update and DP.Output

1: **procedure** DP.Update($e, \boldsymbol{r}', \boldsymbol{q}'$)
2: **if** $e \in S$ **then**
3: $\widetilde{\pi}^C\left(\boldsymbol{B}^\top \frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right) = \widetilde{\pi}^C\left(\boldsymbol{B}^\top \frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right) + \left(\frac{q'_e}{\sqrt{r'_e}} - \frac{q_e}{\sqrt{r_e}}\right) \cdot \boldsymbol{B}^\top \mathbf{1}_e$
4: $q_e = q'_e$, $r_e = r'_e$
5: **procedure** DP.Output()
6: **return** $\widetilde{\pi}^C\left(\boldsymbol{B}^\top \frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right)$

**DP.Update$(e, \boldsymbol{r}', \boldsymbol{q}')$:** We update the values of $r_e, q_e$. We also update the projection, by noting that since $e \in E(C)$,

$$\boldsymbol{\pi}^C\left(\boldsymbol{B}^\top \frac{\boldsymbol{q}'}{\sqrt{\boldsymbol{r}'}}\right) = \boldsymbol{\pi}^C\left(\boldsymbol{B}^\top \frac{\boldsymbol{q}}{\sqrt{\boldsymbol{r}}}\right) + \left(\frac{q'_e}{\sqrt{r'_e}} - \frac{q_e}{\sqrt{r_e}}\right) \boldsymbol{B}^\top \mathbf{1}_e\,,$$

so we change $\widetilde{\pi}^C\left(\boldsymbol{B}^\top \frac{\boldsymbol{q}}{\sqrt{\boldsymbol{r}}}\right)$ by the same amount, which takes time $O(1)$ and does not introduce any additional error in our estimate.

**DP.Output():** We output our estimate $\widetilde{\pi}^C\left(\boldsymbol{B}^\top \frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right)$. Per Lemma 4.5.1 we see that each of the previous $T$ calls to AddTerminal add an error to our estimate of at most $\widehat{\varepsilon}$ in the sense that if $\boldsymbol{\Delta}^t$ were the true change in the demand projection at the $t^{th}$ insertion, and $\widetilde{\boldsymbol{\Delta}}^t$ were the update made to our estimate, then

$$\left|\left\langle \widetilde{\boldsymbol{\Delta}}^t - \boldsymbol{\Delta}^t, \boldsymbol{\phi} \right\rangle\right| \leq \widehat{\varepsilon},$$

w.h.p. for any fixed $\boldsymbol{\phi}$ such that $E_{\boldsymbol{r}^t}(\boldsymbol{\phi}) \leq 1$, where $\boldsymbol{r}^t$ represents the resistances when $t^{th}$ call to AddTerminal is made. Equivalently, for any nonzero $\boldsymbol{\phi}$,

$$\frac{1}{\sqrt{E_{\boldsymbol{r}^t}(\boldsymbol{\phi})}}\left|\left\langle \widetilde{\boldsymbol{\Delta}}^t - \boldsymbol{\Delta}^t, \boldsymbol{\phi} \right\rangle\right| \leq \widehat{\varepsilon},$$

By the invariant satisfied by the resistances passed as parameters to the AddTerminal routine, we have that $\boldsymbol{r}^t \leq \alpha \cdot \boldsymbol{r}^T$ for all $t$. Therefore $1/E_{\boldsymbol{r}^T}(\boldsymbol{\phi}) \leq \alpha/E_{\boldsymbol{r}^t}(\boldsymbol{\phi})$. So we have that

$$\frac{1}{\sqrt{E_{\boldsymbol{r}^T}(\boldsymbol{\phi})}}\left|\left\langle \widetilde{\boldsymbol{\Delta}}^t - \boldsymbol{\Delta}^t, \boldsymbol{\phi} \right\rangle\right| \leq \widehat{\varepsilon} \cdot \sqrt{\alpha}\,.$$

Summing up over $T$ insertions, we obtain the desired error bound. Furthermore, note that returning the estimate takes time proportional to $|C|$, which is $\widetilde{O}(\beta m + T)$.

□

# Chapter 5

# Decomposable Submodular Function Minimization via Maximum Flow

## 5.1 Introduction

A significant amount of work has been dedicated to the study of submodular functions. While this topic has garnered a lot of excitement from the theory community due to its the multiple connections to diverse algorithmic areas [116, 78], on the practical side minimizing submodular functions has been intensively used to model discrete problems in machine learning. MAP inference in Markov Random Fields [99], image segmentation [8, 147], clustering [126], corpus extraction problems [110] are just a few success stories of submodular minimization.

Polynomial time algorithms for this problem have been known ever since the 80's [78], and they have seen major running time improvements in more recent years [143, 84, 61, 132, 107, 35]. However, the massive scale of the problems that use submodular minimization nowadays drives the need for further developments.

One great advantage offered by the submodular functions that occur in practice is that they are *structured*. For example, in many common cases (hypergraph cuts [165], covering functions [158], MAP inference [60, 99, 166]) these can be decomposed into sums of simple submodular functions defined on small subsets. For these instances, prior work [90, 131, 56, 57, 94] has focused on providing efficient algorithms in the regime where the functions in the decomposition admit fast optimization oracles.

Notably, many of these recent developments have leveraged a mix of ideas coming from both discrete and continuous optimization. In particular, Ene et al. [57] present algorithms for decomposable function minimization that are based on both continuous methods (i.e. gradient descent) and discrete algorithms, as the authors employ a version of the preflow-push algorithm for maximum flow [76]. As this work was paralleled by multiple improvements to the running time for maximum flow [118, 117, 114, 95, 25, 73], most of which stemmed from innovations in convex optimization, it seemed plausible that the same new optimization techniques could be helpful for improving the running times of other fundamental problems in combinatorial optimization, including submodular function minimization. In this context, a particularly intriguing question emerged:

*Can we leverage the techniques used to obtain faster algorithms for maximum flow to provide faster algorithms for submodular function minimization?*

We answer this question in the affirmative, by showing how to solve decomposable submodular function minimization using *black-box access* to any routine that can compute the maximum flow in

a capacitated directed graph. To compare the running times, in the case where all the functions in the decomposition act on $O(1)$ elements of the ground set and are polynomially bounded (such as the case of a hypergraph cut function, with $O(1)$ sized hyperedges), our algorithm has – up to polylogarithmic factors – the same running time as that of computing maximum flow in a sparse graph with $O(|V|)$ vertices, and polynomial integral capacities [75, 25, 73].

As it turns out, to achieve this it is not sufficient to directly use off-the-shelf maximum flow algorithms. Instead, our approach is based on solving submodular minimization in the more general parametric setting, where we further parametrize the problem with an additional time-dependent penalty term on the elements in the set, and want to simultaneously solve *all* the problems in this family. In turn, our reduction requires solving the parametric minimum cut problem, which has been intensely studied in the classical graph theoretic literature [72, 122, 160, 77]. In this setting, which is essentially a particular case of parametric submodular minimization, the capacities of certain arcs in the graph evolve in a monotonic fashion.

While some of the existing work on parametric cuts and flows does provide efficient algorithms via reductions to maximum flow [160], the type of parametric capacities it supports does not cover the requirements for our more general scenario. Therefore, we develop a new efficient algorithm for computing parametric cuts under a broad range of parametric capacities. Our algorithm is nearly optimal from the perspective of weakly-polynomial time algorithms, since its running time matches (up to polylogarithmic factors involving certain parameters) that of the fastest maximum flow algorithm in a directed graph with integer capacities. In addition, our reduction also provides novel improvements in several other regimes, involving the strongly polynomial case, and that of planar graphs, both of which may be of independent interest.

### 5.1.1 Our Results

In this chapter we establish further connections between discrete and continuous optimization to provide an efficient algorithm for solving the decomposable submodular function minimization problem in the more general parametric setting. Our algorithm is at its core based on a continuous optimization method, but whose progress steps are driven by a new combinatorial algorithm we devise for the parametric cut problem. In this sense, our approach leverages the paradigm of combinatorial preconditioning from scientific computing literature [156, 17, 102, 162].

To properly state our main result, we need to introduce some notation. Let $V = \{1, \ldots, n\}$ and let $F : 2^V \to \mathbb{N}$ a submodular set function with the special property that

$$F(S) = \sum_{i=1}^{r} F_i(S), \quad \text{for all } S \subseteq V\,,$$

where each $F_i : 2^V \to \mathbb{N}$ is a submodular set function acting on a subset $V_i \subseteq V$ of elements, in the sense that $F_i(S) = F_i(S \cap V_i)$ for all $S \subseteq V$. Let $\mathrm{EO}_i$ be the time required to evaluate $F_i(S)$ for any $S \subseteq V_i$, and let $\mathcal{O}_i$ be the time required to minimize $F_i(S) + w(S)$ over $V_i$, where $w$ is any linear function, and suppose that $\max_{S \subseteq V} F(S) = n^{O(1)}$. Furthermore, for each $i \in V$ let $\psi_i : \mathbb{R} \to \mathbb{R}$ be a strictly convex function satisfying $n^{-O(1)} \leq |\psi_i''(x)| \leq n^{O(1)}$ and $|\psi_i'(0)| \leq n^{O(1)}$.

Then our main theorem is the following.

**Theorem 5.1.1.** *There is an algorithm which for all $\lambda \in \mathbb{R}$ simultaneously optimizes the objective*

$$\min_{S \subseteq V} F(S) + \sum_{i \in S} \psi_i'(\lambda)$$

*by returning a vector $x$ such that for any $\lambda \in \mathbb{R}$ the set $S^\lambda = \{u \ : \ x_u \geq \lambda\}$ satisfies*

$$F(S^\lambda) + \sum_{u \in S^\lambda} \psi'(\lambda) \leq \min_{S \subseteq V} F(S) + \sum_{u \in S} \psi'(\lambda) + \varepsilon \, .$$

*Furthermore, if $T_{\text{maxflow}}(n, m)$ is the time required to compute the maximum flow in a directed graph with polynomially bounded integral capacities, then our algorithm runs in time*

$$\widetilde{O}\left( \max_i |V_i|^2 \left( \sum_{i=1}^{r} |V_i|^2 \mathcal{O}_i + T_{\text{maxflow}}\left(n, n + \sum_{i=1}^{r} |V_i|^2\right)\right) \log \frac{1}{\varepsilon}\right) .$$

To better understand this result, let us consider the case where each submodular function in the decomposition acts on a small number of elements, i.e. $|V_i| = O(1)$. In this case we have the following corollary:

**Corollary 5.1.1.** *If each function $F_i$ in the decomposition acts on $O(1)$ elements of the ground set, then we can the solve parametric submodular minimization problem to $\varepsilon$ precision in time*

$$\widetilde{O}\left(T_{\text{maxflow}}(n, n + r) \log \frac{1}{\varepsilon}\right) .$$

While our statements concern the parametric setting, it is easy to use them to recover the solution to the standard submodular minimization problem. Simply by letting $\psi_i'(t) = t$ for all $i$, and thresholding the returned vector at 0 we obtain the desired result. Using Goldberg-Rao [75] or the current state of the art maximum flow advancements [25, 73, 37], we see that this significantly improves over all the previous algorithms for decomposable submodular minimization, in the regime where all sets $V_i$ are small. Following [57] it has remained widely open whether algorithms improving upon the $\widetilde{O}(\min\{n^2, nr\} \log^{O(1)}(1/\varepsilon))$ running time exist, and it has been conjectured that faster running times could be obtained by leveraging the newer techniques for graph algorithms based on interior point methods.

Using [37], we obtain a running time of $O\left((n + r)^{1+o(1)} \log 1/\varepsilon\right)$.

The crucial subroutine our algorithm is based on is a novel efficient algorithm for solving the parametric cut problem using a maximum flow oracle. We give an overview of our reduction and its additional applications in Section 5.3, and describe it in detail in Appendix 9.3.3.

### 5.1.2 Previous Work

**Related Works on Submodular Minimization** Submodular function minimization is a classical problem in combinatorial optimization, which goes back to the seminal work of Edmonds [54]. The first polynomial-time algorithm was obtained by Grötschel et al. [78] using the ellipsoid method. This was followed by a plethora of improvements, among which the more recent ones [43, 91] leveraged related techniques. On a different front, there has been significant work dedicated to obtaining strongly polynomial time algorithms for this problem [61, 84, 85, 132, 143, 107, 43, 91].

For the more structured regime of *decomposable submodular function minimization*, algorithms based on both discrete and continuous methods have been developed. Kolmogorov [100] has shown that this problem reduces to computing maximum submodular flows, and gave an algorithm for this problem based on augmenting paths. This was followed by further algorithms based on discrete methods [8, 60]. The continuous methods are based on convex optimization on the submodular base polytope, which is also used here. Notably, Stobbe and Krause [158] tackled this problem using gradient descent, Nishihara et al. [131] used alternating projections to obtain an algorithm with

linear convergence, Ene and Nguyen [56] achieved an improved algorithm with a linear convergence rate based on accelerated coordinate descent, while Ene et al. provided further improvements both via gradient descent and combinatorial techniques [57].

**Related Works on Parametric Min Cut** The seminal work of Gallo et al. [72] studied the generalization of the maximum flow problem where some edge-capacities, instead of being fixed, are allowed to be (possibly different) monotonic functions of a single parameter. They showed how to modify certain versions of the push-relabel algorithm for ordinary maximum flow to the parametric problem with the same asymptotic time complexity. In particular, using the Goldberg-Tarjan max-flow algorithm [76] they gave an $O(nm \log(n^2/m))$ time bound for the parametric version. Their algorithm can compute the min cuts either when a set of parameter values are given [79] or the capacity functions are all affine functions of the parameter $\lambda$.

Several other max-flow algorithms were also shown to fit into their framework (see e.g., [77]) though all requiring $\Omega(mn)$ time in the worst case. Further generalizations of the parametric min-cut problems have also been considered [122, 77]. When all parameterized capacities are equal to the parameter $\lambda$, Tarjan et al. [160] give a divide and conquer approach that can use any maximum flow algorithm as a black box and is a factor $\min\{n, \log(nU)\}$ worse.

## 5.2 Background and Preliminaries

### 5.2.1 Submodular Set Functions and Convex Analysis

Let $V$ be a finite ground set of size $n$, and we assume w.l.o.g. that $V = \{1, \ldots, n\}$. A set function $F : 2^V \to \mathbb{R}$ is submodular if $F(A) + F(B) \geq F(A \cup B) + F(A \cap B)$ for any two sets $A, B \subseteq V$. We are concerned with minimizing submodular set functions of the form $F = \sum_{i=1}^{r} F_i$, where each $F_i$ is a submodular set function:

$$\min_{A \subseteq V} F(A) = \min_{A \subseteq V} \sum_{i=1}^{r} F_i(A) \ .$$

For the rest of the chapter we will assume that $F_i$ are non-negative, integral, and that $\max_{S \subseteq V} F(S) \leq F_{\max}$. The non-negativity constraint holds without loss of generality, as we can simply shift each $F_i$ by a constant until it becomes non-negative. For rational functions that are represented on bounded bit precision, the integrality can be enforced simply by scaling them, at the expense of increasing $F_{\max}$. As we will see, some of our subroutines depend on the magnitude of $F_{\max}$, so we will generally assume that this is polynomially bounded.

As in previous works [131, 56, 57, 94], in this chapter we are concerned with the regime where each function $F_i$ in the decomposition acts on few elements of the ground set $V$. More precisely for each $i \in \{1, \ldots, r\}$ there is a *small* set $V_i \subseteq V$ such that $F_i(A) = F_i(A \cap V_i)$ for all $A \subseteq V$. We assume w.l.o.g. that $F_i(\emptyset) = F_i(V_i)$, which we discuss in more detail in Section 9.3.4. The running time of our algorithm depends on $\max_{1 \leq i \leq r} |V_i|$. This assumption is important as, furthermore, the final running time of our algorithm depends on (i) the time $\mathcal{O}_i$ to optimize functions of the form $F_i(S) + w(S)$ over $V_i$, where $w$ is a linear function and (ii) the time $\mathrm{EO}_i$ to evaluate $F_i$ for subsets of $V_i$. In the case where $|V_i| = O(1)$, this is also constant time.

Given an arbitrary vector $w \in \mathbb{R}^n$ and a subset $A \subseteq V$, we use the notation $w(A) = \sum_{i \in A} w_i$.

**Definition 5.2.1.** *Given a submodular set function $F : 2^V \to \mathbb{R}$, such that $F(\emptyset) = 0$, its submodular*

*base polytope $B(F)$ is defined as follows:*

$$B(F) = \{w \in \mathbb{R}^n : w(A) \leq F(A) \text{ for all } A \subseteq V,\, w(V) = F(V)\} .$$

**Definition 5.2.2.** *Given a submodular set function $F : 2^V \to \mathbb{R}$, $F(\emptyset) = 0$, its Lovász extension $f : \mathbb{R}^n \to \mathbb{R}$ is defined over $[0,1]^n$ as the convex closure of $F$. However, it will be more convenient to consider its extension of $\mathbb{R}^n$, given by*

$$f(x) = \int_0^\infty F(\{i : x_i \geq t\})\, dt + \int_{-\infty}^0 (F(\{i : x_i \geq t\}) - F(V))\, dt .$$

**Fact 5.2.1.** *It is well known [15] that the Lovász extension of a submodular set function $F$ can be equivalently characterized in terms of its submodular base polytope $B(F)$. More precisely, if $F(\emptyset) = 0$, then:*

$$f(x) = \max_{w \in B(F)} \langle w, x \rangle .$$

For parametric submodular function minimization we consider a family of functions parameterized by $\alpha \in \mathbb{R}$:

$$F_\alpha(A) = F(A) + \sum_{i \in A} \psi_i'(\alpha) , \tag{5.1}$$

where $\psi_j : \mathbb{R} \to \mathbb{R}$ are strictly convex differentiable functions, satisfying $\lim_{\alpha \to -\infty} \psi_i'(\alpha) = -\infty$ and $\lim_{\alpha \to \infty} \psi_i'(\alpha) = \infty$, for all $i$. A common example is $\psi_j'(\alpha) = \alpha$, which imposes an $\ell_1$ penalty on the size of the set $A$. It is shown in [36, 15] that minimizing $F_\alpha(A)$ for the entire range of scalars $\alpha$ amounts to minimizing a regularized version of the Lovász extension.

**Lemma 5.2.3.** *Let $F_\alpha$ be the family of parameterized submodular set functions defined as in (5.1), where $\psi_i$ are strictly convex functions. Let $f$ be the Lovász extension of $F$, and consider the optimization problem*

$$\min_{x \in \mathbb{R}^n} f(x) + \sum_{i \in V} \psi_i(x_i) . \tag{5.2}$$

*Let $A^\alpha = \arg\min_{A \subseteq V} F_\alpha(A)$, and let $x^*$ be the minimizer of (5.2). Then*

$$A^\alpha = \{i : x_i^* \geq \alpha\} . \tag{5.3}$$

For completeness we reproduce the proof of Lemma 5.2.3 in Section 9.3.2.

Via convex duality one can prove that minimizing (5.2) is equivalent to a dual optimization problem on the submodular base polytope $B(F)$:

$$\min_{w \in B(F)} \sum_{i \in V} \psi_i^*(-w_i) , \tag{5.4}$$

where $\psi_i^*$ is the *Fenchel dual* of $\psi_i$.

**Definition 5.2.4** (Fenchel dual)**.** *Let $g : \mathbb{R}^n \to \mathbb{R} \cup \{-\infty, +\infty\}$ be a convex function. Its Fenchel dual or convex conjugate $g^* : \mathbb{R}^n \to \mathbb{R} \cup \{-\infty, +\infty\}$ is defined as*

$$g^*(w) = \sup_x \langle w, x \rangle - g(x) .$$

We will refer to (5.2) as the primal problem and (5.4) as the dual problem. The algorithm described in this chapter will focus on optimizing (5.4) while strongly leveraging the decomposable

structure of $F$. We assume that all functions $\psi_i$ have "nice" second derivatives, which will play an important role in the algorithm, since this will also ensure that the minimizers of (5.2) and (5.4) are unique.

**Assumption 5.2.2.** *The function $\psi_i$ is $L$-smooth and $\sigma$-strongly convex for all $i \in V$. Equivalently for each $i$, its second derivative satisfies $0 < \sigma \leq \psi_i''(x) \leq L$, for all $x \in \mathbb{R}$. Furthermore, $|\psi_i'(0)| \leq n^{O(1)}$, for all $i \in V$.*

This condition also helps us ensure that we can efficiently convert between the primal and dual spaces onto which the $\psi_i$ and its Fenchel dual $\psi_i^*$ act. Also, whenever it is convenient, we will use the notation $\psi(x) = \sum_{i \in V} \psi_i(x_i)$, $\psi^*(y) = \sum_{i \in V} \psi_i^*(y_i)$.

### 5.2.2 Overview of Approach

**Decomposable Submodular Minimization** Here we provide an overview of our approach for minimizing decomposable submodular functions. Our approach for the parametric setting yields a strictly stronger result without sacrificing running time, so we will focus on this more general problem.

Our approach is based on minimizing a convex function on the submodular base polytope $B(F)$. As it has been seen in previous works [15], in order to solve the parametric problem (5.1), it suffices to solve the dual problem (5.4), which is a convex optimization problem over $B(F)$. For convenience let us denote by $h(w) = \sum_{i \in V} \psi_i^*(-w_i)$, so that our objective becomes computing $\min_{w \in B(F)} h(w)$.

We use an iterative method, which maintains a point $w \in B(F)$ and updates it in such a way that the objective value improves significantly in each step. To do so, we find a polytope $P$ such that

$$w + \frac{1}{\alpha} \cdot P \subseteq B(F) \subseteq w + P \tag{5.5}$$

and such that we can efficiently minimize $\psi$ over $w + \frac{1}{\alpha} \cdot P$. If we can find the minimizer $w'$ over $w + \frac{1}{\alpha} \cdot P$, then moving our iterate to $w'$ also guarantees that

$$h(w') - h(w^*) \leq \left(1 - \frac{1}{\alpha}\right) (h(w) - h(w^*)) \ ,$$

where $w^*$ is the minimizer of $h$ over $B(F)$. This is true due to the convexity of $h$. Indeed, let $\widetilde{w} = w + t(w^* - w)$ where $t = \max\{t \leq 1 : w + t(w^* - w) \in w + \frac{1}{\alpha} P\}$; in other words $\widetilde{w}$ represents the furthest point on the segment connecting $w$ and $w^*$ such that $\widetilde{w}$ still lies inside the small polytope $w + \frac{1}{\alpha} P$. Due to the sandwiching property of the polytopes (5.5), we have that $t \geq 1/\alpha$. Hence, using the convexity of $h$, we obtain that

$$h(\widetilde{w}) - h(w^*) = h(w + t(w^* - w)) - h(w^*) \leq (1 - t)\,(h(w) - h(w^*)) \leq (1 - 1/\alpha)(h(w) - h(w^*))\,.$$

Since $w'$ minimizes $h$ over $w + \frac{1}{\alpha} \cdot B(F)$, we must have $h(w') \leq h(\widetilde{w})$, and we obtain the desired progress in function value. Thus iterating $\widetilde{O}(\alpha)$ times we obtain a high precision solution, which we then convert back to a combinatorial solution to the original problem using some careful error analysis.

More importantly, we need to address the question of finding a polytope $P$ satisfying (5.5).

To do so, for each $i$, we define the "residual" submodular functions $F_i'(A) = F_i(A) - w_i(A)$ for all $A \subseteq V$, where $w_i \in B(F_i)$ such that $\sum_{i=1}^r w_i = w$. The existence of such a decomposition of $w \in B(\sum_{i=1}^r F_i)$ is well-known, and goes back to Edmonds [53]. Very importantly, we note that since $F_i$ were non-negative, $F_i'$ remain non-negative submodular set functions.

It is known [45] that non-negative submodular set functions can be approximated by graph cuts. Following the proof from [45], for each $i$ we construct a graph on $O(|V_i|)$ vertices whose cuts approximate the value of $F_i'$ within a factor of $O(|V_i|^2)$. Combining all these graphs into a single one, we obtain a weighted directed graph on $O(|V|)$ vertices and $O(|V| + \sum_{i=1}^{r} |V_i|^2)$ arcs such that its cut function $G$ approximates $F'$ within a factor of $O(\max_i |V_i|^2)$.

Crucially, we can show that if $G$ is the cut function which approximates $F'$, then we also have that

$$\frac{1}{\max_i |V_i|^2} \cdot B(G) \subseteq B(F_i') \subseteq B(G)\,,$$

and therefore it suffices to implement a routine that minimizes $h$ over $w + \frac{1}{\max_i |V_i|^2} \cdot B(G)$ in order to obtain an algorithm that terminates in $\widetilde{O}(\max_i |V_i|^2)$ such iterations.

To implement this routine, we devise a new combinatorial algorithm for solving the parametric flow problem, with general parameterized capacities. By comparison to previous literature, our algorithm efficiently leverages a maximum flow oracle on a sequence of graphs obtained via contracting edges, and whose running time is up to polylogarithmic factors equal to that of computing a maximum flow in a capacitated directed graph with $O(|V|)$ vertices and $O(|V| + \sum_{i=1}^{r} |V_i|^2)$ arcs.

Following this, we convert the combinatorial solution to the parametric flow problem into a solution to its corresponding dual problem on the submodular base polytope, which returns the new iterate $w'$.

Throughout the algorithm we need to control the errors introduced by the fact that both the solution we receive for the parametric flow problem and the one we return as an approximate minimizer of $h$ over $B(F)$ are *approximate*, but these are easily tolerable since our main routines return high precision solutions.

## 5.3 Parametric Min $s, t$-Cut

In the general parametric min $s, t$-cut problem [72], the capacities of the source's outgoing edges $(s, v)$ are (possibly different) nonnegative real nondecreasing functions of a parameter $\lambda \in \mathbb{D}$, where $\mathbb{D} \subseteq \mathbb{R}$ is some domain, whereas the capacities of the sink's incoming edges $vt$ are nonincreasing functions of $\lambda$. The goal is to compute the representation of the *cut function* $\kappa : \mathbb{D} \to \mathbb{R}$ such that $\kappa(\lambda)$ equals the capacity of the minimum $s, t$-cut in $G_\lambda$ obtained from $G$ by evaluating the parameterized capacity functions at $\lambda$. It is known that $\kappa$ consists of $O(n)$ pieces, where $\kappa$ equals the parameterized capacity of some fixed cut in $G$.

More formally, let $\lambda_{\min} \in \mathbb{D}$ be such that the minimal min $s, t$-cuts of $G_{\lambda_{\min}}$ and $G_{\lambda'}$ are equal for all $\lambda' \in \mathbb{D}$, $\lambda' < \lambda_{\min}$. Similarly, let $\lambda_{\max} \in \mathbb{D}$ be such that the minimal min $s, t$-cuts of $G_{\lambda_{\max}}$ and $G_{\lambda'}$ are equal for all $\lambda' \in \mathbb{D}$ with $\lambda' > \lambda_{\max}$. We will consider $\lambda_{\min}$ and $\lambda_{\max}$ inputs to our problem. Then, there exist $O(n)$ *breakpoints* $\Lambda = \{\lambda_1, \ldots, \lambda_k\}$, $\lambda_{\min} = \lambda_0 < \lambda_1 < \ldots < \lambda_k$ and an embedding of vertices $\tau : V \to \Lambda \cup \{\lambda_{\min}, \infty\}$ such that for all $i = 0, \ldots, k-1$, $\lambda' \in [\lambda_i, \lambda_{i+1}) \cap \mathbb{D}$, $\kappa(\lambda')$ equals the capacity of the cut $S(\lambda_i) = \{v \in V : \tau(v) \leq \lambda_i\}$ in $G_{\lambda'}$, and also $S(\lambda_k)$ is a min $s, t$-cut of $G_{\lambda_{\max}}$.

Motivated by our submodular minimization application, our algorithm in the most general setting solves the $\varepsilon$-approximate parametric min $s, t$-cut problem.

**Definition 5.3.1** ($\varepsilon$-approximate parametric min $s, t$-cut)**.** *Let $\Lambda$, $\tau$, and $S : \mathbb{D} \to 2^V$ be as defined above. A pair $(\Lambda, \tau)$ is called an $\varepsilon$-approximate parametric min $s, t$-cut of $G$ if:*

1. *For $i = 0, \ldots, k-1$, $S(\lambda_i)$ is a min $s, t$-cut of $G_{\lambda'}$ for all $\lambda' \in [\lambda_i, \lambda_{i+1} - \varepsilon) \cap \mathbb{D}$.*

2. *$S(\lambda_k)$ is a min $s,t$-cut of $G_{\lambda_{\max}}$.*

3. *For $i = 0, \ldots, k-1$, $S(\lambda_i) \subsetneq S(\lambda_{i+1})$.*

We prove that an $\varepsilon$-approximate parametric min $s,t$-cut yields breakpoints within $\varepsilon$ additive error wrt. to the breakpoints of the exact parametric min $s,t$-cut. Our algorithm solves the above problem assuming only constant-time black-box access to the capacity functions.

**Theorem 5.3.2.** *Let $R = \lambda_{\max} - \lambda_{\min}$ be an integral multiple of $\varepsilon > 0$. Let $T_{\text{maxflow}}(n', m') = \Omega(m' + n')$ be a convex function bounding the time needed to compute maximum flow in a graph with $n'$ vertices and $m'$ edges obtained from $G_\lambda$ by edge/vertex deletions and/or edge contractions (with merging parallel edges by summing their capacities) for any $\lambda = \lambda_{\min} + \ell\varepsilon$ and any integer $\ell \in [0, R/\varepsilon]$. Then, $\varepsilon$-approximate parametric min $s,t$-cut in $G$ can be computed in $O(T_{\text{maxflow}}(n, m \log n) \cdot \log \frac{R}{\varepsilon} \cdot \log n)$ time.*

The algorithm is recursive. In order to ensure uniqueness of the minimum cuts considered, it always computes cuts with *minimal* $s$-side.

Roughly speaking, given initial guesses $\lambda_{\min}, \lambda_{\max}$ the algorithm finds, using $O(\log((\lambda_{\max} - \lambda_{\min})/\varepsilon))$ maximum flow computations, the most balanced split $\lambda_1, \lambda_2$ of the domain such that (1) the $s$-sides for all the min-cuts of $G_{\lambda'}$ for $\lambda' > \lambda_2$ have size at least $n/2$, (2) the $t$-sides of all the min-cuts of $G_{\lambda'}$ for $\lambda' < \lambda_1$ have size at least $n/2$, (3) $\lambda_2 - \lambda_1 \leq \varepsilon$. We then recurse on the intervals $[\lambda_{\min}, \lambda_1]$ and $[\lambda_2, \lambda_{\max}]$ on *minors* of $G$ with at least $n/2$ vertices contracted. Even though the contraction requires merging parallel edges in order to have at most $m + n$ (as opposed to $2m$) edges in the recursive calls, we are able to guarantee that the capacity functions in the recursive calls are all obtained by shifting the original capacity functions by a real number, and thus can be evaluated in constant time as well. Since the number of vertices decreases by a factor of two in every recursive call, one can prove that for each level of the recursion tree, the sum of numbers of vertices in the calls at that level is $O(n)$, whereas the sum of sizes of edge sets is $O(m + n \log n)$.

We show that the $\varepsilon$-approximate algorithm can be turned into an exact algorithm in two important special cases. First of all, if the capacity functions are low-degree (say, at most 4) polynomials with *integer coefficients* in $[-U, U]$, then one can compute parametric min $s,t$-cut only $O(\text{polylog}\{n, U\})$ factors slower than best known max-flow algorithm for integer capacities [73, 25, 75]. Second, we can solve the *discrete domain* case, i.e., when $\mathbb{D}$ has finite size $\ell$ with only $O(\text{polylog}\{n, \ell\})$ multiplicative overhead wrt. the respective maximum flow algorithm.

Moreover, since our reduction runs maximum flow computations only on *minors* of the input graph $G$, it also yields very efficient parametric min $s,t$-cut algorithms for planar graphs. In particular, since near-optimal strongly-polynomial $s,t$-max flow algorithms for planar graphs are known [22, 58], we obtain near-optimal algorithms for the integer polynomial capacity functions (as above) and discrete domains. What is perhaps more surprising, using our reduction we can even obtain a *strongly polynomial* exact parametric min $s,t$-cut algorithm for planar graphs with linear capacity functions with real coefficients. The algorithm runs in $\tilde{O}(n^{1.21875})$ time and constitutes the only known subquadratic strongly polynomial parametric min-$s,t$-cut algorithm.

The details of our parametric min $s,t$-cut algorithm and its applications are covered in Appendix 9.3.3. It should be noted that the idea of using cuts contraction is not new and appeared previously in [160]. Compared to [160], our reduction provably handles more general parameterized capacity functions. As it does not operate on any auxiliary networks that may not preserve structural properties of $G$, but merely on minors of $G$, it proves much more robust in important special cases such as planar graphs. Finally, we believe that our reduction is also more natural and operates on the breakpoints of the cut function directly, whereas the reduction of [160] operates on so-called balanced flows.

## 5.4 Parametric Decomposable Submodular Minimization via Base Polytope Approximations

### 5.4.1 Algorithm Overview

In Algorithm 9 we give the description of our the main routine.

**Algorithm 9** Parametric Decomposable Submodular Function Minimization

**Input:** $\varepsilon$: error tolerance ▷ Returns $\varepsilon$-optimal solution of $\min_{w \in B(F)} \psi^*(-w)$
Set $w^{0,i} = 0$ for $i \in [r]$ ▷ Initialize a feasible solution
Set $T = \alpha \log \frac{\psi^*(-w^0)+\psi(0)}{\varepsilon}$
**for** $t = 1 \dots T$ **do**
    Set $G_i(V_i, E_i, c_i) = \textsc{GraphApprox}(w^{t-1,i})$, for $i \in [r]$, and construct $G(V, E, c)$ by combining the graphs $G_i$
    Set $\phi(x) := \psi(x) + \langle \sum_{i=1}^{r} w^{t-1,i}, x \rangle$
    Set $\widetilde{w} = \textsc{FindMinCuts}(G(V, E, c), \phi, \frac{1}{3L})$ ▷ Find parametric min $s,t$-cuts
    Round all the entries of $\widetilde{w}$ to the nearest integer
    Decompose $\widetilde{w} = \sum_{i=1}^{r} \widetilde{w}^i$, with $\widetilde{w}_i \in B(G_i)$ using Lemma 5.4.3.
    Set $w^{t,i} = w^{t-1,i} + \widetilde{w}^i$, for all $i \in [r]$.
**return** $\sum_{i=1}^{r} w^{T,i}$

### 5.4.2 Removing Assumptions

In Section 5.2 we assumed that for all $i$, $F_i(\emptyset) = F_i(V_i) = 0$ and $F_i(S) \geq 0$ for all $S$. These assumptions hold without loss of generality. In Section 9.3.4 we show how to preprocess the input such that these assumptions are valid.

### 5.4.3 From Parametric Minimum Cut to Cut Base Polytope Optimization

In this section, we will focus on the problem of minimizing a convex function over the base polytope of the $s,t$-cut function of a graph $G(V \cup \{s,t\}, E, c)$. We define the $s,t$-cut function $G : 2^V \to \mathbb{R}$ as $G(S) = c^+(S \cup \{s\})$ for $S \subseteq V$, and let $B(G)$ be the base polytope of $G$. Now, the parametric min $s,t$-cut problem can be written as

$$\min_{S \subseteq V} G(S) + \sum_{u \in S} \phi'_u(\lambda)\,, \tag{5.6}$$

where the capacity of an edge $(u,t)$ at time $\lambda$ is $c_{ut} + \phi'_u(\lambda)$ and $\phi$ is a function satisfying Assumption 5.2.2. In particular, our goal in this section is, given solutions to (5.6) for all $\lambda$, to solve the following dual problem:

$$\min_{w \in B(G)} \phi^*(-w)\,. \tag{5.7}$$

**Definition 5.4.1** ($W$-restricted function)**.** *A submodular function $F : 2^V \to \mathbb{R}_{\geq 0}$ is called $W$-restricted if $F(S) = F(S \cap W)$ for all $S \subseteq V$, where $W \subseteq V$.*

As the cut functions $G(S)$ that we will be concerned with will be decomposable, i.e. $G(S) = \sum_{i=1}^{r} G_i(S)$ for all $S \subseteq V$, we introduce the following notion of a *decomposition* of some $w \in B(G)$ into a sum of $w^i \in B(G_i)$.

**Definition 5.4.2** ($F$-decomposition)**.** *Let $F : 2^V \to \mathbb{R}_{\geq 0}$ with $|V| = n$ be a submodular function that is decomposable, i.e. $F(S) = \sum_{i=1}^{r} F_i(S)$ for all $S \subseteq V$, where $F_i : 2^V \to \mathbb{R}_{\geq 0}$ are submodular functions. Then for any $w \in B(F)$ there exist vectors $w^1, \ldots, w^r \in \mathbb{R}^n$, where $w^i \in B(F_i)$ for all $i \in [r]$ and $\sum_{i=1}^{r} w^i = w$. We call the sequence of vectors $w^1, \ldots, w^r$ an $(F_1, \ldots, F_r)$-decomposition of $w$, or just an $F$-decomposition of $w$ if the $F_i$'s are clear from context.*

What follows is the main lemma of this section, whose full proof appears in Appendix 9.3.5.

**Lemma 5.4.3** (From parametric min-cut to cut base polytope optimization)**.** *Consider a graph $G(V, E, c \geq 0)$ and a function $\phi(x) = \sum_{u \in V} \phi_u(x_u)$ that satisfies Assumption 5.2.2. Additionally, let $G(S) = c^+(S)$ for all $S \subseteq V$ be the cut function associated with the graph, and suppose it is decomposable as $G(S) = \sum_{i=1}^{r} G_i(S)$ where $G_i : 2^V \to \mathbb{Z}_{\geq 0}$ are $V_i$-restricted cut functions defined as $G_i(S) = c^{i+}(S)$ that correspond to graphs $G_i(V, E, c^i \geq 0)$, and $c = \sum_{i=1}^{r} c^i$.*

*We define an extended vertex set $V' = V \cup \{s, t\}$ with edge set $E' = E \cup \bigcup_{u \in V} (s, u) \cup \bigcup_{u \in V} (u, t)$, the parametric capacity of an edge $(u, v) \in E'$ as*

$$c_\lambda(u, v) = \begin{cases} \max\{0, \phi'_u(-\lambda)\} & \text{if } u \in V, v = t \\ \max\{0, -\phi'_u(-\lambda)\} & \text{if } u = s, v \in V \\ c_{uv} & \text{otherwise} \end{cases}$$

*and let $(\Lambda, \tau)$ be a $\frac{1}{3L}$-approximate parametric min $s, t$-cut of $G'(V', E', c_\lambda)$.*

*There exists an algorithm that, given $(\Lambda, \tau)$, outputs $\widetilde{w}^* = \operatorname{argmin}_{w \in B(G)} \phi^*(-w)$ and a $G$-decomposition $\widetilde{w}^{*1}, \ldots, \widetilde{w}^{*r}$ of $\widetilde{w}^*$. The running time of this algorithm is $O\left(n + \sum_{i=1}^{r} |V_i|^2\right)$.*

### 5.4.4 Dual Progress Analysis in One Step

**Lemma 5.4.4** (Dual progress in one step)**.** *Let $F : 2^V \to \mathbb{Z}_{\geq 0}$ be a submodular function that is separable, i.e. $F(S) = \sum_{i=1}^{r} F_i(S)$ for all $S \subseteq V$, where $F_i : 2^V \to \mathbb{Z}_{\geq 0}$ are $V_i$-restricted submodular functions with $F_i(\emptyset) = 0$. Additionally, let $\psi : \mathbb{R}^n \to \mathbb{R}$ be a function that satisfies Assumption 5.2.2, where $|V| = n$.*

*Given a feasible dual solution $w \in B(F)$ and an $F$-decomposition $w^1, \ldots, w^r \in \mathbb{Z}^n$ of $w$, there is an algorithm that outputs a vector $w' \in \mathbb{Z}^n$, along with an $(F_1, \ldots, F_r)$-decomposition $w'^1, \ldots, w'^r \in \mathbb{Z}^n$ of $w'$, such that*

$$\psi^*(-w') - \psi^*(-w^*) \leq \left(1 - \frac{1}{\alpha}\right)(\psi^*(-w) - \psi^*(-w^*))$$

*where* $\alpha = \max_{i\in[r]} \{|V_i|^2/4\}$ *and* $w^* = \operatorname{argmin}_{w^*\in B(F)} \psi^*(-w^*)$ *is the dual optimum. The running time of the algorithm is*

$$\widetilde{O}\left(\sum_{i=1}^{r} |V_i|^2 \mathcal{O}_i + T_{\text{maxflow}}\left(n, n + \sum_{i=1}^{r} |V_i|^2\right)\right).$$

The full proof of Lemma 5.4.4 appears in Appendix 9.3.5.

### 5.4.5 Main Theorem

*Proof of Theorem 5.1.1.* We repeatedly apply Lemma 5.4.4, and let $w^0, \ldots, w^T$ be the iterates after $T = \alpha \log \frac{\psi^*(-w^0)-\psi^*(-w^*)}{\zeta}$ iterations. We have

$$\psi^*(-w^T) - \psi^*(-w^*) \leq \left(1 - \frac{1}{\alpha}\right)(\psi^*(-w^{T-1}) - \psi^*(-w^*)).$$

Applying induction over $T$ steps we obtain that

$$\psi^*(-w^T) - \psi^*(-w^*) \leq \left(1 - \frac{1}{\alpha}\right)^T (\psi^*(-w^0) - \psi^*(-w^*)) \leq e^{-T/\alpha}(\psi^*(-w^0) - \psi^*(-w^*)) \leq \zeta.$$

We have obtained a high precision solution to the objective (5.4). Finally, setting $\frac{1}{\zeta} = \text{poly}(\frac{L}{\sigma} n F_{\max}/\varepsilon) = n^{O(1)}/\varepsilon$ and applying Corollary 9.3.1 to convert from this solution to the actual sets, we obtain the desired solution.

As Assumption 5.2.2 implies $\psi^*(-w^0) - \psi^*(-w^*) = n^{O(1)}$, the total running time is

$$\widetilde{O}\left(\max_i |V_i|^2 \left(\sum_{i=1}^{r} |V_i|^2 \mathcal{O}_i + T_{\text{maxflow}}\left(n, n + \sum_{i=1}^{r} |V_i|^2\right)\right) \log \frac{1}{\varepsilon}\right).$$

□

## Acknowledgements

KA was supported in part by the NSF grants CCF-1553428 and CNS-1815221. AK, AM and PS were supported in part by ERC CoG project TUgbOAT no 772346. We are grateful to Michael B. Cohen for discussions on the potential of second order methods for submodular minimization, which motivated parts of this project. We thank Alina Ene for pointing us to the relevant material from [15].

# Part II

# Optimization Under Sparsity Constraints

# Chapter 6

# Sparse Convex Optimization via Adaptively Regularized Hard Thresholding

## 6.1 Introduction

*Sparse Convex Optimization* is the problem of optimizing a convex objective, while constraining the sparsity of the solution (its number of non-zero entries). Variants and special cases of this problem have been studied for many years, and there have been countless applications in Machine Learning, Signal Processing, and Statistics. In Machine Learning it is used to regularize models by enforcing parameter sparsity, since a sparse set of parameters often leads to better model generalization. Furthermore, in a lot of large scale applications the number of parameters of a trained model is a significant factor in computational efficiency, thus improved sparsity can lead to improved time and memory performance. In applied statistics, a single extra feature translates to a real cost from increasing the number of samples. In compressed sensing, finding a sparse solution to a Linear Regression problem can be used to significantly reduce the sample size for the recovery of a target signal. In the context of these applications, decreasing sparsity by even a small amount while not increasing the accuracy can have a significant impact.

### 6.1.1 Sparse Optimization

Given a function $f : \mathbb{R}^n \to \mathbb{R}$ and any $s^*$-sparse *(unknown) target solution* $x^*$, the Sparse Optimization problem is to find an $s$-sparse solution $x$, i.e. a solution with at most $s$ non-zero entries, such that $f(x) \leq f(x^*) + \varepsilon$ and $s \leq s^*\gamma$, where $\varepsilon > 0$ is a desired accuracy and $\gamma \geq 1$ is an approximation factor for the target sparsity. Even if $f$ is a convex function, the sparsity constraint makes this problem non-convex, and it has been shown that it is an intractable problem, even when $\gamma = O\left(2^{\log^{1-\delta} n}\right)$ and $f$ is the Linear Regression objective [127, 63]. However, this worst-case behavior is not observed in practice, and so a large body of work has been devoted to the analysis of algorithms under the assumption that the *restricted condition number* $\kappa_{s+s^*} = \frac{\rho^+_{s+s^*}}{\rho^-_{s+s^*}}$ (or just $\kappa = \frac{\rho^+}{\rho^-}$) of $f$ is bounded [127, 146, 170, 16, 113, 89, 169, 148, 149, 89, 154]. Note: Here, $\rho^+_{s+s^*}$ is the maximum smoothness constant of any restriction of $f$ on an $(s+s^*)$-sparse subset of coordinates and $\rho^-_{s+s^*}$ is the minimum strong convexity constant of any restriction of $f$ on an $(s+s^*)$-sparse subset of coordinates.

The first algorithm for this problem, often called *Orthogonal Matching Pursuit (OMP)* or *Greedy*,

was analyzed by [127] for Linear Regression, and subsequently for general $f$ by [146], obtaining the guarantee that the sparsity of the returned solution is $O\left(s^*\kappa \log \frac{f(x^0)-f(x^*)}{\varepsilon}\right)$. [1] In applications where having low sparsity is crucial, the dependence of sparsity on the required accuracy $\varepsilon$ is undesirable. The question of whether this dependence can be removed was answered positively [146, 89] giving a sparsity guarantee of $O(s^*\kappa^2)$. As remarked in [146], this bound sacrifices the linear dependence on $\kappa$, while removing the dependence on $\varepsilon$ and $f(x^0) - f(x^*)$.

Since then, there has been some work on improving these results by introducing non-trivial assumptions, such as the target solution $x^*$ being close to globally optimal. More specifically, [170] defines the *Restricted Gradient Optimal Constant (RGOC) at level s*, $\zeta_s$ (or just $\zeta$) as the $\ell_2$ norm of the top-$s$ elements in $\nabla f(x^*)$ and analyzes an algorithm that gives sparsity $s = O\left(s^*\kappa \log\left(s^*\kappa\right)\right)$, and such that $f(x) \leq f(x^*) + O(\zeta^2/\rho^-)$. [154] strengthens this bound to $f(x) \leq f(x^*) + O(\zeta^2/\rho^+)$ with sparsity $s = O\left(s^*\kappa \log \kappa\right)$. However, this means that $f(x)$ might be much larger than $f(x^*) + \varepsilon$ in general. To the best of our knowledge, no improvement has been made over the $O\left(s^* \min\left\{\kappa \frac{f(x^0)-f(x^*)}{\varepsilon}, \kappa^2\right\}\right)$ bound in the general case.

**Related work.** Sparse convex optimization is closely related to the problem of optimizing a convex function under a rank constraint, which is a very general optimization problem encompassing matrix completion, robust principal component analysis, and others. Close analogues of OMP and OMPR that give guarantees on the rank of the solution based on the condition number have been analyzed for this setting [144, 13].

Another line of work studies a maximization version of the sparse convex optimization problem as well as its generalizations for matroid constraints [6, 55, 38].

### 6.1.2 Sparse Solution and Support Recovery

Often, as is the case in compressed sensing, one needs a guarantee on the closeness of the solution $x$ to the target solution $x^*$ in absolute terms, rather than in terms of the value of $f$. The goal is usually either to recover (a superset of) the target support, or to ensure that the returned solution is close to the target solution in $\ell_2$ norm. The results for this problem either assume a constant upper bound on the *Restricted Isometry Property (RIP)* constant $\delta_r := \frac{\kappa_r - 1}{\kappa_r + 1}$ for some $r$ (RIP-based recovery), or that $x^*$ is close to being a global optimum (RIP-free recovery). This problem has been extensively studied and is an active research area in the vast compressed sensing literature. See also the survey by [21].

In the seminal papers of [33, 32, 46, 31] it was shown that for the Linear Regression problem when $\delta_{2s^*} < \sqrt{2} - 1 \approx 0.41$, the LASSO algorithm [161] can recover a solution with $\|x - x^*\|_2^2 \leq C f(x^*)$, where $C$ is a constant depending only on $\delta_{2s^*}$ and $f(x^*) = \frac{1}{2}\|Ax^* - b\|_2^2$ is the error of the target solution[2]. Since then, a multitude of results of similar flavor have appeared, either giving related guarantees for the LASSO algorithm while improving the RIP upper bound [67, 30, 64, 29, 124, 7] which culminate in a bound of $\delta_{2s^*} < 0.6248$, or showing that similar guarantees can be obtained by greedy algorithms under more restricted RIP conditions, but that are typically faster than LASSO [130, 129, 128, 19, 87, 65, 66]. See also the comprehensive surveys by [68, 125].

[128] presents a greedy algorithm called *CoSaMP* and shows that for Linear Regression it achieves a bound in the form of [31] while having a more efficient implementation. Their method works for the more restricted RIP upper bound of $\delta_{2s^*} < 0.025$, or $\delta_{4s^*} < 0.4782$ as improved by [68]. [19] proves that another greedy algorithm called *Iterative Hard Thresholding (IHT)* achieves a similar

[1] Even though [127] states a less general result, this is what is implicitly proven.

[2] $f(x^*)$ is also commonly denoted as $\frac{1}{2}\|\eta\|_2^2$, where $Ax^* = b + \eta$, i.e. $\eta$ is the measurement *noise*.

bound to that of CoSaMP for Linear Regression, with the condition $\delta_{3s^*} < 0.067$, which is improved to $\delta_{2s^*} < \frac{1}{3}$ by [87] and to $\delta_{3s^*} < 0.5774$ by [65].

The RIP-free line of research has shown that strong results can be achieved without a RIP upper bound, given that the target solution is sufficiently close to being a global optimum. These results typically require that $s$ is significantly larger than $s^*$. In particular, [170] shows that if $\zeta$ is the RGOC of $f$ it can be guaranteed that $\|x - x^*\|_2 \leq 2\sqrt{6}\frac{\zeta}{\rho^-}$ (or $(1+\sqrt{6})\frac{\zeta}{\rho^-}$ with a slightly tighter analysis). [154] strengthens this bound to $\left(1+\sqrt{1+\frac{5}{\kappa}}\right)\frac{\zeta}{\rho^-}$. Furthermore, it has been shown that as long as a "Signal-to-Noise" condition holds, one can actually recover a superset of the target support. Typically the condition is a lower bound on $|x^*_{\min}|$, the minimum magnitude non-zero entry of the target solution. Different lower bounds that have been devised include $\Omega\left(\frac{\sqrt{s+s^*}\|\nabla f(x^*)\|_\infty}{\rho^-_{s+s^*}}\right)$ [89], which was later improved to $\Omega\left(\sqrt{\frac{f(x^*)-f(\overline{x}^*)}{\rho^-_{2s}}}\right)$, where $\overline{x}^*$ is an optimal $s$-sparse solution [169]. Finally, [154] improves the sparsity bound to $O(s^*\kappa\log(s^*\kappa))$ in the statistical setting and [149] shows that the sparsity can be brought down to $s = s^* + O(\kappa^2)$ if a stronger lower bound of $\Omega\left(\sqrt{\kappa}\frac{\zeta}{\rho}\right)$ is assumed.

### 6.1.3 Our Work

In this work we present a new algorithm called *Adaptively Regularized Hard Thresholding (ARHT)*, that closes the longstanding gap between the $O\left(s^*\kappa\frac{f(x^0)-f(x^*)}{\varepsilon}\right)$ and $O\left(s^*\kappa^2\right)$ bounds by getting a sparsity of $O(s^*\kappa)$ and thus achieving the best of both worlds. As [63] shows that for a general class of algorithms (including greedy algorithms like OMP, IHT as well as LASSO) the linear dependence on $\kappa$ is necessary even for the special case of Sparse Regression, our result is tight for this class of algorithms. In Section 6.5.1 we briefly describe this example and also state a conjecture that it can be turned into an inapproximability result in Conjecture 6.5.1. Furthermore, in Section 6.5.2 we show that the $O(s^*\kappa^2)$ sparsity bound is tight for OMPR, thus highlighting the importance of regularization in our method. Our algorithm is efficient, as it requires roughly $O\left(s\log^3\frac{f(x^0)-f(x^*)}{\varepsilon}\right)$ iterations, each of which includes one function minimization in a restricted support of size $s$ and is simple to describe and implement. Furthermore, it directly implies non-trivial results in the area of compressed sensing.

We also provide a new analysis of OMPR [87] and show that under the condition that $s > s^*\frac{\kappa^2}{4}$, or equivalently under the RIP condition $\delta_{s+s^*} < \frac{2\sqrt{\frac{s}{s^*}}-1}{2\sqrt{\frac{s}{s^*}}+1}$, it is possible to approximately minimize the function $f$ up to some error depending on the RIP constant and the closeness of $x^*$ to global optimality. More specifically, we show that for any $\varepsilon > 0$ OMPR returns a solution $x$ such that

$$f(x) \leq f(x^*) + \varepsilon + C_1(f(x^*) - f(x^{\text{opt}}))\,,$$

where $x^{\text{opt}}$ is the globally optimal solution, as well as

$$\|x - x^*\|_2^2 \leq \varepsilon + C_2(f(x^*) - f(x^{\text{opt}}))]\,,$$

where $C_1, C_2$ are constants that only depend on $\frac{s}{s^*}$ and $\delta_{s+s^*}$. An important feature of our approach is that it provides a meaningful tradeoff between the RIP constant upper bound and the sparsity of the solution, even when the sparsity $s$ is arbitrarily close to $s^*$. In other words, one can relax the RIP condition at the expense of increasing the sparsity of the returned solution. Furthermore, our

Table 6.1: Compressed sensing tradeoffs implied by Theorem 6.4.2: Sparsity vs RIP condition

| $s$ | RIP condition |
|---|---|
| $s^*$ | $\delta_{2s^*} < 0.33$ |
| $2s^*$ | $\delta_{3s^*} < 0.47$ |
| $3s^*$ | $\delta_{4s^*} < 0.55$ |
| $30s^*$ | $\delta_{31s^*} < 0.83$ |

analysis applies to general functions with bounded RIP constant.

Experiments with real data suggest that ARHT and a variant of OMPR which we call *Exhaustive Local Search* achieve promising performance in recovering sparse solutions.

### 6.1.4 Comparison to Previous Work

In this section we compare our results with previous work on sparse optimization, solution, and support recovery.

#### Sparse Optimization

Our Algorithm 15 (ARHT) returns a solution with $s = O(s^*\kappa)$ without any additional assumptions, thus significantly improving over the bound $O\left(s^* \min\left\{\kappa\frac{f(x^0)-f(x^*)}{\varepsilon}, \kappa^2\right\}\right)$ that was known in previous work. This proves that neither any dependence on the required solution accuracy $\varepsilon$, nor the quadratic dependence on the condition number $\kappa$ is necessary. Furthermore, no assumption on the function or the target solution is required to achieve this bound. Importantly, previous results imply that our bound is tight up to constants for a general class of algorithms, including Greedy-type algorithms and LASSO [63].

#### Sparse Solution Recovery

In Corollary 6.3.11, we show that the improved guarantees of Theorem 6.3.1 immediately imply that ARHT gives a bound of $\|x - x^*\|_2 \leq (2+\theta)\frac{\zeta}{\rho^-}$ for any $\theta > 0$, where $\zeta$ is the Restricted Gradient Optimal Constant. This improves the constant factor in front of the corresponding results of [170, 154].

As we saw, our Theorem 6.4.2 directly implies that OMPR gives an upper bound on $\|x - x^*\|_2^2$ of the same form as the RIP-based bounds in previous work, under the condition $\delta_{s+s^*} < \frac{2\sqrt{\frac{s}{s^*}}-1}{2\sqrt{\frac{s}{s^*}}+1}$. While previous results either concentrate on the case $s = s^*$, or $s\boldsymbol{g}s^*$, our analysis provides a way to trade off increased sparsity for a more relaxed RIP bound, allowing for a whole family of RIP conditions where $s$ is arbitrarily close to $s^*$. Specifically, if we set $s = s^*$ our work implies recovery for $\delta_{2s^*} < \frac{1}{3} \approx 0.33$, which matches the best known bound for any greedy algorithm [87], although it is a stricter condition than the $\delta_{2s^*} < 0.62$ required by LASSO [68]. Table 6.1 contains a few such RIP bounds implied by our analysis and shows that it readily surpasses the bounds for Subspace Pursuit $\delta_{3s^*} < 0.35$, CoSaMP $\delta_{4s^*} < 0.48$, and OMP $\delta_{31s^*} < 0.33$ [87, 170]. Another important feature compared to previous work is that all our guarantees are not restricted to Linear Regression and are true for any function $f$, as long as it satisfies the required RIP condition, which makes the result more general.

**Sparse Support Recovery**

Corollary 6.3.12 shows that as a direct consequence of our work, the condition $|x^*_{\min}| > \frac{\zeta}{\rho^-}$ suffices for our algorithm to recover a superset of the support with size $s = O(s^*\kappa)$. Compared to [89], we improve both the size of the superset, as well as the condition, since $\sqrt{s}\frac{\|\nabla f(x^*)\|_\infty}{\rho^-} \geq \sqrt{\frac{s}{s^*}}\frac{\zeta}{\rho^-} = \Omega\left(\frac{\zeta}{\rho^-}\right)$. Compared to [149], the bounds on the superset size are incomparable in general, but our $|x^*_{\min}|$ condition is more relaxed, since $\sqrt{\kappa}\frac{\zeta}{\rho^-} = \Omega(\frac{\zeta}{\rho^-})$. Finally, [169] works under the condition $|x^*_{\min}| > \sqrt{\frac{2(f(x^*) - \min_{\|x\|_0 \leq s} f(x))}{\rho^-}}$, which is more relaxed since this quantity is always in $\left[\frac{1}{\sqrt{\kappa}}\frac{\zeta}{\rho^-}, \frac{\zeta}{\rho^-}\right]$, but uses a larger superset size of $O(s^*\kappa^2)$ instead of $O(s^*\kappa)$. Although not explicitly stated, [170, 154] also give a similar lower bound of $\sqrt{1 + \frac{10}{\kappa}}\frac{\zeta}{\rho^-}$ which we improve by a constant factor.

### 6.1.5 Runtime discussion

ARHT has the advantage of being very simple to implement in practice. The runtime of Algorithm 15 (ARHT) is comparable to that of the most efficient greedy algorithms (e.g. OMP/OMPR), as it requires a single function minimization per iteration. On the other hand, Algorithm 13 (Exhaustive Local Search) is less efficient, as it requires $O(n)$ function minimizations in each iteration, although in practice one might be able to speed it up by exploiting the fact that the problems being solved in each iteration are very closely related.

### 6.1.6 Naming Conventions

The algorithm that we call *Orthogonal Matching Pursuit (OMP)*, is also known as "Greedy" [127], "Fully Corrective Forward Greedy Selection" or just "Forward Selection". What we call *Orthogonal Matching Pursuit with Replacement (OMPR)* [87] is also known by various other names. It is referenced in [146] as a simpler variant of their "Fully Corrective Forward Greedy Selection with Replacement" algorithm, or just Forward Selection with Replacement, or "Partial Hard Thresholding with parameter $r = 1$ (PHT($r$) where $r = 1$)" [88] which is a generalization of Iterative Hard Thresholding. Finally, what we call *Exhaustive Local Search* is essentially a variant of "Orthogonal Least Squares" that includes replacement steps, and also appears in [146] as "Fully Corrective Forward Greedy Selection with Replacement", or just "Forward Stepwise Selection with Replacement". See also [20] regarding naming conventions.

*Remark:* Most of the results in the literature either only apply to, or are only presented for the Linear Regression problem. Since all of our results apply to general function minimization, we present them as such.

## 6.2 Preliminaries

We will make use of $\widetilde{\kappa}_s = \rho_2^+/\rho_s^-$ which is at most $\kappa_s$ as long as $s \geq 2$. The following lemma stems from the definitions of $\rho_2^+, \rho_1^+$ and can be used to relate $\rho_2^+$ with $\rho_1^+$

**Lemma 6.2.1.** *For any function $f$ that has the RSC property at sparsity level $\geq 2$ and RSS constants $\rho_1^+, \rho_2^+$ at sparsity levels 1 and 2 respectively, we have $\rho_2^+ \leq 2\rho_1^+$.*

*Proof.* For any $x, y \in \mathbb{R}^n$ such that $|\text{supp}(y - x)| \leq 2$, We will prove that

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{2\rho_1^+}{2} \|y - x\|_2^2 .$$

Let $y = x + \alpha \vec{1}_i + \beta \vec{1}_j$ for some $i, j \in [n]$ and $\alpha, \beta \in \mathbb{R}$. We assume $i \neq j$ and since otherwise the claim already follows from RSS at sparsity level 1. We apply the RSS property with sparsity level 1 to get the inequalities

$$f(x + 2\alpha \vec{1}_i) \leq f(x) + 2\langle \nabla f(x), \alpha \vec{1}_i \rangle + 4\frac{\rho_1^+}{2} \left\| \alpha \vec{1}_i \right\|_2^2$$

and

$$f(x + 2\beta \vec{1}_j) \leq f(x) + 2\langle \nabla f(x), \beta \vec{1}_j \rangle + 4\frac{\rho_1^+}{2} \left\| \beta \vec{1}_j \right\|_2^2 .$$

Now, by using convexity (more precisely restricted convexity at sparsity level 2 that is implied by RSC) we have

$$\begin{aligned} f(y) &= f(x + \alpha \vec{1}_i + \beta \vec{1}_j) \\ &\leq \frac{1}{2} \left( f(x + 2\alpha \vec{1}_i) + f(x + 2\beta \vec{1}_j) \right) \\ &\leq f(x) + \langle \nabla f(x), \alpha \vec{1}_i + \beta \vec{1}_j \rangle + \frac{2\rho_1^+}{2} \left\| \alpha \vec{1}_i + \beta \vec{1}_j \right\|_2^2 \\ &= f(x) + \langle \nabla f(x), y - x \rangle + \frac{2\rho_1^+}{2} \|y - x\|_2^2 . \end{aligned}$$

□

**Definition 6.2.2** (Restricted Gradient Optimal Constant (RGOC))**.** *Given a differentiable function $f$ and a "target" solution $x^*$, the* Restricted Gradient Optimal Constant *[170] at sparsity level $s$ is the minimum $\zeta_s \in \mathbb{R}_+$ such that*

$$|\langle \nabla f(x^*), y \rangle| \leq \zeta_s \|y\|_2$$

*for all $s$-sparse $y$. Setting $y = \nabla_S f(x^*)$ for some set $S$ with $|S| \leq s$, this implies that $\zeta_s \geq \|\nabla_S f(x^*)\|$. An alternative definition is that $\zeta_s$ is the $\ell_2$ norm of the $s$ elements of $\nabla f(x^*)$ with highest absolute value.*

**Definition 6.2.3** ($S$-restricted minimizer)**.** *Given $f : \mathbb{R}^n \to \mathbb{R}$, $x^* \in \mathbb{R}^n$, and $S \subseteq [n]$, we will call $x^*$ an $S$-restricted minimizer of $f$ if* $\text{supp}(x^*) \subseteq S$ *and for all $x$ such that* $\text{supp}(x) \subseteq S$ *we have $f(x^*) \leq f(x)$.*

In Lemma 6.2.4 we state a standard martingale concentration inequality that we will use. See also [40] for more on martingales.

**Lemma 6.2.4** (Martingale concentration inequality [40])**.** *Let $Y_0 = 0, Y_1, \ldots, Y_n$ be a martingale with respect to the sequence $i_1, \ldots, i_n$ such that*

$$\text{Var}\left(Y_k \mid i_1, \ldots, i_{k-1}\right) \leq \sigma^2$$

*and*

$$Y_{k-1} - Y_k \leq M ,$$

*for all $k \in [n]$, then for any $\lambda > 0$,*

$$\Pr\left[Y_n \leq -\lambda\right] \leq e^{-\lambda^2/\left(2\left(n\sigma^2 + M\lambda/3\right)\right)}.$$

### 6.2.1 Algorithms

#### $\ell_1$ optimization (LASSO)

The LASSO approach is to relax the $\ell_0$ constraint into an $\ell_1$ one, thus solving the following optimization problem:

$$\min_x f(x) + \lambda \|x\|_1 , \tag{6.1}$$

for some parameter $\lambda > 0$.

#### Iterative Hard Thresholding (IHT):

[19] define the hard thresholding operator $H_r(x)$ as

$$[H_r(x)]_i = \begin{cases} x_i & \text{if } |x_i| \text{ is one of the } r \text{ entries of } x \\ & \text{with largest magnitude} \\ 0 & \text{otherwise} \end{cases}.$$

Using this, the algorithm is described in Algorithm 10.

**Algorithm 10** Iterative Hard Thresholding (IHT)

**function** IHT($s, T$)
    function to be minimized $f : \mathbb{R}^n \to \mathbb{R}$
    number of iterations $T$
    output sparsity $s$
    $S^0 \leftarrow \emptyset$
    $x^0 \leftarrow \vec{0}$
    **for** $t = 0 \ldots T-1$ **do**
        $x^{t+1} \leftarrow H_s\left(x^t - \eta \nabla f(x^t)\right)$
    **return** $x^T$

#### Orthogonal Matching Pursuit (Greedy/OMP/Fwd stepwise selection)

The algorithm is described in Algorithm 11.

#### Orthogonal Matching Pursuit with Replacement (Local search/OMPR/Fwd stepwise selection with replacement steps)

The algorithm is described in Algorithm 12.

#### Exhaustive Local Search

The algorithm in this section is similar to OMPR, in that it iteratively inserts a new element in the support while removing one from it at the same time. While, as in OMPR, the element to be

**Algorithm 11** Greedy/OMP/Fwd stepwise selection

**function** greedy($s$)
    function to be minimized $f : \mathbb{R}^n \to \mathbb{R}$
    output sparsity $s$
    $x^0 \leftarrow \vec{0}$
    **for** $t = 0 \dots s-1$ **do**
        $i \leftarrow \operatorname{argmax}\{|\nabla_i f(x^t)| \mid i \in [n] \backslash S^t\}$
        $S^{t+1} \leftarrow S^t \cup \{i\}$
        $x^{t+1} \leftarrow \operatorname{argmin}\{f(x) \mid \operatorname{supp}(x) \subseteq S^{t+1}\}$
    **return** $x^s$

**Algorithm 12** Orthogonal Matching Pursuit with Replacement

**function** OMPR($s$)
    function to be minimized $f : \mathbb{R}^n \to \mathbb{R}$
    output sparsity $s$
    $S^0 \leftarrow [s]$
    $x^0 \leftarrow \operatorname{argmin}\{f(x) \mid \operatorname{supp}(x) \subseteq S^0\}$
    $t \leftarrow 0$
    **while true do**
        $i \leftarrow \operatorname{argmax}\{|\nabla_i f(x^t)| \mid i \in [n] \backslash S^t\}$
        $j \leftarrow \operatorname{argmin}\{|x^t_j| \mid j \in S^t\}$
        $S^{t+1} \leftarrow S^t \cup \{i\} \backslash \{j\}$
        $x^{t+1} \leftarrow \operatorname{argmin}\{f(x) \mid \operatorname{supp}(x) \subseteq S^{t+1}\}$
        **if** $f(x^{t+1}) \geq f(x^t)$ **then**
            **break**
        $t \leftarrow t+1$
    $T \leftarrow t$
    **return** $x^T$

removed is the minimum magnitude entry, the one to be inserted is chosen to be the one which results in the maximum decrease in the value of the objective. It is described in Algorithm 13.

**Remark 6.2.5.** *In the following sections, we will denote the minimization objective by $f$, the RSS and RSC parameters $\rho^+_2$ and $\rho^-_{s+s^*}$ by $\rho^+$ and $\rho^-$ respectively, as well as $\kappa = \frac{\rho^+_{s+s^*}}{\rho^-_{s+s^*}}$ and $\widetilde{\kappa} = \frac{\rho^+_2}{\rho^-_{s+s^*}}$. Note that the use of $\rho^+_2$ instead of $\rho^+_1$ used in some works is not restrictive. As shown in Lemma 6.2.1, $\rho^+_2 \leq 2\rho^+_1$ and so in all the bounds involving $\widetilde{\kappa}$, it can be replaced by $2\frac{\rho^+_1}{\rho^-_{s+s^*}}$, thus only losing a factor of 2. Furthermore, we state our results in terms of $\widetilde{\kappa}$ as opposed to $\kappa$. This is always more general since $\widetilde{\kappa} \leq \kappa$.*

*In order to simplify the exposition, we will assume that $\min_x f(x) = 0$. This property can be ensured by adding a constant to $f$.*

*When no additional context is provided, we denote current solution by $x$ and the target solution $x^*$, with respective support sets $S$ and $S^*$ and sparsities $s = |S|$ and $s^* = |S^*|$.*

**Algorithm 13** Exhaustive Local Search

function to be minimized $f : \mathbb{R}^n \to \mathbb{R}$
target sparsity $s$
number of iterations $T$
$S^0 \leftarrow [s]$
$x^0 \leftarrow \operatorname{argmin}\{f(x) \mid \operatorname{supp}(x) \subseteq S^0\}$
**for** $t = 0 \dots T-1$ **do**
    $j \leftarrow \operatorname*{argmin}_{j \in S^t} x_j^2$
    $i \leftarrow \operatorname*{argmin}_{i \in [n] \setminus S^t} \left\{ \min_{x \,:\, \operatorname{supp}(x) \subseteq S^t \cup \{i\} \setminus \{j\}} f(x) \right\}$
    $S^{t+1} \leftarrow S^t \cup \{i\} \setminus \{j\}$
    $x^{t+1} \leftarrow \operatorname{argmin}\{f(x) \mid \operatorname{supp}(x) \subseteq S^{t+1}\}$
    **if** $f(x^{t+1}) \geq f(x^t)$ **then**
        **return** $x^t$
**return** $x^T$

## 6.3 Adaptively Regularized Hard Thresholding (ARHT)

### 6.3.1 Overview and Main Theorem

Our algorithm is essentially a hard thresholding algorithm (and more specifically OMPR, also known as PHT(1)) with the crucial novelty that it is applied on an adaptively regularized objective function. Hard thresholding algorithms maintain a solution $x$ supported on $S \subseteq [n]$, which they iteratively update by inserting new elements into the support set $S$ and removing the same number of elements from it, in order to preserve the sparsity of $x$. More specifically, OMPR makes one insertion and one removal in each iteration. In order to evaluate the element $i$ to be *inserted* into $S$, OMPR uses the fact that, because of smoothness, $\frac{(\nabla_i f(x))^2}{2\rho_2^+}$ is a lower bound on the decrease of $f(x)$ caused by inserting $i$ into the support, and therefore picks $i$ to maximize $|\nabla_i f(x)|$. Similarly, in order to evaluate the element $j$ to be *removed* from $S$, OMPR uses the fact that $\frac{\rho_2^+}{2} x_j^2$ upper bounds the increase of $f(x)$ caused by setting $x_j = 0$, and therefore picks $j$ to minimize $|x_j|$. However, the real worth of $j$ might be as small as $\frac{\rho_2^-}{2} x_j^2$, so the upper bound can be loose by a factor of $\frac{\rho_2^+}{\rho_2^-} \geq \frac{\rho_2^+}{\rho_{s+s^*}^-} = \widetilde{\kappa}$.

We eliminate this discrepancy by running the algorithm on the regularized function $g(z) := f(z) + \frac{\rho_2^+}{2} \|z\|_2^2$. As the restricted condition number of $g$ is now $O(1)$, the real worth of a removal candidate $j$ matches the upper bound up to a constant factor.

However, even though $g$ is now well conditioned, the analysis can only guarantee the quality of the solution in terms of the *original* objective $f$ if the regularization is *not* applied on elements $S^*$, i.e. $\frac{\rho_2^+}{2} \left\| x_{R \setminus S^*} \right\|_2^2$ for some sufficiently large $R \subseteq [n]$; if this is the case, a solution with sparsity $O(s^* \widetilde{\kappa})$ can be recovered. Unfortunately, there is no way of knowing a priori which elements not to regularize, as this is equivalent to finding the target solution. As a result, the algorithm can get trapped in local minima, which are defined as states in which one iteration of the algorithm does not decrease $g(x)$, even though $x$ is a suboptimal solution in terms of $f$ (i.e. $f(x) > f(x^*)$).

The main contribution of this work is to characterize such local minima and devise a procedure that is able to successfully escape them, thus allowing $x$ to converge to a desired solution for the original objective.

The core algorithm is presented in Algorithm 14. The full algorithm additionally requires some

standard routines like binary search and is presented in Algorithm 15.

**Algorithm 14** Adaptively Regularized Hard Thresholding core routine

**function** ARHT_CORE($s, \widehat{f}^*, \varepsilon$)
  function to be minimized $f : \mathbb{R}^n \to \mathbb{R}$
  target sparsity $s$
  target value $\widehat{f}^*$ (current guess for the optimal value)
  target error $\varepsilon$
  Define $g_R(x) := f(x) + \frac{\rho_2^+}{2} \|x_R\|_2^2$ for all $R \subseteq [n]$.
  $R^0 \leftarrow [n]$
  $S^0 \leftarrow [s]$
  $x^0 \leftarrow \underset{\text{supp}(x) \subseteq S^0}{\text{argmin}} \; g_{R^0}(x)$
  $T = 2s \log \frac{f(\vec{0}) - \min_x f(x)}{\varepsilon}$ (number of iterations)
  **for** $t = 0 \dots T-1$ **do**
    **if** $\min_{\text{supp}(x) \subseteq S^t} f(x) \leq \widehat{f}^*$ **then**
      **return** $\underset{\text{supp}(x) \subseteq S^t}{\text{argmin}} \; f(x)$
    $i \leftarrow \underset{i \in [n]}{\text{argmax}} \; |\nabla_i g_{R^t}(x^t)|$
    $j \leftarrow \underset{j \in S^t}{\text{argmin}} \; |x_j|$
    $S^{t+1} \leftarrow S^t \cup \{i\} \backslash \{j\}$
    $x^{t+1} \leftarrow \underset{\text{supp}(x) \subseteq S^{t+1}}{\text{argmin}} \; g_{R^t}(x)$
    **if** $g_{R^t}(x^t) - g_{R^t}(x^{t+1}) < \frac{1}{s}\left(g_{R^t}(x^t) - \widehat{f}^*\right)$ **then**
      $S^{t+1} \leftarrow S^t$
      Sample $i \in R^t$ proportional to $(x_i^t)^2$
      $R^{t+1} \leftarrow R^t \backslash \{i\}$
      $x^{t+1} \leftarrow \underset{\text{supp}(x) \subseteq S^{t+1}}{\text{argmin}} \; g_{R^{t+1}}(x)$
  **return** $x^T$

In the following, we will let $\widehat{f}^*$ denote a guess on the target value $f(x^*)$. Also, $x^0$ will denote the initial solution, which is an $S^0$-restricted minimizer for an arbitrary set $S^0 \subseteq [n]$ with $|S^0| = s$. In Algorithm 14, $S^0$ is defined explicitly as $[s]$, however in practice one might want to pick a better initial set (e.g. returned by running OMP).

It should be noted that even though the value $\rho_2^+$ is used by the algorithm to define the regularizer, exact knowledge of $\rho_2^+$ is not required, and an upper bound $U$ on it can be used. Of course, the final sparsity and runtime bound will then depend on $U/\rho_{s+s^*}^-$ instead of $\rho_2^+/\rho_{s+s^*}^-$. One such upper bound is $2\rho_1^+$. For linear and logistic regression where $A$ is the (# examples) $\times$ (# features) data matrix, $\rho_1^+$ is the maximum $\ell_2$ norm of a column of the data matrix $A$ (so $\rho_1^+ = 1$ if the columns of $A$ are normalized). More generally, getting such a bound would depend on the specific function we are trying to minimize. For example, if we are trying to minimize $\sum_{i=1}^m L(Ax)_i$ where $A$ is a data matrix with normalized columns and $L : \mathbb{R} \to \mathbb{R}$ is a 1-smooth loss function, then $\rho_1^+ \leq 1$ so 2 is a good upper bound for $\rho_2^+$. Another option is to run the algorithm for different candidate values for

**Algorithm 15** Adaptively Regularized Hard Thresholding

**function** ARHT_ROBUST($s, \widehat{f}^*, \varepsilon$)
  function to be minimized $f : \mathbb{R}^n \to \mathbb{R}$
  $x^{\text{ret}} \leftarrow \vec{0}$
  **for** $z = 1 \dots 5 \log\left(6n \log \frac{f(\vec{0})}{\varepsilon}\right)$ **do**
    $x \leftarrow \text{ARHT\_core}(s, \widehat{f}^*, \varepsilon)$
    **if** $f(x) < f(x^{\text{ret}})$ **then**
      $x^{\text{ret}} \leftarrow x$
  **return** $x^{\text{ret}}$
**function** ARHT($s, \varepsilon$)
  function to be minimized $f : \mathbb{R}^n \to \mathbb{R}$
  target sparsity $s$
  target error $\varepsilon$
  $l \leftarrow \vec{0}$
  $r \leftarrow f(\vec{0})$
  $b \leftarrow \vec{0}$
  **while** $r - l > \varepsilon$ **do**
    $m \leftarrow \frac{l+r}{2}$
    $x \leftarrow \text{ARHT\_robust}(s, m, \varepsilon/3)$
    **if** $f(x) > m + \varepsilon/3$ **then**
      $l \leftarrow m$
    **else**
      $b \leftarrow x$
      $r \leftarrow f(x)$
  **return** $b$

$\rho_2^+$ until we get the desired performance.

We are now ready for the main result of this section. It basically states that for any solution $x^*$ with sparsity $s^*$, a solution $x$ with $f(x) \leq f(x^*) + \varepsilon$ and sparsity $O(\kappa s^*)$ can be recovered by Algorithm 15. In comparison, previously known analyses could guarantee either a sparsity $O(\kappa s^* \log \frac{f(\vec{0})}{\varepsilon})$ (OMP) or $O(\kappa^2 s^*)$ (OMPR, IHT, LASSO). It is useful to note here that, because of the constant factor in front of $\kappa s^*$ (which is less than 20), this result cannot be directly used to obtain compressed sensing results with sparsity close to $s^*$, but is instead useful in the asymptotic regime (with relaxed sparsity $s\boldsymbol{g}s^*$).

**Theorem 6.3.1.** *Given a function $f$ and an (unknown) $s^*$-sparse solution $x^*$, with probability at least $1 - \frac{1}{n}$ Algorithm 15 returns an $s$-sparse solution $x$ with $f(x) \leq f(x^*) + \varepsilon$, as long as $s \geq s^* \max\{4\widetilde{\kappa} + 7, 12\widetilde{\kappa} + 6\}$. The number of iterations is $O\left(s \log^2 \frac{f(\vec{0})}{\varepsilon} \log\left(n \log \frac{f(\vec{0})}{\varepsilon}\right)\right)$.*

The following corollary that bounds the total runtime can be immediately extracted. Note that in practice the total runtime heavily depends on the choice of $f$, and it can often be improved for various special cases (e.g. linear regression).

**Corollary 6.3.2** (Theorem 6.3.1 runtime)**.** *If we denote by $G$ the time needed to compute $\nabla f$ and by $M$ the time to minimize $f$ in a restricted subset of $[n]$ of size $s$, the total runtime of Algorithm 15 is $O\left((G + M)s \log^2 \frac{f(\vec{0})}{\varepsilon} \log\left(n \log \frac{f(\vec{0})}{\varepsilon}\right)\right)$. If gradient descent is used for the implementation of the*

*inner optimization problem, then* $M = O\left(G\widetilde{\kappa}\log\frac{f(\vec{0})}{\varepsilon}\right)$ *and so the total runtime can be bounded by* $O\left(Gs\widetilde{\kappa}\log^3\frac{f(\vec{0})}{\varepsilon}\log\left(n\log\frac{f(\vec{0})}{\varepsilon}\right)\right)$.

Before proving the above theorem, we provide the main components that are needed for its proof. It is important to split the iterations of Algorithm 14 into two categories: Those that make enough progress, i.e. for which the condition in Line 19 of Algorithm 14 is *false*, and those that don't, i.e. for which the condition in Line 19 is *true*. We call the former *Type 1* iterations and the latter *Type 2* iterations. Intuitively, Type 1 iterations signify that $g(x)$ is decreasing at a sufficient rate to achieve the desired convergence, while Type 2 iterations indicate a local minimum that should be dealt with. Our argument consists of two steps: Showing that as long as there are enough Type 1 iterations, a desired solution will be obtained (Lemma 6.3.3), and bounding the total number of Type 2 iterations with constant probability (Lemma 6.3.4).

**Lemma 6.3.3** (Convergence rate)**.** *If Algorithm 14 executes at least* $T_1 = s\log\frac{g(x^0)-\widehat{f}^*}{\varepsilon}$ *Type 1 iterations, then* $f(x^T) \leq \widehat{f}^* + \varepsilon$.

The proof of this lemma can be found in Appendix 9.4.1.

**Lemma 6.3.4** (Bounding Type 2 iterations)**.** *If* $s \geq s^*\max\{4\widetilde{\kappa}+7, 12\widetilde{\kappa}+6\}$ *and* $\widehat{f}^* \geq f(x^*)$*, then with probability at least* 0.2 *the number of Type 2 iterations is at most* $(s^*-1)(4\widetilde{\kappa}+6)$.

The proof of this lemma appears in Section 6.3.2. These lemmas can now be directly used to obtain the following lemma, which states the performance guarantee of the ARHT core routine (Algorithm 14).

**Lemma 6.3.5** (Algorithm 14 guarantee)**.** *If* $s \geq s^*\max\{4\widetilde{\kappa}+7, 12\widetilde{\kappa}+6\}$ *and* $\widehat{f}^* \geq f(x^*)$*, with probability at least* 0.2 $\mathrm{ARHT\_core}(s, \widehat{f}^*, \varepsilon)$ *returns an* $s$*-sparse solution* $x$ *such that* $f(x) \leq \widehat{f}^* + \varepsilon$.

*Proof.* By Lemma 6.3.4, with probability at least 0.2 there will be at most $(s^*-1)(4\widetilde{\kappa}+6)$ Type 2 iterations. This means that the number of Type 1 iterations is at least

$$T - (s^*-1)(4\widetilde{\kappa}+6) \geq s\log\frac{f(\vec{0})}{\varepsilon} \geq s\log\frac{g^0(x^0)-\widehat{f}^*}{\varepsilon},$$

where the latter inequality follows from the fact that $f(\vec{0}) = g^0(\vec{0}) \geq g^0(x^0)$ and $\widehat{f}^* \geq f(x^*) \geq 0$. Lemma 6.3.3 then implies that $f(x^T) \leq \widehat{f}^* + \varepsilon$. □

In other words, as long as $\widehat{f}^* \geq f(x^*)$, a solution of value $\leq \widehat{f}^* + \varepsilon$ will be found. As the value $f(x^*)$ is not known a priori, $\widehat{f}^*$ is just an estimate for it. We perform binary search over $\widehat{f}^*$, as described in Algorithm 15. The reason we need this estimate of $f(x^*)$ is line 19 of Algorithm 14. As our algorithm is randomized, we use this estimate to decide whether there was enough progress in one iteration (Type 1 iteration), or not (Type 2 iteration, in which case we perform the randomization step). If $\widehat{f}^*$ is smaller than $f(x^*)$, the algorithm might mistake a Type 1 iteration for a Type 2 iteration, thus performing the randomization step even though the algorithm makes enough progress. On the other hand, if $\widehat{f}^*$ is much larger than $f(x^*)$, the algorithm might terminate with a suboptimal solution of value much larger than $f(x^*)$.

The probability of success in the previous lemma can be boosted by repeating multiple times. Combining these arguments will lead us to the proof of Theorem 6.3.1. First, we turn the result of Lemma 6.3.5 into a high probability result by repeating multiple times:

**Lemma 6.3.6.** *If $s \geq s^* \max\{4\widetilde{\kappa}+7, 12\widetilde{\kappa}+6\}$ and $\widehat{f}^* \geq f(x^*)$, $\text{ARHT\_robust}(s, \widehat{f}^*, \varepsilon)$ returns an $s$-sparse solution $x$ such that $f(x) \leq \widehat{f}^* + \varepsilon$ with probability at least $1 - \frac{1}{6n \log \frac{f(\vec{0})}{\varepsilon}}$.*

*Proof.* From Lemma 6.3.5, the probability that a given call to ARHT_core fails is at most 0.8. Since this random experiment is executed $5 \log\left(6n \log \frac{f(\vec{0})}{\varepsilon}\right)$ times independently, the probability that it never succeeds is at most $(0.8)^{5 \log\left(6n \log \frac{f(\vec{0})}{\varepsilon}\right)} < \frac{1}{6n \log \frac{f(\vec{0})}{\varepsilon}}$, therefore the statement follows. □

**Lemma 6.3.7.** *If $s \geq s^* \max\{4\widetilde{\kappa}+7, 12\widetilde{\kappa}+6\}$, $\text{ARHT}(s, \varepsilon)$ (in Algorithm 15) returns an $s$-sparse solution $x$ such that $f(x) \leq f(x^*) + \varepsilon$. The algorithm succeeds with probability at least $1 - \frac{1}{n}$. and the number of calls to* ARHT_robust *is $\leq 6 \log \frac{f(\vec{0})}{\varepsilon}$.*

*Proof.* First we will bound the number of calls to ARHT_robust. Let $L_k$ be the equal to $r - l$ before the $k$-th iteration in Line 21 of Algorithm 15. Then either $L_{k+1} = L_k/2$ (Line 25) or $L_{k+1} \leq L_k/2 + \varepsilon/3 < 5L_k/6$ (Line 28). Therefore in any case we have $L_{k+1} < 5L_k/6$ which implies that after $T = 6 \log \frac{f(\vec{0})}{\varepsilon}$ iterations we will have $r - l \leq \varepsilon$.

Now let us compute the probability that all the calls to ARHT_robust are successful. The number of such calls is at most $6 \log \frac{f(\vec{0})}{\varepsilon}$ and we know each one of them independently fails with probability less than $\frac{1}{6n \log \frac{f(\vec{0})}{\varepsilon}}$, so by a union bound the probability that at least one call fails is less than $\frac{1}{n}$.

To prove correctness, note that by Lemma 6.3.6, for each $r \geq f(x^*)$ we have $f(\text{ARHT\_robust}(s, r, \varepsilon/3)) \leq r + \varepsilon/3$. After Line 20 of Algorithm 15, we will have $l = 0 \leq f(x^*)$. In the while construct, it is always true that $f(x^*) \geq l$. This is initially true, as we saw. For each $m$ chosen in Line 22 and $x$ in Line 23, note that if $f(x) > m + \varepsilon/3$, then by Lemma 6.3.6 $f(x^*) > m$ and so the invariant that $f(x^*) \geq l$ stays true. On the other hand, it is always true that $f(b) \leq r$. Initially this is so because $f(\vec{0}) = r$, and when we decrease $r$ to some $f(x)$ we also update $b = x$. This implies that in the end of the algorithm the returned solution will have the required property, since we will have $f(b) \leq r \leq l + \varepsilon \leq f(x^*) + \varepsilon$. □

The proof Theorem 6.3.1 now easily follows.

**Proof of Theorem 6.3.1.** Lemma 6.3.7 already establishes the correctness of the algorithm with probability at least $1 - \frac{1}{n}$. For the runtime, note that ARHT_core takes $O\left(s \log \frac{f(\vec{0})}{\varepsilon}\right)$ iterations, ARHT_robust takes $O\left(\log\left(n \log \frac{f(\vec{0})}{\varepsilon}\right)\right)$ iterations, and ARHT takes $O\left(\log \frac{f(\vec{0})}{\varepsilon}\right)$ iterations. In conclusion, the total number of iterations is $O\left(s \log^2 \frac{f(\vec{0})}{\varepsilon} \log\left(n \log \frac{f(\vec{0})}{\varepsilon}\right)\right)$, each of which requires a constant number of minimizations of $f$. ■

### 6.3.2 Bounding Type 2 Iterations

When $x$ has significant $\ell_2^2$ mass in the target support, the regularization term $\frac{\rho_2^+}{2}\|x\|_2^2$ might penalize the target solution too much, leading to a Type 2 iteration. In this case, we use random sampling to detect an element in the optimal support and unregularize it. This procedure escapes all local minima, thus leading to a bound in the total number of Type 2 iterations.

More concretely, we show that if at some iteration of the algorithm the value of $g(x)$ does not decrease sufficiently (Type 2 iteration), then roughly at least a $\frac{1}{\widetilde{\kappa}}$-fraction of the $\ell_2^2$ mass of $x$ lies

in the target support $S^*$. We exploit this property by sampling an element $i$ proportional to $x_i^2$ and removing its corresponding term from the regularizer (*unregularizing* it). We show that with constant probability this will happen at most $O(s^*\widetilde{\kappa})$ times, as after that all the elements in $S^*$ will have been unregularized.

When referring to the $t$-th iteration of Algorithm 14, we let $x^t$ be the current solution with support set $S^t$ and $R^t \subseteq [n]$ the current regularization set as defined in the algorithm. For ease of notation, we will drop the subscript of the regularizer, i.e. $\Phi^t(z) := \frac{\rho_2^+}{2} \|z_{R^t}\|_2^2$ and of the regularized function, i.e. $g^t(z) := f(z) + \Phi^t(z)$. Note that by definition of the algorithm $x^t$ is an $S^t$-restricted minimizer of $g^t$.

Let $(\rho_2^+)'$ and $(\rho_{s+s^*}^-)'$ be RSS and RSC parameters of $g^t$. We start with a lemma that relates $(\rho_2^+)'$ to $\rho_2^+$ and $(\rho_{s+s^*}^-)'$ to $\rho_{s+s^*}^-$, and is proved in Appendix 9.4.1.

**Lemma 6.3.8** (RSC, RSS of regularized function)**.** *$(\rho_2^+)' \le 2\rho_2^+$ and $(\rho_{s+s^*}^-)' \ge \rho_{s+s^*}^-$*

This states that the restricted smoothness and strong convexity constants of the regularized function are always within a constant factor of those of the original function, and thus we can make our statements in terms of the RSC, RSS of the original function. Next, we present a lemma that establishes a lower bound on the progress $g^t(x^t) - g^{t+1}(x^{t+1})$ in one iteration. This will be helpful in order to diagnose the cause of having insufficient progress in one iteration.

**Lemma 6.3.9** (ARHT Progress Lemma)**.** *If $\widehat{f^*} \ge f(x^*)$, for the progress $g^t(x^t) - g^t(x^{t+1})$ in Line 19 of Algorithm 14 it holds that*

$$g^t(x^t) - g^t(x^{t+1})$$
$$\ge \frac{\rho^-}{2|S^*\backslash S^t|\rho^+}\Big(f(x^t) - f(x^*) + \langle \nabla_{S^t\backslash S^*}\Phi^t(x^t), x^t_{S^t\backslash S^*}\rangle - \frac{1}{2\rho^-}\left\|\nabla_{S^t\cap S^*}\Phi^t(x^t)\right\|_2^2\Big) - \rho^+(x_j^t)^2\,.$$

*Proof.* The proof will proceed as follows: We first use the smoothness of $g^t$ to get a lower bound on the progress in one step, $g^t(x^t) - g^t(x^{t+1})$. This lower bound will depend on $\|\nabla_{S^*\backslash S^t} g^t(x^t)\|_2^2$, which is the norm of the gradient of $g^t$ restricted to the set $S^*\backslash S^t$, as well as $(x_j^t)^2$, where $j$ is the position of the minimum-magnitude entry of $x^t$. Then, we use the strong convexity of $g^t$ to relate $\|\nabla_{S^*\backslash S^t} g^t(x^t)\|_2^2$ to the difference in function value $f(x^t) - f(x^*)$, plus some terms that come from the regularizer.

First of all, since the condition in Line 12 ("if $\min_{\mathrm{supp}(x)\subseteq S^t} f(x) \le \widehat{f^*}$") was not triggered, we have that $\min_{\mathrm{supp}(x)\subseteq S^t} f(x) > \widehat{f^*} \ge f(x^*)$ and so $S^*\backslash S^t \neq \emptyset$. By Lemma 6.3.8 we have that $(\rho^+)' \le 2\rho^+$, therefore the decrease in $g^t$ that is achieved is

$$\begin{aligned} &g^t(x^t) - g^t(x^{t+1}) \\ &\ge \max_{\eta\in\mathbb{R}} \left\{g^t(x^t) - g^t(x^t + \eta\vec{1}_i - x_j^t\vec{1}_j)\right\} \\ &\ge \max_{\eta\in\mathbb{R}} \left\{-\langle\nabla g^t(x^t), \eta\vec{1}_i - x_j^t\vec{1}_j\rangle - \rho^+\eta^2 - \rho^+(x_j^t)^2\right\} := B\,. \end{aligned}$$

Note that, as defined by the algorithm, $x^t$ is an $S^t$-restricted minimizer of $g^t$ and since $j \in S^t$, we

have $\nabla_j g^t(x^t) = 0$. Therefore

$$\begin{aligned} B =& \max_{\eta \in \mathbb{R}} \{ -\langle \nabla g^t(x^t), \eta \vec{1}_i \rangle - \rho^+ \eta^2 - \rho^+ (x_j^t)^2 \} \\ =& \frac{\left[\nabla_i g^t(x^t)\right]^2}{4\rho^+} - \rho^+ (x_j^t)^2 \\ \geq& \max_{k \in S^* \setminus S} \frac{\left[\nabla_k g^t(x^t)\right]^2}{4\rho^+} - \rho^+ (x_j^t)^2 \\ \geq& \frac{\left\| \nabla_{S^* \setminus S^t} g^t(x^t) \right\|_2^2}{4|S^* \setminus S^t| \rho^+} - \rho^+ (x_j^t)^2 , \end{aligned} \tag{6.2}$$

where we used the fact that $i$ was picked to maximize $\left|\nabla_k g^t(x^t)\right|$. Now we would like to relate this to $g^t(x^t) - f(x^*)$ (and not $g^t(x^t) - g^t(x^*)$). By applying the Restricted Strong Convexity property,

$$\begin{aligned} & f(x^*) - f(x^t) \\ & \geq \langle \nabla f(x^t), x^* - x^t \rangle + \frac{\rho^-}{2} \left\| x^t - x^* \right\|_2^2 \\ & \geq \langle \nabla f(x^t), x^* - x^t \rangle + \frac{\rho^-}{2} \left\| x^*_{S^* \setminus S^t} \right\|_2^2 + \frac{\rho^-}{2} \left\| (x^t - x^*)_{S^t \cap S^*} \right\|_2^2 . \end{aligned}$$

Now note that $f(x^t) = g^t(x^t) - \Phi^t(x^t)$, $\nabla_{S^t} g^t(x^t) = \vec{0}$ (since $x^t$ is an $S^t$-restricted minimizer of $g^t$), and $\nabla \Phi^t(x^t) = \nabla_{S^t} \Phi^t(x^t)$ therefore

$$\begin{aligned} & \langle \nabla f(x^t), x^* - x^t \rangle \\ & = \langle \nabla g^t(x^t), x^* - x^t \rangle - \langle \nabla \Phi^t(x^t), x^* - x^t \rangle \\ & = \langle \nabla g^t_{S^* \setminus S^t}(x^t), x^*_{S^* \setminus S^t} \rangle + \langle \nabla_{S^t \setminus S^*} \Phi^t(x^t), x^t_{S^t \setminus S^*} \rangle + \langle \nabla_{S^t \cap S^*} \Phi^t(x^t), (x^t - x^*)_{S^t \cap S^*} \rangle . \end{aligned}$$

Plugging this into the previous inequality, we get

$$\begin{aligned} & f(x^*) - f(x^t) \\ & \geq \langle \nabla g^t_{S^* \setminus S^t}(x^t), x^*_{S^* \setminus S^t} \rangle + \frac{\rho^-}{2} \left\| x^*_{S^* \setminus S^t} \right\|_2^2 + \langle \nabla_{S^t \setminus S^*} \Phi^t(x^t), x^t_{S^t \setminus S^*} \rangle \\ & + \langle \nabla_{S^t \cap S^*} \Phi^t(x^t), (x^t - x^*)_{S^t \cap S^*} \rangle + \frac{\rho^-}{2} \left\| (x^t - x^*)_{S^t \cap S^*} \right\|_2^2 \\ & \geq -\frac{1}{2\rho^-} \left\| \nabla_{S^* \setminus S^t} g^t(x^t) \right\|_2^2 + \langle \nabla_{S^t \setminus S^*} \Phi^t(x^t), x^t_{S^t \setminus S^*} \rangle - \frac{1}{2\rho^-} \left\| \nabla_{S^t \cap S^*} \Phi^t(x^t) \right\|_2^2 , \end{aligned}$$

where we twice used the inequality $\langle u, v \rangle + \frac{\lambda}{2} \|v\|_2^2 \geq -\frac{1}{2\lambda} \|u\|_2^2$ for any $\lambda > 0$. This inequality is derived by expanding $\frac{1}{2} \left\| \frac{1}{\sqrt{\lambda}} u + \sqrt{\lambda} v \right\|_2^2 \geq 0$. So plugging in $\left\| \nabla_{S^* \setminus S^t} g^t(x^t) \right\|_2^2$ into (6.2),

$$\begin{aligned} & B \\ & \geq \frac{\rho^-}{2|S^* \setminus S^t| \rho^+} \Big( f(x^t) - f(x^*) + \langle \nabla_{S^t \setminus S^*} \Phi^t(x^t), x^t_{S^t \setminus S^*} \rangle - \frac{1}{2\rho^-} \left\| \nabla_{S^t \cap S^*} \Phi^t(x^t) \right\|_2^2 \Big) - \rho^+ (x_j^t)^2 . \end{aligned}$$

□

Let $R \subseteq [n]$ be the set of currently regularized elements. The following invariant is a crucial

ingredient for bringing the sparsity from $O(s^*\widetilde{\kappa}^2)$ down to $O(s^*\widetilde{\kappa})$, and we intend to enforce it at all times. It essentially states that there will always be enough elements in the current solution that are being regularized.

**Invariant 6.3.10.**

$$|R \cap S| \geq s^* \max\{1, 8\widetilde{\kappa}\}$$

To give some intuition on this, ARHT owes its improved $\widetilde{\kappa}$ dependence on the regularizer $\frac{\rho^+}{2}\|x\|_2^2$. However, during the algorithm, some elements are being unregularized. Our analysis requires that the current solution support always contains $\Omega(s^*\widetilde{\kappa})$ regularized elements, which is what Invariant 6.3.10 states.

We can now proceed to show that, with constant probability, Algorithm 14 will only have $O(s^*\widetilde{\kappa})$ Type 2 iterations, which is the goal of this section.

**Proof of Lemma 6.3.4.** The idea of the proof is to use the progress bound in Lemma 6.3.9 to obtain a necessary condition under which the progress (i.e. the decrease of $g^t(x^t)$) is not sufficient in one iteration (thus, we have a Type 2 iteration). For our choice of regularizer, this condition implies that at least an $\Omega(\frac{1}{\widetilde{\kappa}})$ fraction of the $\ell_2^2$ mass of $x^t$ lies in the optimal support set $S^*$, which means that we can find an element in $S^*$ with decent probability by appropriately sampling an element of $x^t$. We finally apply a probabilistic analysis over all iterations, to show that if each sampled element is unregularized, with constant probability the total number of Type 2 iterations cannot exceed $\Theta(\widetilde{\kappa}s^*)$.

We first observe some useful properties of our regularizer, which can be verified by simple substitution. The definition of $\Phi^t(x^t)$ implies that

$$\begin{aligned}
\langle \nabla_{S^t\setminus S^*}\Phi^t(x^t), x^t_{S^t\setminus S^*}\rangle &= \rho^+\langle x^t_{(R^t\cap S^t)\setminus S^*}, x^t_{S^t\setminus S^*}\rangle \\
&= \rho^+ \sum_{i\in(R^t\cap S^t)\setminus S^*} x_i^2 \\
&= \rho^+ \sum_{i\in R^t\setminus S^*} x_i^2 \\
&= \rho^+ \left\| x^t_{R^t\setminus S^*}\right\|_2^2 ,
\end{aligned} \tag{6.3}$$

where the second-to-last equality follows because $x^t$ is 0 outside of $S^t$. For the same reason, we also have

$$\left\|\nabla_{S^t\cap S^*}\Phi^t(x^t)\right\|_2^2 = (\rho^+)^2 \left\|x^t_{R^t\cap S^*}\right\|_2^2 . \tag{6.4}$$

By combining (6.3) and (6.4) we get that

$$\Phi^t(x^t) = \frac{1}{2}\langle \nabla_{S^t\setminus S^*}\Phi^t(x^t), x^t_{S^t\setminus S^*}\rangle + \frac{1}{2\rho^+}\left\|\nabla_{S^t\cap S^*}\Phi^t(x^t)\right\|_2^2 . \tag{6.5}$$

Equations (6.3),(6.4), and (6.5) will be used later on. Now, before the first iteration we have $\left|R^0\cap S^0\right| = \left|S^0\right| = s$. Since in each Type 2 iteration we have $\left|R^{t+1}\right| = \left|R^t\right| - 1$,

$$\left|R^t\cap S^t\right| \geq s - [\text{number of Type 2 iterations up to } t] .$$

This implies that for the first $(s^*-1)(4\widetilde{\kappa}+6)$ Type 2 iterations,

$$|R^t \cap S^t| \geq s - (s^*-1)(4\widetilde{\kappa}+6) \geq s^* \max\{1, 8\widetilde{\kappa}\}\,, \tag{6.6}$$

since $s \geq s^* \max\{4\widetilde{\kappa}+7, 12\widetilde{\kappa}+6\}$. From this it follows that

$$\begin{aligned} |(R^t \cap S^t)\backslash S^*| &= |R^t \cap S^t| - |R^t \cap S^t \cap S^*| \\ &\geq s^* \max\{1, 8\widetilde{\kappa}\} - |S^t \cap S^*| \\ &\geq |S^* \backslash S^t| 8\widetilde{\kappa} \\ &= |S^* \backslash S^t| 8 \frac{\rho^+}{\rho^-} \end{aligned}$$

and so

$$\begin{aligned} (x_j^t)^2 &\leq \frac{1}{|(R^t \cap S^t)\backslash S^*|} \left\| x^t_{(R^t \cap S^t)\backslash S^*} \right\|_2^2 \\ &\leq \frac{\rho^-}{8|S^* \backslash S^t|\rho^+} \left\| x^t_{R^t \backslash S^*} \right\|_2^2 \\ &= \frac{\rho^-}{8|S^* \backslash S^t|(\rho^+)^2} \langle \nabla_{S^t \backslash S^*} \Phi^t(x^t), x^t_{S^t \backslash S^*} \rangle\,, \end{aligned}$$

where $j \in S^t$ is the element that the algorithm removes from $S^t$, and we used (6.3). Combining this inequality with the statement of Lemma 6.3.9 we have

$$\begin{aligned} & g^t(x^t) - g^t(x^{t+1}) \\ & \geq \frac{\rho^-}{2|S^* \backslash S^t|\rho^+} \Big( f(x^t) - f(x^*) + \langle \nabla_{S^t \backslash S^*} \Phi^t(x^t), x^t_{S^t \backslash S^*} \rangle - \frac{1}{2\rho^-} \left\| \nabla_{S^t \cap S^*} \Phi^t(x^t) \right\|_2^2 \Big) - \rho^+ (x_j^t)^2 \\ & \geq \frac{\rho^-}{2|S^* \backslash S^t|\rho^+} \Big( f(x^t) - f(x^*) + \frac{3}{4} \langle \nabla_{S^t \backslash S^*} \Phi^t(x^t), x^t_{S^t \backslash S^*} \rangle - \frac{1}{2\rho^-} \left\| \nabla_{S^t \cap S^*} \Phi^t(x^t) \right\|_2^2 \Big)\,. \end{aligned} \tag{6.7}$$

By definition of a Type 2 iteration,

$$\begin{aligned} g^t(x^t) - g^t(x^{t+1}) &< \frac{1}{s} \left( g^t(x^t) - \widehat{f}^* \right) \\ &\leq \frac{\rho^-}{2|S^* \backslash S^t|\rho^+} \left( g^t(x^t) - f(x^*) \right) \\ &= \frac{\rho^-}{2|S^* \backslash S^t|\rho^+} \left( f(x^t) - f(x^*) + \Phi^t(x^t) \right) \end{aligned} \quad , \tag{6.8}$$

where we used the fact that $s \geq 2s^*\widetilde{\kappa} \geq 2|S^* \backslash S^t|\widetilde{\kappa}$ and $f(x^*) \leq \widehat{f}^*$. Combining inequalities (6.7) and (6.8) we get

$$\Phi^t(x^t) > \frac{3}{4} \langle \nabla_{S^t \backslash S^*} \Phi^t(x^t), x^t_{S^t \backslash S^*} \rangle - \frac{1}{2\rho^-} \left\| \nabla_{S^t \cap S^*} \Phi^t(x^t) \right\|_2^2\,,$$

or equivalently, by replacing $\Phi^t(x^t)$ from (6.5),

$$\frac{1}{2} \left( \frac{1}{\rho^-} + \frac{1}{\rho^+} \right) \left\| \nabla_{S^t \cap S^*} \Phi^t(x^t) \right\|_2^2 > \frac{1}{4} \langle \nabla_{S^t \backslash S^*} \Phi(x^t), x^t_{S^t \backslash S^*} \rangle\,.$$

Further applying (6.3) and (6.4), we equivalently get

$$2\left(1+\widetilde{\kappa}\right)\left\|x^t_{R^t\cap S^*}\right\|_2^2 > \left\|x^t_{R^t\setminus S^*}\right\|_2^2 . \tag{6.9}$$

Now, note that in Lines 21-22 the algorithm picks an element $i \in R^t$ with probability proportional to $(x^t_i)^2$ and unregularizes it, i.e. sets $R^{t+1} \leftarrow R^t \backslash \{i\}$. We denote this probability distribution over $i \in R^t$ by $\mathcal{D}$. From what we have established already in (6.9), we can lower bound the probability that $i$ lies in the target support:

$$\begin{aligned}
\Pr_{i\sim\mathcal{D}}[i \in S^*] &= \frac{\left\|x^t_{R^t\cap S^*}\right\|_2^2}{\left\|x^t_{R^t\cap S^*}\right\|_2^2 + \left\|x^t_{R^t\setminus S^*}\right\|_2^2} \\
&> \frac{\frac{1}{2(1+\widetilde{\kappa})}}{1+\frac{1}{2(1+\widetilde{\kappa})}} \\
&= \frac{1}{2\widetilde{\kappa}+3} \\
&:= p\,.
\end{aligned} \tag{6.10}$$

Note that this event can happen at most once for each $i \in S^*$ during the whole execution of the algorithm, since each element can only be removed once from the set of regularized elements.

We will prove that with constant probability the number of Type 2 steps will be at most $(s^*-1)(4\widetilde{\kappa}+6) := b$. For $1 \le k \le b$, we define the following random variables:

- $i_k \in [n]$ is the index picked in the $k$-th Type 2 iteration, or $\perp$ if there are less than $k$ Type 2 iterations.

- $q_k$ is the probability of picking an index in the optimal support in the $k$-th Type 2 iteration (i.e. $i_k \in S^*$):

$$q_k = \begin{cases} \left\|x^{t_k}_{R^{t_k}\cap S^*}\right\|_2^2 / \left\|x^{t_k}_{R^{t_k}}\right\|_2^2 & \text{if } i_k \neq \perp \\ 0 & \text{otherwise} \end{cases},$$

  where $t_k \in [T]$ is the index of the $k$-th Type 2 iteration within all iterations of the algorithm. Note that, by (6.10), $q_k > 0$ implies $q_k \ge p$.

- $X_k$ is 1 if the index picked in the $k$-th Type 2 step was in the optimal support:

$$X_k = \begin{cases} 1 & \text{with probability } q_k \\ 0 & \text{otherwise} \end{cases}$$

Our goal is to upper bound $\Pr\left[\sum_{k=1}^{b} X_k \le s^*-1\right]$. This automatically implies the same upper bound on the probability that there will be more than $b$ Type 2 iterations.

We define another sequence of random variables $Y_0, \ldots, Y_b$, where $Y_0 = 0$, and

$$Y_k = \begin{cases} Y_{k-1} + \frac{p}{q_k} - p & \text{if } X_k = 1 \\ Y_{k-1} - p & \text{if } X_k = 0 \end{cases},$$

for $k \in [b]$. Since if $q_k > 0$ we have $\frac{p}{q_k} \leq 1$, it is immediate that

$$Y_k - Y_{k-1} \leq X_k - p$$

and so $Y_b \leq \sum_{k=1}^{b} X_k - bp$. Furthermore,

$$\begin{aligned}\mathbb{E}\left[Y_k \mid i_1, \ldots, i_{k-1}\right] &= Y_{k-1} + q_k \left(\frac{p}{q_k} - p\right) - (1 - q_k)\, p \\ &= Y_{k-1}\,,\end{aligned}$$

meaning that $Y_0, \ldots, Y_b$ is a martingale with respect to $i_1, \ldots, i_b$. We will apply the inequality from Lemma 6.2.4. We compute a bound on the differences

$$\begin{aligned}Y_{k-1} - Y_k &= \begin{cases} p - \frac{p}{q_k} & \text{if } X_k = 1 \\ p & \text{if } X_k = 0 \end{cases} \\ &\leq p\end{aligned} \tag{6.11}$$

and the variance

$$\begin{aligned}\mathrm{Var}\left(Y_k \mid i_1, \ldots, i_{k-1}\right) &= \mathbb{E}\left[\left(Y_k - \mathbb{E}\left[Y_k \mid i_1, \ldots, i_{k-1}\right]\right)^2 \mid i_1, \ldots, i_{k-1}\right] \\ &= \mathbb{E}\left[\left(Y_k - Y_{k-1}\right)^2 \mid i_1, \ldots, i_{k-1}\right] \\ &= q_k \cdot \left(p - \frac{p}{q_k}\right)^2 + (1 - q_k) \cdot p^2 \\ &= q_k \cdot p^2 \left(1 - \frac{2}{q_k} + \frac{1}{q_k^2}\right) + (1 - q_k) \cdot p^2 \\ &= p^2 \left(\frac{1}{q_k} - 1\right) \\ &\leq p\,,\end{aligned}$$

where we used (6.11) along with the fact that $q_k \geq p$. Using the concentration inequality from Lemma 6.2.4 we obtain

$$\begin{aligned}\Pr\left[\sum_{k=1}^{b} X_k \leq s^* - 1\right] &\leq \Pr\left[Y_b \leq s^* - 1 - b \cdot p\right] \\ &\leq e^{-(bp - s^* + 1)^2 / (2(b \cdot p + p \cdot (bp - s^* + 1)/3))} \\ &= e^{-(s^* - 1)/(2(2 + p/3))} \\ &\leq e^{-1/(2(2 + 1/9))} \\ &< 0.8\,,\end{aligned}$$

where we used the fact that $bp = 2(s^* - 1)$, $s^* \geq 2$ (otherwise the problem is trivial), and $p = \frac{1}{2\widetilde{\kappa} + 3} \leq \frac{1}{3}$. Therefore we conclude that the probability that we have not unregularized the whole set $S^*$ after $b$ steps is at most 0.8. Since we can only have a Type 2 step if there is a regularized element in $S^*$ (this is immediate e.g. from (6.10)), this implies that with probability at least 0.2 the number of Type 2 steps is at most $b = (s^* - 1)(4\widetilde{\kappa} + 6)$.

■

### 6.3.3 Corollaries

As the first corollary of Theorem 6.3.1, we show that it directly implies solution recovery bounds similar to those of [170], while also improving the recovery bound by a constant factor.

**Corollary 6.3.11** (Solution recovery)**.** *Given a function $f$ and an (unknown) $s^*$-sparse solution $x^*$, such that the Restricted Gradient Optimal Constant at sparsity level $s$ is $\zeta$, i.e.*

$$|\langle \nabla f(x^*), y \rangle| \leq \zeta \|y\|_2 ,$$

*for all $s$-sparse $y$ and as long as*

$$s \geq s^* \max\{4\widetilde{\kappa} + 7, 12\widetilde{\kappa} + 6\} ,$$

*Algorithm 15 ensures that*

$$f(x) \leq f(x^*) + \varepsilon$$

*and*

$$\|x - x^*\|_2 \leq \frac{\zeta}{\rho^-} \left(1 + \sqrt{1 + 2\varepsilon \frac{\rho^-}{\zeta^2}}\right) .$$

*For any $\theta > 0$ and $\varepsilon \leq \frac{\zeta^2}{\rho^-}\theta(1 + \frac{\theta}{2})$, this implies that*

$$\|x - x^*\|_2 \leq (2 + \theta)\frac{\zeta}{\rho^-} .$$

*Proof.* By strong convexity we have

$$\begin{aligned} \varepsilon &\geq f(x) - f(x^*) \\ &\geq \langle \nabla f(x^*), x - x^* \rangle + \frac{\rho^-}{2} \|x - x^*\|_2^2 \\ &\geq -\zeta \|x - x^*\|_2 + \frac{\rho^-}{2} \|x - x^*\|_2^2 , \end{aligned}$$

therefore

$$\frac{\rho^-}{2} \|x - x^*\|_2^2 - \zeta \|x - x^*\|_2 - \varepsilon \leq 0 ,$$

looking at which as a quadratic polynomial in $\|x - x^*\|_2$, it follows that

$$\begin{aligned}\|x - x^*\|_2 &\leq \frac{\zeta + \sqrt{\zeta^2 + 2\varepsilon\rho^-}}{\rho^-} \\ &= \frac{\zeta}{\rho^-}\left(1 + \sqrt{1 + 2\varepsilon\frac{\rho^-}{\zeta^2}}\right) \\ &= (2+\theta)\frac{\zeta}{\rho^-}\,,\end{aligned}$$

by setting $\varepsilon = \frac{\zeta^2}{\rho^-}\left(\theta + \frac{1}{2}\theta^2\right)$. □

The next corollary shows that our Theorem 6.3.1 can be also used to obtain support recovery results under a "Signal-to-Noise" condition given as a lower bound to $|x^*_{\min}|$.

**Corollary 6.3.12** (Support recovery)**.** *As long as*

$$s \geq s^* \max\left\{4\widetilde{\kappa} + 7, 12\widetilde{\kappa} + 6\right\}$$

*and* $|x^*_{\min}| > \frac{\zeta}{\rho^-}$*, Algorithm 15 with* $\varepsilon < -\frac{1}{2\rho^-}\zeta^2 + \frac{\rho^-}{2}(x^*_{\min})^2$ *returns a solution* $x$ *with support* $S$ *such that*

$$S^* \subseteq S\,.$$

*Proof.* Let us suppose that $S^* \backslash S^t \neq \emptyset$. By restricted strong convexity we have

$$\begin{aligned}-\frac{1}{2\rho^-}\zeta^2 + \frac{\rho^-}{2}\left(x^*_{\min}\right)^2 &> \varepsilon \\ &\geq f(x) - f(x^*) \\ &\geq \langle \nabla f(x^*), x - x^*\rangle + \frac{\rho^-}{2}\|x - x^*\|_2^2 \\ &\geq \langle \nabla f(x^*), x\rangle + \frac{\rho^-}{2}\left\|x_{S^t\backslash S^*}\right\|_2^2 + \frac{\rho^-}{2}\left\|x^*_{S^*\backslash S^t}\right\|_2^2 \\ &\geq -\frac{1}{2\rho^-}\left\|\nabla_{S^t\backslash S^*} f(x^*)\right\|_2^2 + \frac{\rho^-}{2}\left\|x^*_{S^*\backslash S^t}\right\|_2^2 \\ &\geq -\frac{1}{2\rho^-}\zeta^2 + \frac{\rho^-}{2}\left(x^*_{\min}\right)^2\end{aligned}$$

a contradiction. Here we used the fact that by local optimality $\nabla_{S^*} f(x^*) = \vec{0}$, the inequality $\langle u, v\rangle + \frac{\lambda}{2}\|v\|_2^2 \geq -\frac{1}{2\lambda}\|u\|_2^2$ for any vectors $u, v$ and scalar $\lambda > 0$, and the fact that $\left\|\nabla_{S^t\backslash S^*} f(x^*)\right\|_2^2 \leq \zeta^2$ by Definition 6.2.2. Therefore $S^* \subseteq S^t$. □

## 6.4 Analysis of Orthogonal Matching Pursuit with Replacement (OMPR)

### 6.4.1 Overview and Main Theorem

The OMPR algorithm was first described (under a different name) in [146]. It is an extension of OMP but after each iteration some element is removed from $S^t$ so that the sparsity remains the

same. The algorithm description is in Algorithm 12.

For each iteration $t$ of Algorithm 12, we will define a solution

$$\widetilde{x}^t = \underset{\text{supp}(x)\subseteq S^t\cup S^*}{\text{argmin}} f(x)$$

to be the optimal solution supported on $S^t \cup S^*$. Furthermore, we let $\overline{x}^*$ be the optimal $(s+s^*)$-sparse solution, i.e.

$$\overline{x}^* = \underset{|\text{supp}(x)|\leq s+s^*}{\text{argmin}} f(x)\,.$$

By definition, the following chain of inequalities holds

$$\min_{x\in\mathbb{R}^n} f(x) \leq f(\overline{x}^*) \leq f(\widetilde{x}^t) \leq \min\{f(x^t), f(x^*)\}\,.$$

We will denote $\mu = \sqrt{\frac{s^*}{s}}$. The following technical lemma is important for our approach, and roughly states that if there is significant $\ell_2$ norm difference between $x^t$ and $x^*$, at least one of $x^t, x^*$ is significantly larger than $\widetilde{x}^t$ in function value. Its importance lies on the fact that instead of directly applying strong convexity between $x^t$ and $x^*$, it gets a tighter bound by making use of $\widetilde{x}^t$.

**Lemma 6.4.1.** *For any function $f$ with RSC constant $\rho^-$ at sparsity level $s+s^*$ and any two solutions $x^t$,$x^*$ with respective supports $S^t$, $S^*$ and sparsity levels $s$, $s^*$, we have that*

$$\left(\sqrt{f(x^t)-f(\widetilde{x}^t)}+\sqrt{f(x^*)-f(\widetilde{x}^t)}\right)^2 \geq \frac{\rho^-}{2}\left(\left\|x^*_{S^*\setminus S^t}\right\|_2^2 + \left\|x^t_{S^t\setminus S^*}\right\|_2^2\right)\,.$$

The proof can be found in Appendix 9.4.1.

The following theorem is the main result of this section. Its strength lies in its generality, and various useful corollaries can be directly extracted from it. It can be seen as a careful and general analysis of OMPR, and, in contrast to Theorem 6.3.1, the interesting part is not the asymptotic sparsity bound (which is $O(\kappa^2 s^*)$ and is known), but the constant factor in front of it, which allows its use in recovering a solution with sparsity close to $s^*$, under a RIP bound. It roughly states that for any solution $x^*$ that is $s^*$-sparse, OMPR can be used to obtain a solution $x$

- with sparsity $\leq \kappa^2 s^*$ that comes arbitrarily close to $x^*$ in function value, i.e. $f(x) \leq f(x^*)+\varepsilon$
- with sparsity $\leq \kappa^2 s^*/4$ that approximates $x^*$ in function value, and the approximation depends on how close $x^*$ is to being a global optimum.

The latter can be used to obtain compressed sensing results, as it gives sparsity very close to $s^*$ given that upper bounds on $\kappa$ (equivalently, RIP upper bounds) are met. In comparison with previously known results, our work is the first to obtain compressed sensing RIP bounds for general functions $f$ and for a wide range of sparsity levels from $s^*$ to much larger than that.

**Theorem 6.4.2.** *Given a function $f$, an (unknown) $s^*$-sparse solution $x^*$, a desired solution sparsity level $s$, and error parameters $\varepsilon > 0$ and $0 < \theta < 1$, Algorithm 12 returns an $s$-sparse solution $x$ such that*

- *If $\widetilde{\kappa}\sqrt{\frac{s^*}{s}} \leq 1$, then*

$$f(x) \leq f(x^*)+\varepsilon$$

*in* $O\left(\sqrt{ss^*}\log\frac{f(x^0)-f(x^*)}{\varepsilon}\right)$ *iterations.*

- *If* $1 < \widetilde{\kappa}\sqrt{\frac{s^*}{s}} < 2-\theta$, *then*

$$f(x) \le f(x^*) + B$$

*where*

$$B = \varepsilon + \frac{4(1-\theta)\left(\widetilde{\kappa}\sqrt{\frac{s^*}{s}}-1\right)}{\left(2-\widetilde{\kappa}\sqrt{\frac{s^*}{s}}-\theta\right)^2}(f(x^*)-f(\overline{x}^*))$$

*in* $O\left(\frac{\sqrt{ss^*}}{\theta}\log\frac{f(x^0)-f(x^*)}{B}\right)$ *iterations.*

### 6.4.2 Progress Lemma and Theorem Proof

The main ingredient needed to prove Theorem 6.4.2 is the following lemma, which bounds the progress of Algorithm 12 in one iteration.

**Lemma 6.4.3** (OMPR Progress Lemma)**.** *We can bound the progress of one step of the algorithm by distinguishing the following three cases:*

- *If* $\mu\widetilde{\kappa} \le 1$, *then*

$$f(x^{t+1}) - f(x^*) \le \left(f(x^t)-f(x^*)\right)\left(1-\frac{\mu}{|S^*\backslash S^t|}\right)$$

- *If* $\mu\widetilde{\kappa} > 1$ *and* $f(x^*) = f(\widetilde{x}^t)$, *then*

$$f(x^{t+1}) - f(x^*) \le \left(f(x^t)-f(x^*)\right)\cdot\left(1-\frac{\mu}{|S^*\backslash S^t|}\left(2-\mu\widetilde{\kappa}\right)\right)$$

- *If* $\mu\widetilde{\kappa} > 1$ *and* $f(x^*) > f(\widetilde{x}^t)$, *then*

$$f(x^{t+1}) - f(x^*) \le \left(f(x^t)-f(x^*)\right)\cdot\left(1-\frac{\mu}{|S^*\backslash S^t|}\left(2-\mu\widetilde{\kappa}-\frac{2(\mu\widetilde{\kappa}-1)}{\sqrt{\frac{f(x^t)-f(\widetilde{x}^t)}{f(x^*)-f(\widetilde{x}^t)}}-1}\right)\right)$$

*Proof.* The proof will proceed as follows: We will use the smoothness and strong convexity of $f$ to get a bound on the progress of one step of the algorithm in decreasing $f$, based on $f(x^t)-f(x^*)$. This progress will be offset by the $\ell_2$ norm of $x^*$ restricted to $S^*\backslash S^t$. We use Lemma 6.4.1 to upper bound this norm by a quantity that depends on $f(x^t)-f(\widetilde{x}^t)$ and $f(x^*)-f(\widetilde{x}^t)$, where $\widetilde{x}^t$ is the optimal solution in the joint support $S^t \cup S^*$. Finally, by a careful case analysis based on the value of $\mu$, we obtain the three bullet points in the lemma statement.

First of all, if $S^* \subseteq S^t$ then, since $x^t$ is an $S^t$-restricted minimizer, we have $f(x^T) \le f(x^t) \le f(x^*)$ and we are done. So suppose otherwise, i.e. $S^*\backslash S^t \neq \emptyset$ and $f(x^t) > f(x^*)$. Let $i = \underset{i\notin S^t}{\operatorname{argmax}} \left|\nabla_i f(x^t)\right|$ and $j = \underset{j\in S^t}{\operatorname{argmin}} \left|x^t_j\right|$. By definition of OMPR (Algorithm 12) and restricted smoothness of $f$, we

have

$$\begin{aligned}
f(x^{t+1}) &\leq \min_{\eta\in\mathbb{R}} f(x^t + \eta\vec{1}_i - x_j^t\vec{1}_j) \\
&\leq \min_{\eta\in\mathbb{R}} f(x^t) + \langle \nabla f(x^t), \eta\vec{1}_i - x_j^t\vec{1}_j\rangle + \frac{\rho^+}{2}\left\|\eta\vec{1}_i - x_j^t\vec{1}_j\right\|_2^2 \\
&= \min_{\eta\in\mathbb{R}} f(x^t) + \eta\nabla_i f(x^t) + \frac{\rho^+}{2}\eta^2 + \frac{\rho^+}{2}(x_j^t)^2 \\
&= f(x^t) - \frac{\left(\nabla_i f(x^t)\right)^2}{2\rho^+} + \frac{\rho^+}{2}(x_j^t)^2 \\
&\leq f(x^t) - \frac{\left\|\nabla_{S^*\setminus S^t} f(x^t)\right\|_2^2}{2\rho^+|S^*\setminus S^t|} + \frac{\rho^+}{2|S^t\setminus S^*|}\left\|x^t_{S^t\setminus S^*}\right\|_2^2,
\end{aligned} \tag{6.12}$$

where the second to last equality follows from the fact that $\nabla_j f(x^t) = \vec{0}$, as $x^t$ is an $S^t$-restricted minimizer of $f$, and the last inequality since

$$(x_j^t)^2 = \min_{j\in S^t\setminus S^*} (x_j^t)^2 \leq \frac{\left\|x^t_{S^t\setminus S^*}\right\|_2^2}{|S^t\setminus S^*|}.$$

Re-arranging (6.12), we get

$$|S^*\setminus S^t|(f(x^t) - f(x^{t+1})) \geq \frac{\left\|\nabla_{S^*\setminus S^t} f(x^t)\right\|_2^2}{2\rho^+} - \frac{\rho^+}{2}\frac{|S^*\setminus S^t|}{|S^t\setminus S^*|}\left\|x^t_{S^t\setminus S^*}\right\|_2^2. \tag{6.13}$$

On the other hand, by restricted strong convexity of $f$,

$$\begin{aligned}
f(x^*) - f(x^t) &\geq \langle \nabla f(x^t), x^* - x^t\rangle + \frac{\rho^-}{2}\left\|x^* - x^t\right\|_2^2 \\
&= \langle \nabla_{S^*\setminus S^t} f(x^t), x^*_{S^*\setminus S^t}\rangle + \frac{\rho^-}{2}\left\|x^* - x^t\right\|_2^2 \\
&\geq \langle \nabla_{S^*\setminus S^t} f(x^t), x^*_{S^*\setminus S^t}\rangle + \frac{\rho^-}{2}\left\|x^*_{S^*\setminus S^t}\right\|_2^2 + \frac{\rho^-}{2}\left\|x^t_{S^t\setminus S^*}\right\|_2^2 \\
&\geq \langle \nabla_{S^*\setminus S^t} f(x^t), x^*_{S^*\setminus S^t}\rangle + \frac{\mu\rho^+}{2}\left\|x^*_{S^*\setminus S^t}\right\|_2^2 \\
&+ \frac{\rho^- - \mu\rho^+}{2}\left\|x^*_{S^*\setminus S^t}\right\|_2^2 + \frac{\rho^-}{2}\left\|x^t_{S^t\setminus S^*}\right\|_2^2 \\
&\geq -\frac{1}{2\mu\rho^+}\left\|\nabla_{S^*\setminus S^t} f(x^t)\right\|_2^2 + \frac{\rho^- - \mu\rho^+}{2}\left\|x^*_{S^*\setminus S^t}\right\|_2^2 + \frac{\rho^-}{2}\left\|x^t_{S^t\setminus S^*}\right\|_2^2,
\end{aligned} \tag{6.14}$$

where the first equality follows from the fact that $\nabla_{S^t} f(x^t) = \vec{0}$ as $x^t$ is an $S^t$-restricted minimizer of $f$ and the last inequality from using the fact that $\langle u, v\rangle + \frac{\lambda}{2}\|v\|_2^2 \geq -\frac{1}{2\lambda}\|u\|_2^2$ for any $\lambda > 0$.

Re-arranging (6.14), we get

$$\frac{1}{2\mu\rho^+}\left\|\nabla_{S^*\setminus S^t} f(x^t)\right\|_2^2 \geq f(x^t) - f(x^*) - \frac{\mu\rho^+ - \rho^-}{2}\left\|x^*_{S^*\setminus S^t}\right\|_2^2 + \frac{\rho^-}{2}\left\|x^t_{S^t\setminus S^*}\right\|_2^2.$$

By substituting this into (6.13),

$$\begin{aligned}
&|S^*\backslash S^t|(f(x^t)-f(x^{t+1})) \\
&\geq \mu\left(f(x^t)-f(x^*)\right)-\frac{\mu^2\rho^+-\mu\rho^-}{2}\left\|x^*_{S^*\backslash S^t}\right\|_2^2+\frac{\mu\rho^-}{2}\left\|x^t_{S^t\backslash S^*}\right\|_2^2-\frac{\rho^+}{2}\frac{|S^*\backslash S^t|}{|S^t\backslash S^*|}\left\|x^t_{S^t\backslash S^*}\right\|_2^2.
\end{aligned}$$

Note that by our choice of $\mu$ and since $s^*\leq s$,

$$\mu^2\rho^+=\rho^+\frac{s^*}{s}\geq\rho^+\frac{s^*-|S^*\cap S^t|}{s-|S^*\cap S^t|}=\rho^+\frac{|S^*\backslash S^t|}{|S^t\backslash S^*|}$$

and so

$$\begin{aligned}
&\mu\left(f(x^t)-f(x^*)\right)-\frac{\mu^2\rho^+-\mu\rho^-}{2}\left\|x^*_{S^*\backslash S^t}\right\|_2^2+\frac{\mu\rho^-}{2}\left\|x^t_{S^t\backslash S^*}\right\|_2^2-\frac{\rho^+}{2}\frac{|S^*\backslash S^t|}{|S^t\backslash S^*|}\left\|x^t_{S^t\backslash S^*}\right\|_2^2 \\
&\geq \mu(f(x^t)-f(x^*))-\frac{\mu}{2}\left(\mu\rho^+-\rho^-\right)\left(\left\|x^*_{S^*\backslash S^t}\right\|_2^2+\left\|x^t_{S^t\backslash S^*}\right\|_2^2\right),
\end{aligned}$$

concluding that

$$|S^*\backslash S^t|(f(x^t)-f(x^{t+1}))\geq\mu(f(x^t)-f(x^*))-\frac{\mu}{2}\left(\mu\rho^+-\rho^-\right)\left(\left\|x^*_{S^*\backslash S^t}\right\|_2^2+\left\|x^t_{S^t\backslash S^*}\right\|_2^2\right).$$

For $\mu\widetilde{\kappa}\leq 1\Leftrightarrow\mu\rho^+-\rho^-\leq 0$, this automatically implies that

$$\begin{aligned}
&f(x^t)-f(x^{t+1})\geq\frac{\mu}{|S^*\backslash S^t|}\left(f(x^t)-f(x^*)\right) \\
\Leftrightarrow&f(x^{t+1})-f(x^*)\leq\left(1-\frac{\mu}{|S^*\backslash S^t|}\right)\left(f(x^t)-f(x^*)\right).
\end{aligned}$$

On the other hand, if $\mu\widetilde{\kappa}>1$ we have

$$\begin{aligned}
&|S^*\backslash S^t|\left(f(x^t)-f(x^{t+1})\right) \\
&\geq\mu(f(x^t)-f(x^*))-\frac{\mu}{2}\left(\mu\rho^+-\rho^-\right)\left(\left\|x^*_{S^*\backslash S^t}\right\|_2^2+\left\|x^t_{S^t\backslash S^*}\right\|_2^2\right) \\
&\geq\mu(f(x^t)-f(x^*))-\mu\left(\mu\widetilde{\kappa}-1\right)\left(\sqrt{f(x^t)-f(\widetilde{x}^t)}+\sqrt{f(x^*)-f(\widetilde{x}^t)}\right)^2,
\end{aligned}$$

where we used Lemma 6.4.1. If $f(x^*)=f(\widetilde{x}^t)$ it is immediate that

$$f(x^{t+1})-f(x^*)\leq\left(1-\frac{\mu}{|S^*\backslash S^t|}\left(2-\mu\widetilde{\kappa}\right)\right)\left(f(x^t)-f(x^*)\right),$$

so let us from now on assume that $f(x^*)>f(\widetilde{x}^t)$ and set $a=f(x^t)-f(\widetilde{x}^t)$, $a'=f(x^{t+1})-f(\widetilde{x}^t)$, and $b=f(x^*)-f(\widetilde{x}^t)$. From what we have concluded before

$$|S^*\backslash S^t|\left(a-a'\right)\geq\mu(a-b)-\mu(\mu\widetilde{\kappa}-1)\left(\sqrt{a}+\sqrt{b}\right)^2,$$

or equivalently

$$\begin{aligned}
a' - b &\leq \left(1 - \frac{\mu}{|S^* \backslash S^t|}\right)(a-b) + \frac{\mu}{|S^* \backslash S^t|}(\mu\widetilde{\kappa} - 1)\left(\sqrt{a} + \sqrt{b}\right)^2 \\
&= (a-b)\left(1 - \frac{\mu}{|S^* \backslash S^t|}\left(1 - (\mu\widetilde{\kappa} - 1)\frac{\left(\sqrt{a} + \sqrt{b}\right)^2}{a-b}\right)\right) \\
&= (a-b)\left(1 - \frac{\mu}{|S^* \backslash S^t|}\left(1 - (\mu\widetilde{\kappa} - 1)\frac{\sqrt{\frac{a}{b}} + 1}{\sqrt{\frac{a}{b}} - 1}\right)\right) \\
&= (a-b)\left(1 - \frac{\mu}{|S^* \backslash S^t|}\left(1 - (\mu\widetilde{\kappa} - 1)\left(1 + \frac{2}{\sqrt{\frac{a}{b}} - 1}\right)\right)\right) \\
&= (a-b)\left(1 - \frac{\mu}{|S^* \backslash S^t|}\left(2 - \mu\widetilde{\kappa} - \frac{2(\mu\widetilde{\kappa} - 1)}{\sqrt{\frac{a}{b}} - 1}\right)\right),
\end{aligned}$$

Replacing back $a, a', b$, the desired statement follows:

$$f(x^{t+1}) - f(x^*) \leq (f(x^t) - f(x^*)) \cdot \left(1 - \frac{\mu}{|S^* \backslash S^t|}\left(2 - \mu\widetilde{\kappa} - \frac{2(\mu\widetilde{\kappa} - 1)}{\sqrt{\frac{f(x^t) - f(\widetilde{x}^t)}{f(x^*) - f(\widetilde{x}^t)}} - 1}\right)\right).$$

□

The proof of Theorem 6.4.2 now follows by appropriately applying Lemma 6.4.3.

**Proof of Theorem 6.4.2.** We will directly apply Lemma 6.4.3 over the course of multiple iterations. For the second bullet of the theorem statement, the progress bound of Lemma 6.4.3 also depends on $(f(x^t) - f(\widetilde{x}^t))/(f(x^*) - f(\widetilde{x}^t))$. We show that, as long as $f(x)$ is singificantly larger than $f(x^*)$, this can be lower bounded by a sufficiently large quantity, leading to fast convergence.

*Case 1:* $\mu\widetilde{\kappa} \leq 1$.

By Lemma 6.4.3, we have

$$\begin{aligned}
f(x^T) - f(x^*) &\leq \left(f(x^{T-1}) - f(x^*)\right)\left(1 - \frac{\mu}{|S^* \backslash S^{T-1}|}\right) \\
&\leq \left(f(x^{T-1}) - f(x^*)\right)\left(1 - \frac{\mu}{s^*}\right) \\
&\leq \left(f(x^{T-1}) - f(x^*)\right) e^{-\frac{\mu}{s^*}} \\
&\leq \dots \\
&\leq \left(f(x^0) - f(x^*)\right) e^{-T\frac{\mu}{s^*}} \\
&\leq \varepsilon,
\end{aligned}$$

for our choice of $T = O\left(\sqrt{ss^*}\log\frac{f(x^0) - f(x^*)}{\varepsilon}\right)$ and replacing $\mu = \sqrt{\frac{s^*}{s}}$.

*Case 2:* $\mu\widetilde{\kappa} > 1$.

Let $\mathcal{A}$ be the set of $0 \leq t \leq T-1$ such that $f(x^*) = f(\widetilde{x}^t)$ and $\mathcal{B}$ the set of $0 \leq t \leq T-1$ such

that $f(x^*) > f(\widetilde{x}^t)$. By Lemma 6.4.3, for $t \in \mathcal{A}$ we then have

$$\begin{aligned} f(x^{t+1}) - f(x^*) &\leq \big(f(x^t) - f(x^*)\big)\left(1 - \frac{\mu}{|S^* \backslash S^t|}\left(2 - \mu\widetilde{\kappa}\right)\right) \\ &\leq \big(f(x^t) - f(x^*)\big)\left(1 - \frac{\mu}{s^*}\left(2 - \mu\widetilde{\kappa}\right)\right). \end{aligned}$$

We now consider the case $t \in \mathcal{B}$. By Lemma 6.4.3,

$$f(x^{t+1}) - f(x^*) \leq \big(f(x^t) - f(x^*)\big) \cdot \left(1 - \frac{\mu}{|S^* \backslash S^t|}\left(2 - \mu\widetilde{\kappa} - \frac{2(\mu\widetilde{\kappa} - 1)}{\sqrt{\frac{f(x^t) - f(\widetilde{x}^t)}{f(x^*) - f(\widetilde{x}^t)}} - 1}\right)\right). \tag{6.15}$$

Let us suppose that the theorem statement is not true. This implies

$$\begin{aligned} f(x^t) - f(x^*) &\geq f(x^T) - f(x^*) \\ &> \varepsilon + \frac{4(1-\theta)(\mu\widetilde{\kappa} - 1)}{\left(2 - \mu\widetilde{\kappa} - \theta\right)^2}(f(x^*) - f(\overline{x}^*)) \\ &\geq \varepsilon + \frac{4(1-\theta)(\mu\widetilde{\kappa} - 1)}{\left(2 - \mu\widetilde{\kappa} - \theta\right)^2}(f(x^*) - f(\widetilde{x}^t)) \\ &\geq \frac{4(1-\theta)(\mu\widetilde{\kappa} - 1)}{\left(2 - \mu\widetilde{\kappa} - \theta\right)^2}(f(x^*) - f(\widetilde{x}^t)), \end{aligned} \tag{6.16}$$

for all $0 \leq t \leq T$. Therefore

$$\begin{aligned} f(x^t) - f(\widetilde{x}^t) &> \left(\frac{4(1-\theta)(\mu\widetilde{\kappa} - 1)}{\left(2 - \mu\widetilde{\kappa} - \theta\right)^2} + 1\right)(f(x^*) - f(\widetilde{x}^t)) \\ &= \left(\frac{4(1-\theta)(\mu\widetilde{\kappa} - 1) + 4 + (\mu\widetilde{\kappa} + \theta)^2 - 4(\mu\widetilde{\kappa} + \theta)}{\left(2 - \mu\widetilde{\kappa} - \theta\right)^2}\right) \cdot (f(x^*) - f(\widetilde{x}^t)) \\ &= \frac{(\mu\widetilde{\kappa} - \theta)^2}{\left(2 - \mu\widetilde{\kappa} - \theta\right)^2}(f(x^*) - f(\widetilde{x}^t)), \end{aligned}$$

or equivalently for all $t \in \mathcal{B}$

$$\sqrt{\frac{f(x^t) - f(\widetilde{x}^t)}{f(x^*) - f(\widetilde{x}^t)}} - 1 > \frac{\mu\widetilde{\kappa} - \theta}{2 - \mu\widetilde{\kappa} - \theta} - 1 = \frac{2(\mu\widetilde{\kappa} - 1)}{2 - \mu\widetilde{\kappa} - \theta}.$$

Replacing this into (6.15), we get that for any $t \in \mathcal{B}$

$$\begin{aligned} f(x^{t+1}) - f(x^*) &\leq (f(x^t) - f(x^*)) \cdot \left(1 - \frac{\mu}{|S^* \backslash S^t|}\left(2 - \mu\widetilde{\kappa} - \frac{2(\mu\widetilde{\kappa} - 1)}{\sqrt{\frac{f(x^t) - f(\widetilde{x}^t)}{f(x^*) - f(\widetilde{x}^t)}} - 1}\right)\right) \\ &\leq (f(x^t) - f(x^*))\left(1 - \frac{\mu}{|S^* \backslash S^t|}\theta\right) \end{aligned}$$

and so combining it with the case $t \in \mathcal{A}$ and using the fact that $\mu\widetilde{\kappa} < 2 - \theta \Leftrightarrow \theta < 2 - \mu\widetilde{\kappa}$,

$$\begin{aligned}
f(x^T) - f(x^*) &\leq \left(f(x^{T-1}) - f(x^*)\right)\left(1 - \frac{\mu}{|S^* \backslash S^{T-1}|}\min\left\{2 - \mu\widetilde{\kappa}, \theta\right\}\right) \\
&\leq \left(f(x^{T-1}) - f(x^*)\right)\left(1 - \frac{\mu}{s^*}\theta\right) \\
&\leq \left(f(x^{T-1}) - f(x^*)\right) e^{-\frac{\mu}{s^*}\theta} \\
&\leq \dots \\
&\leq (f(x^0) - f(x^*))e^{-T\frac{\mu}{s^*}\theta} \\
&= \varepsilon + \frac{4(1-\theta)(\mu\widetilde{\kappa} - 1)}{(2 - \mu\widetilde{\kappa} - \theta)^2}(f(x^*) - f(\overline{x}^*)),
\end{aligned}$$

where the last equality follows by our choice of

$$T = \frac{\sqrt{ss^*}}{\theta}\log\frac{f(x^0) - f(x^*)}{B}$$

and replacing $\mu = \sqrt{\frac{s^*}{s}}$. This is a contradiction. ■

### 6.4.3 Corollaries of Theorem 6.4.2

The first corollary states that in the "noiseless" case (i.e. when the target solution is globally optimal), the returned solution can reach arbitrarily close to the target solution. For $s = s^*$, it gives the same condition of $\widetilde{\kappa} < 2$ (or $\delta < 1/3$) as in [87], while working for any function $f$. For $s > s^*$, it additionally gives a tradeoff between the sparsity and the RIP bound required for the algorithm.

**Corollary 6.4.4** (Noiseless case)**.** *If $\widetilde{\kappa}\sqrt{\frac{s^*}{s}} < 2$ and $x^*$ is a globally optimal solution, i.e. $f(x^*) = \min_z f(z)$, Algorithm 12 returns a solution with*

$$f(x) \leq f(x^*) + \varepsilon$$

*in $O\left(\frac{\sqrt{ss^*}}{2 - \widetilde{\kappa}\sqrt{\frac{s^*}{s}}}\log\frac{f(x^0) - f(x^*)}{\varepsilon}\right)$ iterations.*

*Proof.* We apply Theorem 6.4.2 with $\theta = \frac{1}{2}\left(2 - \widetilde{\kappa}\sqrt{\frac{s^*}{s}}\right)$. □

The following result is in the usual form of sparse recovery results, which provide a bound on $\|x - x^*\|_2$ given a RIP constant upper bound. It provides a tradeoff between the RIP constant and the sparsity of the returned solution. Similar results can be found e.g. in [31] using LASSO, albeit they only apply to case of linear regression ($f(x) = \frac{1}{2}\|Ax - b\|_2^2$) and do not offer a sparsity tradeoff.

**Corollary 6.4.5** ($\ell_2$ solution recovery)**.** *Given any parameters $\varepsilon > 0$ and $0 < \theta < 1$, the returned solution $x$ of Algorithm 12 will satisfy*

$$\|x - x^*\|_2^2 \leq \varepsilon + C\left(f(x^*) - \min_z f(z)\right)$$

*as long as*

$$\delta_{s+s^*} < \frac{(2-\theta)\sqrt{\frac{s}{s^*}} - 1}{(2-\theta)\sqrt{\frac{s}{s^*}} + 1},$$

*where $C$ is a constant that depends only on $\theta$, $\delta_{s+s^*}$, and $\frac{s}{s^*}$.*

*Proof.* By triangle inequality and strong convexity, and letting $\widetilde{x}^*$ be the optimal solution in $\mathrm{supp}(x) \cup \mathrm{supp}(x^*)$, we have

$$\begin{aligned}
\|x - x^*\|_2^2 &\leq 2\left(\|x - \widetilde{x}^*\|_2^2 + \|x^* - \widetilde{x}^*\|_2^2\right) \\
&\leq \frac{4}{1-\delta_{s+s^*}}\left(f(x) - f(\widetilde{x}^*) + f(x^*) - f(\widetilde{x}^*)\right) \\
&= \frac{4}{1-\delta_{s+s^*}}\left(f(x) - f(x^*) + 2(f(x^*) - f(\widetilde{x}^*))\right) \\
&= \frac{4}{1-\delta_{s+s^*}}\left(f(x) - f(x^*) + 2(f(x^*) - \min_z f(z))\right).
\end{aligned}$$

Now, by applying Theorem 6.4.2 for some error tolerance $\widehat{\varepsilon} > 0$, we get

$$f(x) \leq f(x^*) + \widehat{\varepsilon} + \widehat{C}(f(x^*) - f(\overline{x}^*)) \leq f(x^*) + \widehat{\varepsilon} + \widehat{C}(f(x^*) - \min_z f(z)),$$

for some $\widehat{C} > 0$, and so we conclude that

$$\|x - x^*\|_2^2 \leq \frac{4\widehat{\varepsilon}}{1-\delta_{s+s^*}} + \frac{4(\widehat{C}+2)}{1-\delta_{s+s^*}}(f(x^*) - \min_z f(z)).$$

The statement follows by setting $\varepsilon = \frac{4\widehat{\varepsilon}}{1-\delta_{s+s^*}}$ and $C = \frac{4(\widehat{C}+2)}{1-\delta_{s+s^*}}$. □

In particular, for $s = s^*$, the above lemma implies recovery under the condition $\delta_{2s^*} < \frac{1}{3}$.

## 6.5 Lower Bounds

### 6.5.1 $\Omega(s^*\kappa)$ Lower Bound due to [63]

In Appendix B of [63] a matrix $A \in \mathbb{R}^{m\times n}$ and a vector $b \in \mathbb{R}^m$ are constructed and let us define $f(x) = \frac{1}{2}\|Ax - b\|_2^2$. If we let $\overline{S^*} = \{1, \ldots, n-2\}$ and $S^* = \{n-1, n\}$, then $f$ has the property that

$$\min_{\mathrm{supp}(x)\subseteq S^*} f(x) = \min_{\mathrm{supp}(x)\subseteq \overline{S^*}} f(x) = 0,$$

but for any $S \subset \overline{S^*}$,

$$\min_{\mathrm{supp}(x)\subseteq S} f(x) > 0.$$

Furthermore, for any $S \subset \overline{S^*}$ and $x = \operatorname*{argmin}_{\mathrm{supp}(x)\subseteq S} f(x)$, it is true that

$$\max_{i\in S^*} |\nabla_i f(x)| < \min_{i\in \overline{S^*}\setminus S} |\nabla_i f(x)|.$$

This means that for any algorithm with an OMP-like criterion like Orthogonal Matching Pursuit, Orthogonal Matching Pursuit with Replacement, Iterative Hard Thresholding, and Partial Hard Thresholding, if the initial solution does not have an intersection with $S^*$, then it will never have, therefore implying that the sparsity returned by the algorithm is $|S| = n - 2 = \Omega(n)$. As for this construction $\kappa = \frac{\rho_n^+}{\rho_n^-} = O(n)$, there exists a constant $c$ such that the sparsity of the returned solution cannot be less than $cs^*\kappa$, since $s^*\kappa = O(n) = O(|S|)$. Therefore none of these algorithms can improve the bound $O(s^*\kappa)$ of Theorem 6.3.1 by more than a constant factor. This example also applies to ARHT and Exhaustive Local Search.

It seems difficult to get past this example and achieve sparsity $s = O(s^*\kappa^{1-\delta})$ for some $\delta > 0$. We conjecture that there might be a way to turn the above example into an inapproximability result:

**Conjecture 6.5.1.** *For any $\delta > 0$, there is no polynomial time algorithm that given a matrix $A \in \mathbb{R}^{m\times n}$, a vector $b \in \mathbb{R}^m$, a target sparsity $s^* \geq 1$, and a desired accuracy $\varepsilon > 0$, returns an $s = O(s^*\kappa_{s+s^*}^{1-\delta})$-sparse solution $x$ such that $\|Ax - b\|_2^2 \leq \min_{\|x^*\|_0 \leq s^*} \|Ax^* - b\|_2^2 + \varepsilon$, if such a solution exists.*

### 6.5.2 $\Omega(s^*\kappa^2)$ Lower Bound for OMPR

The following lemma shows that, without regularization, OMPR requires sparsity $\Omega(s^*\kappa^2)$ in general, and therefore the sparsity upper bound is tight. We assume that the algorithm is run for a fixed $T$ iterations, even when the solution stops improving, for a clearer presentation.

**Lemma 6.5.2.** *There is a function $f(x) = \frac{1}{2}\|Ax - b\|_2^2$ where $A \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^n$ and a target solution $x^*$ of $f$ with sparsity $s^*$, as well as a set $S \subseteq [n]$ with $|S| = \Theta(s^*\kappa^2)$ such that OMPR initialized with support set $S$ returns a solution $x$ with $f(x) = f(x^*) + \Theta(s^*\kappa^2)$.*

*Proof.* Without loss of generality we assume that $\kappa$ is an even integer and set $n = s^*\left(1 + \kappa + \kappa^2\right)$. We then partition $[n]$ into three intervals $I_1 = [1, s^*]$, $I_2 = [s^* + 1, s^*(1 + \kappa)]$, $I_3 = [s^*(1 + \kappa) + 1, s^*(1 + \kappa + \kappa^2)]$. We define the diagonal matrix $A \in \mathbb{R}^{n\times n}$ such that

$$A_{ii} = \begin{cases} 1 & \text{if } i \in I_1 \\ \sqrt{\kappa} & \text{if } i \in I_2 \\ 1 & \text{if } i \in I_3 \end{cases}$$

and vector $b \in \mathbb{R}^n$ such that

$$b_i = \begin{cases} \kappa\sqrt{1 - 4\delta} & \text{if } i \in I_1 \\ \sqrt{\kappa}\sqrt{1 - 2\delta} & \text{if } i \in I_2 \\ 1 & \text{if } i \in I_3 \end{cases},$$

where $\delta > 0$ is a sufficiently small scalar used to avoid ties in the steps of the algorithm. The target solution is defined as

$$x_i^* = \begin{cases} \kappa(1 - 4\delta) & \text{if } i \in I_1 \\ 0 & \text{if } i \in I_2 \cup I_3 \end{cases}$$

and its value is $f(x^*) = s^*\kappa^2(1 - \delta)$. Now consider any initial support set $S^0 \subset I_3$ such that

$|S^0| = s^*\kappa^2/2$. The initial solution will then be

$$x_i^0 = \begin{cases} 0 & \text{if } i \in I_1 \cup I_2 \cup I_3 \backslash S^0 \\ 1 & \text{if } i \in S^0 \end{cases}$$

and its value $f(x^0) = s^*\kappa^2 \left(\frac{5}{4} - 3\delta\right) = f(x^*) + \Theta(s^*\kappa^2)$. The gradient at $x^0$ is

$$\nabla_i f(x^0) = \begin{cases} -\kappa\sqrt{1-4\delta} & \text{if } i \in I_1 \\ -\kappa\sqrt{1-2\delta} & \text{if } i \in I_2 \\ -1 & \text{if } i \in I_3 \backslash S^0 \\ 0 & \text{if } i \in S^0 \end{cases},$$

therefore the algorithm will pick $S^1 = S^0 \cup \{i^0\} \backslash \{j^0\}$ for some $i^0 \in I_2$ and some $j^0 \in S^0$, since the gradient entries in $I_2$ have the largest magnitude among those in $[n]$. The new solution will be

$$x_i^1 = \begin{cases} 0 & \text{if } i \in I_1 \cup I_2 \cup I_3 \backslash S^1 \\ \sqrt{1-2\delta} & \text{if } i = i^0 \\ 1 & \text{if } i \in S^1 \backslash \{i^0\} \end{cases}$$

with value $f(x^1) = s^*\kappa^2 \left(\frac{5}{4} - 3\delta\right) - \frac{1}{2}(\kappa(1-2\delta) - 1)$ and gradient

$$\nabla_i f(x^1) = \begin{cases} -\kappa\sqrt{1-4\delta} & \text{if } i \in I_1 \\ -\kappa\sqrt{1-2\delta} & \text{if } i \in I_2 \backslash S^1 \\ -1 & \text{if } i \in I_3 \backslash S^1 \\ 0 & \text{if } i \in S^1 \end{cases}$$

and therefore the algorithm will pick $S^2 = S^1 \cup \{i^1\} \backslash \{i^0\}$ for some $i^1 \in I_2$. $i^0$ will be the one to be removed from $S^1$ because $x_{i^0}$ has the smallest magnitude out of all entries in $S^1$. Continuing this process, the algorithm will always have $S^t \cap I_2 = 1$ and $S^t \cap I_3 = |S^t| - 1$, and so $f(x^t) = s^*\kappa^2 \left(\frac{5}{4} - 3\delta\right) - \frac{1}{2}(\kappa(1-2\delta) - 1) = f(x^*) + \Theta(s^*\kappa^2)$ for $t \geq 1$. □

## 6.6 Experiments

### 6.6.1 Overview

In this section we evaluate the training performance of different algorithms in the tasks of Linear Regression and Logistic Regression. More specifically, for each algorithm we are interested in how the *loss* over the training set (the quality of the solution) evolves as a function of the the *sparsity* of the solution, i.e. the number of non-zeros.

The algorithms that we will consider are *LASSO*, *Orthogonal Matching Pursuit (OMP)*, *Orthogonal Matching Pursuit with Replacement (OMPR)*, *Adaptively Regularized Hard Thresholding (ARHT)* (Algorithm 15), and *Exhaustive Local Search* (Algorithm 13). We run our experiments on publicly available regression and binary classification data sets, out of which we have presented those on which the algorithms have significantly different performance between each other. In some of the other data sets that we tested, we observed that all algorithms had similar performance. The results are presented in Figures 6-1, 6-2, 6-3, 6-4. Also, in Figure 6-5 we present a runtime comparison

between ARHT and Exhaustive Local Search in the year and census datasets. Another relevant class of algorithms that we considered was $\ell_p$ *Appproximate Message Passing* algorithms [48, 171]. Brief experiments showed its performance in terms of sparsity for $p \leq 0.5$ to be promising (on par with OMPR and ARHT although these had much faster runtimes), however a detailed comparison is left for future work.

In both types of objectives (linear and logistic) we include an intercept term, which is present in all solutions (i.e. it is always counted as +1 in the sparsity of the solution). For consistency, all greedy algorithms (OMPR, ARHT, Exhaustive Local Search) are initialized with the OMP solution of the same sparsity.

We note that this section is not supposed to be a conclusive experimental evaluation of the aforementioned algorithms, but rather a preliminary set of experiments that gives partial evidence for their performance. A more extensive future experimental evaluation should focus on implementing runtime-optimized versions of these algorithms and generating sparsity vs loss and loss vs runtime plots for a larger collection of real datasets and with more datapoints and features.

These experiments suggest that Exhaustive Local Search outperforms the other algorithms. However, ARHT also has promising performance and it might be preferred because of better computational efficiency. As a general conclusion, however, both Exhaustive Local Search and ARHT give sparser solutions than other known methods in all the examples we tested. We leave a comprehensive comparison for future work. As a limitation, we observe that ARHT has inconsistent performance in some cases, oscillating between the Exhaustive Local Search and OMPR solutions.

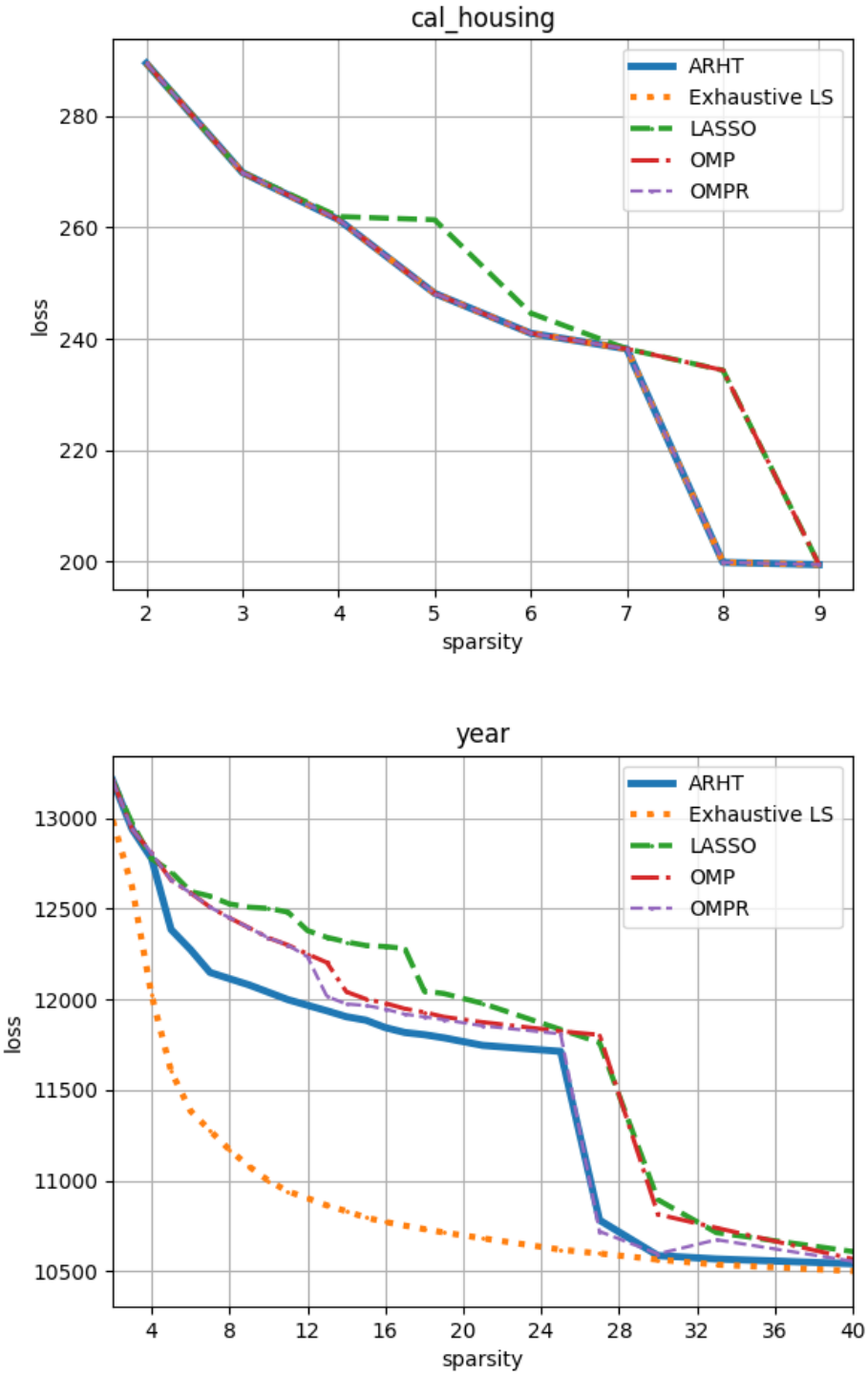



Figure 6-1: Comparison of different algorithms in the Regression data sets *cal_housing* and *year* using the Linear Regression loss.

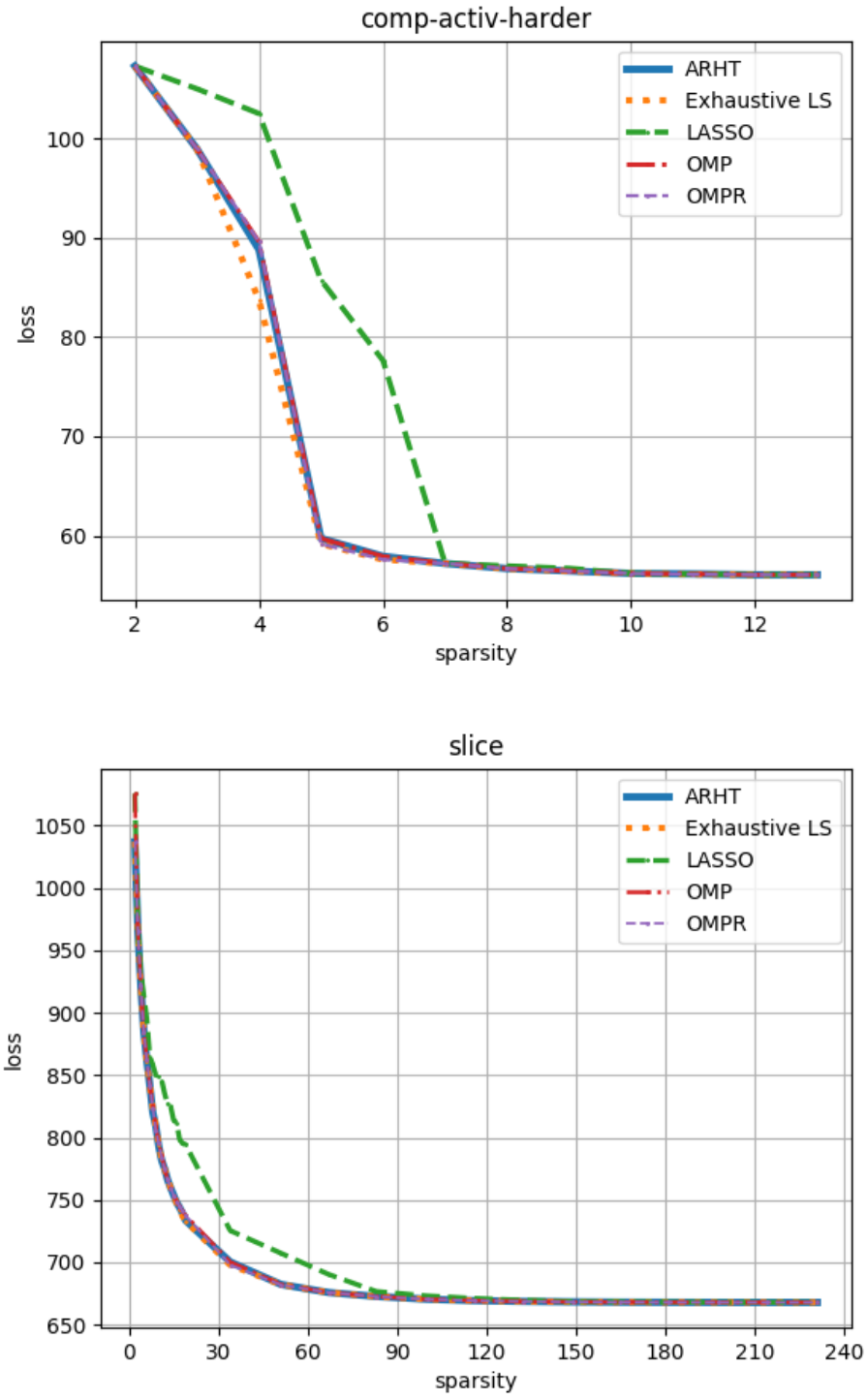



Figure 6-2: Comparison of different algorithms in the Regression data sets *comp-activ-harder* and *slice* using the Linear Regression loss.

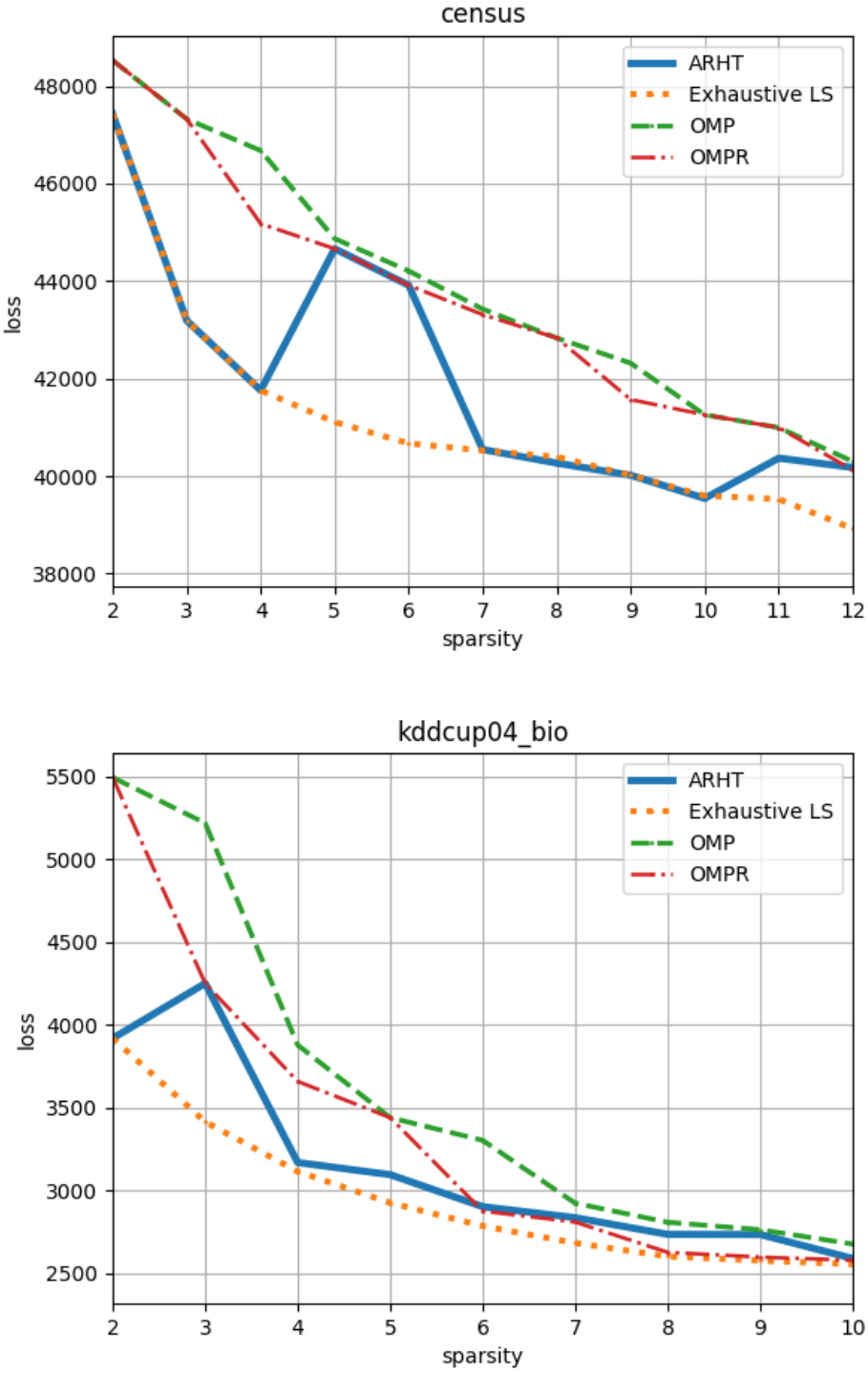



Figure 6-3: Comparison of different algorithms in the Binary classification data sets *census* and *kddcup04_bio* using the Logistic Regression loss.

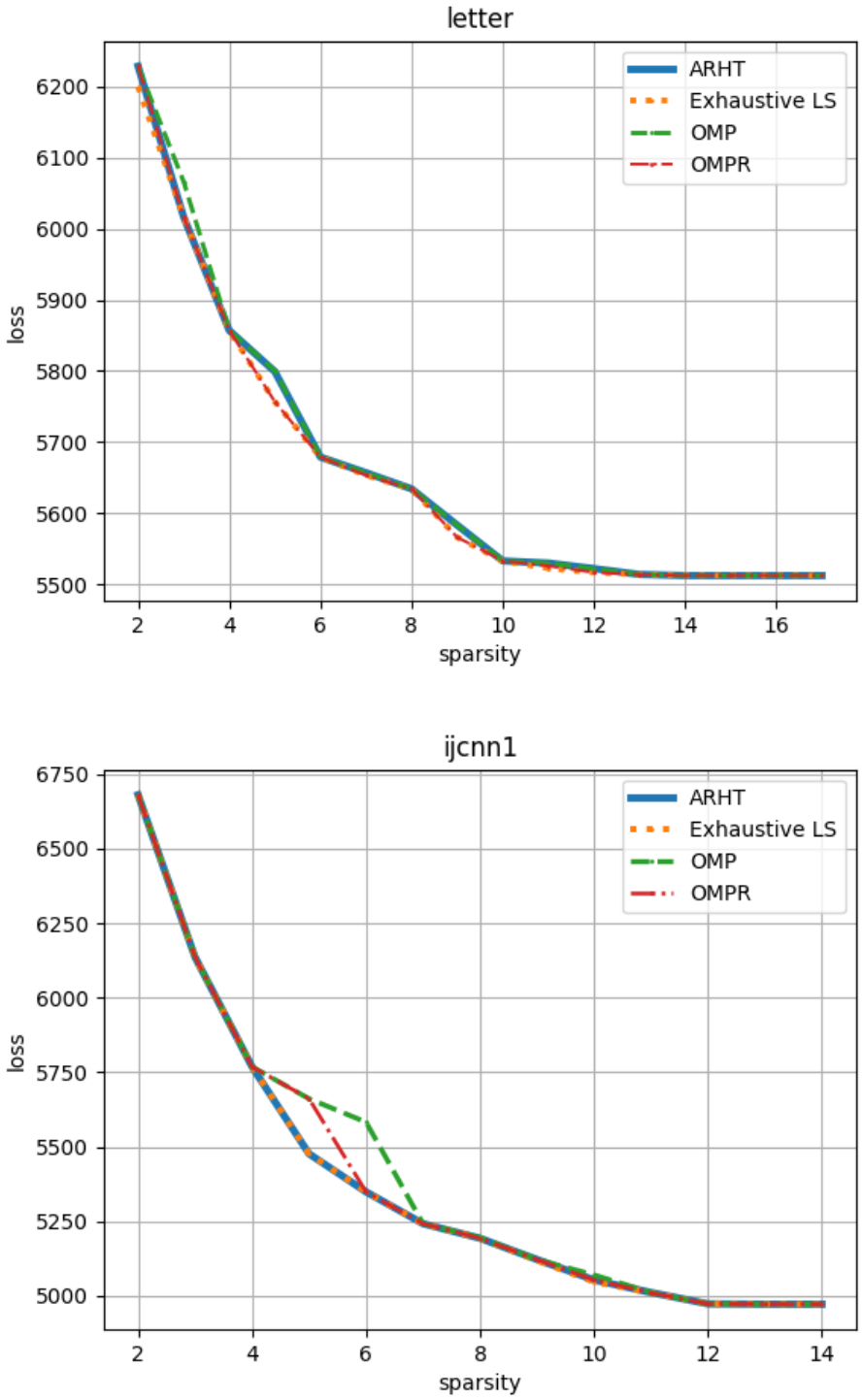



Figure 6-4: Comparison of different algorithms in the Binary classification data sets *letter* and *ijcnn1* using the Logistic Regression loss.

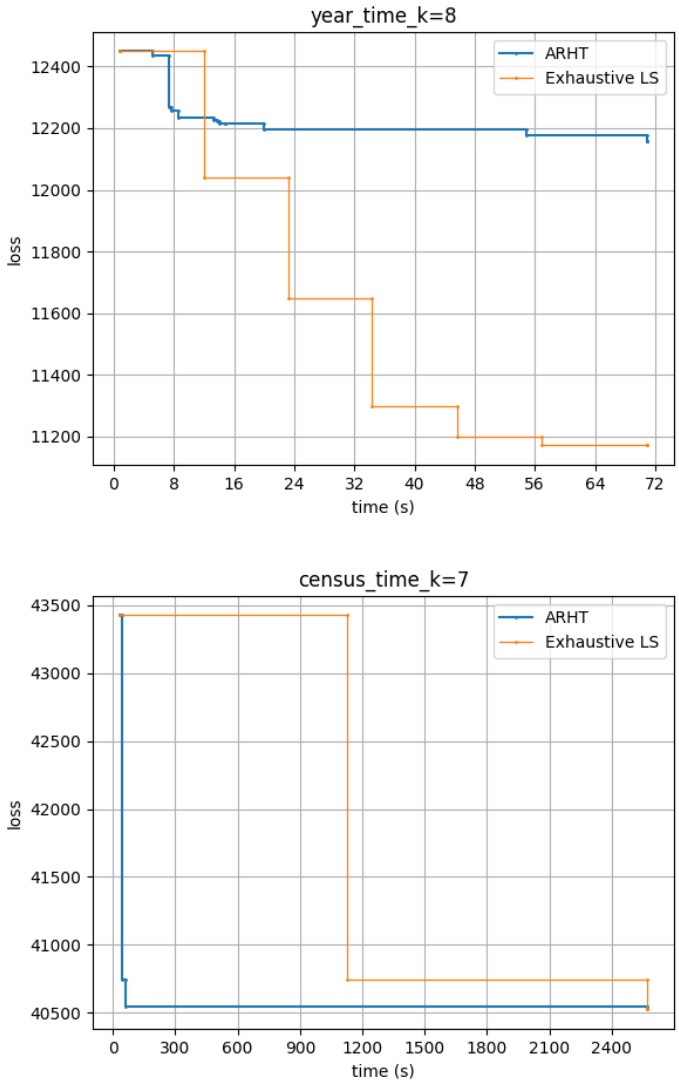



Figure 6-5: Comparison of the loss of a solution vs time elapsed to compute it, between ARHT and Exhaustive Local Search. The first experiment is on the *year* dataset (90 total features) with fixed sparsity 8 and the second is on the *census* dataset (401 total features) with fixed sparsity 7. We note that in the first case Exhaustive Local Search returns significantly sparser solutions without too significant time overhead compared to ARHT. However, in the second case ARHT computes a solution of similar loss to that of Exhaustive Local Search, but around 40 times faster. We attribute this to the fact that the Exhaustive Local Search has an extra $n$ factor in the runtime, so it doesn't scale as well as ARHT as the number of features increases. Note: The reason there are "steps" in the plot is that the solution is improved at discrete time steps, i.e. whenever an insertion and removal from the solution support improves the solution.

For experimental evaluation we used well known and publicly available data sets. Their names and basic properties are outlined in Table 6.2.

Table 6.2: Data sets used for experimental evaluation. The columns are the data set name, the number of examples $m$, and the number of features $n$. The data sets can be downloaded here.

| NAME | $n$ | $d$ | PROBLEM |
|---|---|---|---|
| KDDCUP04_BIO | 145750 | 74 | BINARY |
| CAL_HOUSING | 20639 | 8 | REGRESSION |
| CENSUS | 299284 | 401 | BINARY |
| COMP-ACTIV-HARDER | 8191 | 12 | REGRESSION |
| IJCNN1 | 24995 | 22 | BINARY |
| LETTER | 20000 | 16 | BINARY |
| SLICE | 53500 | 384 | REGRESSION |
| YEAR | 463715 | 90 | REGRESSION |

### 6.6.2 Setup Details

#### Basic Definitions

The two quantities that take part in our experiments are the *sparsity* and the *loss* of a particular solution. We have already defined and discussed the former at length. The latter refers to the training loss for the problems of Linear Regression and Logistic Regression. We let $m$ denote the number of examples and $n$ the number of features in each example.

In the *Linear Regression* task we are given the data set $(A, b)$, where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$. The columns of $A$ correspond to features and the rows to examples. The *($\ell_2$ Linear Regression) loss* of a solution $x \in \mathbb{R}^n$ is defined as $\ell_2\_\text{loss}(x) = \frac{1}{2} \|Ax - b\|_2^2$.

In the *Logistic Regression* task we are given the data set $(A, b)$, where $A \in \mathbb{R}^{m \times n}$, $b \in \{0, 1\}^m$. The columns of $A$ correspond to features and the rows to examples. The *(Logistic Regression) loss* of a solution $x \in \mathbb{R}^n$ is defined as

$$\text{logistic\_loss}(x) = \sum_{i \in [m]} \left(-b_i \log \sigma(Ax)_i - (1 - b_i) \log(1 - \sigma(Ax)_i)\right) ,$$

where $\sigma : \mathbb{R} \to \mathbb{R}$ defined as $\sigma(t) = \frac{1}{1+e^{-t}}$ is the sigmoid function.

#### Data Pre-processing

We apply a very basic form of pre-processing to the data. More specifically, we use one-hot encoding to turn categorical features into numerical ones. Then, we discard any examples with missing data so that all the entries of $A$ are defined. We also augment the matrix $A$ with an extra all-ones column (i.e. $\vec{1}$) in order to encode the constant ($y$-intercept) term into $A$, and we scale all the columns of $A$ so that their $\ell_2$ norm is 1. Finally, for the case of ARHT we further augment $A$ in order to encode the regularizer as well. We do this by adding an identity matrix as extra rows. In other words, $A \leftarrow \begin{pmatrix} A \\ I \end{pmatrix}$ and $b \leftarrow \begin{pmatrix} b \\ \vec{0} \end{pmatrix}$.

### 6.6.3 Implementation Details

The code has been implemented in *python3*, with libraries *numpy*, *sklearn*, and *scipy*.

### Inner Optimization Problem

All the algorithms except for LASSO rely on an inner optimization routine in a restricted subset of coordinates in each step. The inner optimization problem consists of solving a standard Linear Regression or Logistic Regression problem using only a submatrix of $A$ defined by a subset of $s$ of its columns. For that, we use *LinearRegression* and *LogisticRegression* from *sklearn.linear_model*. For Logistic Regression we used an LBFGS solver with 1000 iterations.

### Overall Algorithm

The LASSO solver we used is *Lasso* from *sklearn.linear_model* with 1000 iterations. As LASSO is not tuned in terms of a required sparsity $s$, but rather in terms of the regularization parameter $\alpha$, for each sparsity level we applied binary search on $\alpha$ in order to find a parameter $\alpha$ that gives the required sparsity.

For ARHT, we used a fixed number of 20 iterations at Line 5 of Algorithm 15. In Line 19 of Algorithm 14 we slightly weaken the progress condition to

$$g_{R^t}(x^t) - g_{R^t}(x^{t+1}) \geq \frac{10^{-3}}{s} \left( g_{R^t}(x^t) - \text{opt} \right) . \tag{6.17}$$

Furthermore, we do not perform a fixed number of iterations. Instead, we use a stopping criterion: If the progress condition (6.17) is not met and at least half the elements in $x^t$ have already been unregularized, i.e. $\left|S^t \backslash R^t\right| \geq \frac{1}{2} \left|S^t\right|$, then we stop. If a desirable solution has not been found, it means that this might be an unsuccessful run, and early termination can be used to detect such runs early and re-start, thus improving the runtime. The routine which samples an index $i$ proportional to $x_i^2$ was implementing by a standard sampling method that uses binary search on $i$ and flips a random coin at each step. This requires computation of interval sums of $x_i^2$, which is done by computing partial sums.

# Chapter 7

# Iterative Hard Thresholding with Adaptive Regularization: Sparser Solutions Without Sacrificing Runtime

## 7.1 Introduction

*Sparse optimization* is the task of optimizing a function $f$ over $s$-sparse vectors, i.e. those with at most $s$ non-zero entries. Examples of such optimization problems arise in machine learning, with the goal to make models smaller for efficiency, generalization, or interpretability reasons, and compressed sensing, where the goal is to recover an $s$-sparse signal from a small number of measurements. A closely related problem is *low rank optimization*, where the sparsity constraint is instead placed on the spectrum of the solution (which is a matrix). This problem is central in matrix factorization, recommender systems, robust principal components analysis, among other tasks. More generally, *structured sparsity* constraints have the goal of capturing the special structure of a particular task by restricting the set of solutions to those that are "simple" in an appropriate sense. Examples include group sparsity, tree- and graph-structured sparsity. For more on generalized sparsity measures see e.g. [142].

Among the huge number of algorithms that have been developed for the sparse optimization problems, three stand out as the most popular ones:

- The **LASSO** [161], which works by relaxing the $\ell_0$ (sparsity) constraint to an $\ell_1$ constraint, thus convexifying the problem.
- **Orthogonal matching pursuit** (OMP) [135], which works by building the solution greedily in an incremental fashion.
- **Iterative hard thresholding** (IHT) [19], which performs projected gradient descent on the set of sparse solutions.

Among these, IHT is generally the most efficient, since it has essentially no overhead over plain gradient descent, making it the tool of choice for large-scale applications.

### 7.1.1 Iterative Hard Thresholding (IHT)

Consider the *sparse convex optimization* problem

$$\min_{\|\boldsymbol{x}\|_0 \leq s} f(\boldsymbol{x}), \tag{7.1}$$

where $f$ is convex and $\|\boldsymbol{x}\|_0$ is the number of non-zero entries in the vector $\boldsymbol{x}$, i.e. the sparsity of $\boldsymbol{x}$. IHT works by repeatedly performing the following iteration

$$\boldsymbol{x}^{t+1} = H_{s'}\left(\boldsymbol{x}^t - \eta \cdot \nabla f(\boldsymbol{x}^t)\right), \tag{7.2}$$

where $H_{s'}$ is the *hard thresholding operator* that zeroes out all but the top $s'$ entries, for some (potentially relaxed) sparsity level $s'$, and $\eta > 0$ is the step size.

As (7.1) is known to be NP-hard [127] and even hard to approximate [63], an extra assumption needs to be made for the performance of the algorithm to be theoretically evaluated in a meaningful way. The most common assumption is that the *(restricted) condition number* of $f$ is bounded by $\kappa$ (or the *restricted isometry property* constant is bounded by $\delta$ [31]), but other assumptions have been studied as well, such as incoherence [47] and weak supermodularity [109]. The performance is then measured in terms of the sparsity $s'$ of the returned solution, as well as its error (value of $f$).

As it is known [89], IHT is guaranteed to return an $s' = O(s\kappa^2)$-sparse solution $\boldsymbol{x}$ with $f(\boldsymbol{x}) \leq f(\boldsymbol{x}^*) + \varepsilon$. In fact, as we show in Section 9.5.5, the $\kappa^2$ factor cannot be improved in the analysis. Recently, [14] presented an algorithm called ARHT, which improves the sparsity to $s' = O(s\kappa)$. However, their algorithm is much less efficient than IHT, for many reasons. So the question emerges:

> *Is there a sparse convex optimization algorithm that returns $O(s\kappa)$-sparse solutions, but whose runtime efficiency is comparable to IHT?*

The main contribution of our work is to show that this goal can be achieved, and done so by a surprisingly simple tweak to IHT.

### 7.1.2 Reconciling Sparsity and Efficiency: Regularized IHT

Our main result is the following theorem, which states that running IHT on an *adaptively regularized* objective function returns $O(s\kappa)$-sparse solutions that are $\varepsilon$-optimal in function value, while having no significant runtime overhead over plain gradient descent.

**Theorem 7.1.1** (Regularized IHT)**.** *Let $f \in \mathbb{R}^n \to \mathbb{R}$ be a convex function that is $\beta$-smooth and $\alpha$-strongly convex*[1]*, with condition number $\kappa = \beta/\alpha$, and $\boldsymbol{x}^*$ be an (unknown) $s$-sparse solution. Then, running Algorithm 16 with $\eta = (2\beta)^{-1}$ and $c = s'/(4T)$ for*

$$T = O\left(\kappa \log \frac{f(\boldsymbol{x}^0) + (\beta/2)\left\|\boldsymbol{x}^0\right\|_2^2 - f(\boldsymbol{x}^*)}{\varepsilon}\right)$$

*iterations starting from an arbitrary $s' = O(s\kappa)$-sparse solution $\boldsymbol{x}^0$, the algorithm returns an $s'$-sparse solution $\boldsymbol{x}^T$ such that $f(\boldsymbol{x}^T) \leq f(\boldsymbol{x}^*) + \varepsilon$. Furthermore, each iteration requires $O(1)$ evaluations of $f$, $\nabla f$, and $O(n)$ additional time.*

[1]The theorem also holds if the smoothness and strong convexity constants are replaced by $(s'+s)$-*restricted* smoothness and strong convexity constants.

To achieve this result, we significantly refine and generalize the *adaptive regularization* technique of [14]. This refined version fixes many of the shortcomings of the original, by (i) not requiring *re-optimization* in every iteration (a relic of OMP-style algorithms), (ii) taking $\widetilde{O}(\kappa)$ instead of $\widetilde{O}(s\kappa)$ iterations, (iii) being *deterministic*, (iv) not requiring knowledge of the optimal function value $f(x^*)$ thus avoiding the overhead of an outer binary search, and (v) being more easily generalizable to other settings, like low rank minimization.

In short, our main idea is to run IHT on a regularized function

$$g(\boldsymbol{x}) = f(\boldsymbol{x}) + (\beta/2)\left\|\boldsymbol{x}\right\|_{\boldsymbol{w},2}^2 ,$$

where $\left\|\boldsymbol{x}\right\|_{\boldsymbol{w},2}^2 = \sum_{i=1}^n w_i x_i^2$ and $\boldsymbol{w}$ are non-negative weights. These weights change dynamically during the algorithm, in a way that depends on the value of $\boldsymbol{x}$. The effect is that now the IHT step will instead be given by

$$\boldsymbol{x}^{t+1} = H_{s'}\left(\left(\mathbf{1} - 0.5\boldsymbol{w}^t\right)\boldsymbol{x}^t - \eta \cdot \nabla f(\boldsymbol{x}^t)\right) ,$$

which is almost the same as (7.2), except that it has an extra term that biases the solution towards $\mathbf{0}$. Additionally, in each step the weights $\boldsymbol{w}^t$ are updated based on the current solution as

$$w_i^{t+1} = \left(w_i^t \cdot \left(\mathbf{1} - c \cdot \frac{w_i^t (x_i^t)^2}{\left\|\boldsymbol{x}^t\right\|_{w^t,2}^2}\right)\right)_{\geq 1/2}$$

for some parameter $c > 0$, where $(\cdot)_{\geq 1/2}$ denotes zeroing out all the entries that are $< 1/2$ and keeping the others intact.

In Section 7.2, we will go over the central ideas of our refined adaptive regularization technique, and also explain how it can be extended to deal with more general sparsity measures.

### 7.1.3 Beyond Sparsity: Low Rank Optimization

As discussed, our new techniques transfer to the problem of minimizing a convex function under a rank constraint. In particular, we prove the following theorem:

**Theorem 7.1.2** (Adaptive Regularization for Low Rank Optimization)**.** *Let $f \in \mathbb{R}^{m\times n} \to \mathbb{R}$ be a convex function with condition number $\kappa$ and consider the low rank minimization problem*

$$\min_{\mathrm{rank}(\boldsymbol{A})\leq r} f(\boldsymbol{A}) . \tag{7.3}$$

*For any error parameter $\varepsilon > 0$, there exists a polynomial time algorithm that returns a matrix $\boldsymbol{A}$ with $\mathrm{rank}(\boldsymbol{A}) \leq O\left(r\left(\kappa + \log\frac{f(\boldsymbol{O})-f(\boldsymbol{A}^*)}{\varepsilon}\right)\right)$ and $f(\boldsymbol{A}) \leq f(\boldsymbol{A}^*) + \varepsilon$, where $\boldsymbol{O}$ is the all-zero matrix and $\boldsymbol{A}^*$ is any rank-$r$ matrix.*

This result can be compared to the Greedy algorithm of [13], which works by incrementally adding a rank-1 component to the solution and achieves rank $O(r\kappa\log\frac{f(\boldsymbol{O})-f(\boldsymbol{A}^*)}{\varepsilon})$, as well as their Local Search algorithm, which works by simultaneously adding a rank-1 component and removing another, and achieves rank $O(r\kappa^2)$. In contrast, our Theorem 7.1.2 returns a solution with rank $O\left(r\left(\kappa + \log\frac{f(\boldsymbol{O})-f(\boldsymbol{A}^*)}{\varepsilon}\right)\right)$.

### 7.1.4 Related Work

The sparse optimization and compressed sensing literature has a wealth of different algorithms and analyses. Examples include the seminal paper of [31] on recovery with LASSO and followup works [64], the CoSaMP algorithm [128], orthogonal matching pursuit and variants [127, 146, 87, 14] iterative hard thresholding [19, 89], hard thresholding pursuit [65, 169, 148, 149], partial hard thresholding [88], and message passing algorithms [48]. For a survey on compressed sensing, see [21, 68].

A family of algorithms that is closely related to IHT are Frank-Wolfe (FW) methods [69], which have been used for dealing with generalized sparsity constraints [86]. The basic version can be viewed as a variant of OMP without re-optimization in each iteration. Block-FW methods are more resemblant of IHT without the projection step, see e.g. [5] for an application to the low rank minimization problem.

[112] presented an interesting connection between hard and soft thresholding algorithms by studying a concavity property of the thresholding operator, and proposed new thresholding operators.

Recently it has been shown [138] that IHT can be guaranteed to work for sparse optimization of *non-convex* functions, under appropriate assumptions. In particular, [138] studies a stochastic version of IHT for sparse deep learning problems, from both a theoretical and practical standpoint.

## 7.2 The Adaptive Regularization Method

Consider the sparse optimization problem

$$\min_{\|\boldsymbol{x}\|_0 \leq s} f(\boldsymbol{x}) \tag{7.4}$$

on a convex function $f$ with condition number at most $\kappa$, and an optimal solution $\boldsymbol{x}^*$ that is supported on the set of indices $S^* \subseteq [n]$.

The main hurdle towards solving this problem is that it is NP hard. Therefore, it is common to relax it by a factor depending on $\kappa$. In fact, IHT requires relaxing the sparsity constraint by a factor of $O(\kappa^2)$ (i.e. $\|\boldsymbol{x}\|_0 \leq O(s\kappa^2)$), in order to return a near-optimal solution. Also, the $\kappa^2$ factor is tight for IHT (see Appendix 9.5.5).

**Remark** We state all our results in terms of the condition number $\kappa$, even though the statements can be strengthened to depend on the *restricted* condition number $\kappa_{s'+s}$, specifically the condition number restricted on $(s'+s)$-sparse directions. We state our results in this weaker form for clarity of presentation.

### 7.2.1 Regularized IHT

Perhaps surprisingly, there is a way to *regularize* the objective by a weighted $\ell_2$ norm so that running IHT on the new objective will only require relaxing the sparsity by $O(\kappa)$:

$$\min_{\|\boldsymbol{x}\|_0 \leq s} f(\boldsymbol{x}) + (\beta/2) \|\boldsymbol{x}\|_{\boldsymbol{w},2}^2 . \tag{7.5}$$

One way to do this is by setting the weights $\boldsymbol{w}$ to be 1 everywhere except in the indices from $S^*$, where it is set to 0. An inquisitive reader will protest that this is not a very useful statement, since it requires knowledge of $S^*$, which was our goal to begin with. In fact, we could just as easily have used the regularizer $(\beta/2) \|\boldsymbol{x} - \boldsymbol{x}^*\|_2^2$, thus penalizing everything that is far from the optimum!

### 7.2.2 Learning Weights

Our main contribution is to show that the optimal weights $\boldsymbol{w}$ can in fact be *learned* in the duration of the algorithm[2]. More precisely, consider running IHT starting from the setting of $\boldsymbol{w} = \mathbf{1}$. The regularized objective (7.5) is now $O(1)$-conditioned, which is great news. On the other hand, (7.5) is not what we set out to minimize. In other words, even though this approach might work great for minimizing (7.5), it might (and generally will) fail to achieve sufficient decrease in (7.4)—one could view this as the algorithm getting trapped in a local minimum.

Our main technical tool is to characterize these local minima, by showing that they can only manifest themselves if the current solution $\boldsymbol{x}$ satisfies the following condition:

$$\|\boldsymbol{x}_{S^*}\|_{\boldsymbol{w},2}^2 \geq \Omega(\kappa^{-1}) \|\boldsymbol{x}\|_{\boldsymbol{w},2}^2 \,. \tag{7.6}$$

In words, this means that a significant fraction of the mass of the current solution lies in the support $S^*$ of the optimal solution. Interestingly, this gives us enough information based on which to update the regularization weights $\boldsymbol{w}$ in a way that the sum of weights in $S^*$ drops fast enough compared to the total sum of weights. This implies that the vector $\boldsymbol{w}$ moves in a direction that correlates with the direction of the optimal weight vector.

These are the core ideas needed to bring the sparsity overhead of IHT from $O(\kappa^2)$ down to $O(\kappa)$.

### 7.2.3 Beyond Sparsity: Learning Subspaces

One can summarize the approach of the previous section in the following more general way: If we know that the optimal solution $\boldsymbol{x}^*$ lies in a particular low-dimensional subspace (in our case this was the span of $\mathbf{1}_i$ for all $i \in S^*$), then we can define a regularization term that penalizes all the solutions based on their distance to that subspace. Of course, this subspace is unknown to us, but we can try to adaptively modify the regularization term every time the algorithm gets stuck, just as we did in the previous section.

More concretely, given a collection $\mathcal{A}$ of unit vectors from $\mathbb{R}^n$ (commonly called *atoms*), we define the following problem:

$$\min_{\mathrm{rank}_{\mathcal{A}}(\boldsymbol{x}) \leq r} f(\boldsymbol{x}) \,, \tag{7.7}$$

where $\mathrm{rank}_{\mathcal{A}}(\boldsymbol{x})$ is the smallest number of vectors from $\mathcal{A}$ such that $\boldsymbol{x}$ can be written as their linear combination. We can pick $\mathcal{A} = \{\mathbf{1}_1, \mathbf{1}_2, \ldots, \mathbf{1}_n\}$ to obtain the sparse optimization problem, $\mathcal{A} = \{\mathrm{vec}(\boldsymbol{u}\boldsymbol{v}^\top) \mid \|\boldsymbol{u}\|_2 = \|\boldsymbol{v}\|_2 = 1\}$ for the low rank minimization problem, and other choices of $\mathcal{A}$ can capture more sophisticated problem constraints such as graph structure. Defining an IHT variant for these more general settings is usually straightforward, although the analysis for even obtaining a rank overhead of $O(\kappa^2)$ does not trivially follow and depends on the structure of $\mathcal{A}$.

So, how would a regularizer look in this more general setting? Given our above discussion, it is fairly simple to deduce it. Consider a decomposition of $\boldsymbol{x}$ as the sum of rank-1 components from $\mathcal{A} = \{\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_{|\mathcal{A}|}\}$:

$$\boldsymbol{x} = \sum_{i \in S} \boldsymbol{a}_i \,,$$

[2]The idea of adaptively learning regularization weights looks on the surface similar to adaptive gradient algorithms such as AdaGrad [50]. An important difference is that these algorithms regularize the function around the current solution, while we regularize it around the origin. Still, this is a potentially intriguing connection that deserves to be investigated further.

where $\text{rank}_{\mathcal{A}}(\boldsymbol{a}_i) = 1$, and let $L^* = \text{span}(\{\boldsymbol{a}_i \mid i \in S^*\})$ be a low-dimensional subspace that contains the optimal solution and $L^*_\perp$ is its complement. We can then define the regularizer

$$\Phi^*(\boldsymbol{x}) = (\beta/2) \sum_{i \in S} \left\| \boldsymbol{\Pi}_{L^*_\perp} \boldsymbol{a}_i \right\|_2^2 ,$$

where $\boldsymbol{\Pi}_{L^*_\perp}$ is the orthogonal projection onto the subspace perpendicular to $L^*$—in other words $\left\| \boldsymbol{\Pi}_{L^*_\perp} \boldsymbol{a}_i \right\|_2$ is the $\ell_2$ distance from $\boldsymbol{a}_i$ to $L^*$. An equivalent but slightly more concise way is to write:

$$\Phi^*(\boldsymbol{x}) = (\beta/2) \left\langle \boldsymbol{\Pi}_{L^*_\perp}, \sum_{i \in S} \boldsymbol{a}_i \boldsymbol{a}_i^\top \right\rangle .$$

Then, we can replace the unknown projection matrix $\boldsymbol{\Pi}_{L^*_\perp}$ by a weight matrix $\boldsymbol{W}$ initialized at $\boldsymbol{I}$, and proceed by adaptively modifying $\boldsymbol{W}$ as we did in the previous section.

It should be noted that the full analysis of this framework is not automatic for general $\mathcal{A}$, and there are several technical challenges that arise depending on the choice of $\mathcal{A}$. In particular, it does not directly apply to the low rank minimization case, and we end up using a different choice of regularizer. However, the discussion in this section should serve as a basic framework for improving the IHT analysis in more general settings, as in particular it did to motivate the low rank optimization analysis that we will present in Section 7.4.

## 7.3 Sparse Optimization Using Regularized IHT

The main result of this section is an efficient algorithm for sparse optimization of convex functions that, even though is a slight modification of IHT, improves the sparsity by an $O(\kappa)$ factor, where $\kappa$ is the condition number. The regularized IHT algorithm is presented in Algorithm 16 and its analysis is in Theorem 7.1.1, whose proof can be found in Appendix 9.5.2.

**Algorithm 16** Regularized IHT

$\boldsymbol{x}^0$: initial $s'$-sparse solution
$\boldsymbol{w}^0 = \boldsymbol{1}$: initial regularization weights
$\eta$: step size, $T$: #iterations
$c$: weight step size
**for** $t = 0 \dots T-1$ **do**
  $\boldsymbol{x}^{t+1} = H_{s'}\left((\boldsymbol{1} - 0.5\boldsymbol{w}^t)\boldsymbol{x}^t - \eta \cdot \nabla f(\boldsymbol{x}^t)\right)$
  $\boldsymbol{w}^{t+1} = \left(\boldsymbol{w}^t - c \cdot (\boldsymbol{w}^t \boldsymbol{x}^t)^2 / \left\|\boldsymbol{x}^t\right\|^2_{\boldsymbol{w}^t,2}\right)_{\geq 1/2}$
  **if** $f(\boldsymbol{x}^{t+1}) + (4\eta)^{-1}\left\|\boldsymbol{x}^{t+1}\right\|^2_{\boldsymbol{w}^{t+1},2} > f(\boldsymbol{x}^t) + (4\eta)^{-1}\left\|\boldsymbol{x}^t\right\|^2_{\boldsymbol{w}^{t+1},2}$ **then**
    $\boldsymbol{x}^{t+1} = \boldsymbol{x}^t$ ▷ In practice there is no need to perform this step.

**Theorem 7.1.1** (Regularized IHT)**.** *Let $f \in \mathbb{R}^n \to \mathbb{R}$ be a convex function that is $\beta$-smooth and $\alpha$-strongly convex, with condition number $\kappa = \beta/\alpha$, and $\boldsymbol{x}^*$ be an (unknown) $s$-sparse solution. Then, running Algorithm 16 with $\eta = (2\beta)^{-1}$ and $c = s'/(4T)$ for*

$$T = O\left(\kappa \log \frac{f(\boldsymbol{x}^0) + (\beta/2)\left\|\boldsymbol{x}^0\right\|_2^2 - f(\boldsymbol{x}^*)}{\varepsilon}\right)$$

*iterations starting from an arbitrary $s' = O(s\kappa)$-sparse solution $\boldsymbol{x}^0$, the algorithm returns an $s'$-sparse solution $\boldsymbol{x}^T$ such that $f(\boldsymbol{x}^T) \leq f(\boldsymbol{x}^*) + \varepsilon$. Furthermore, each iteration requires $O(1)$ evaluations of $f$, $\nabla f$, and $O(n)$ additional time.*

The main ingredient for proving Theorem 7.1.1 is Lemma 7.3.1, which states that each step of the algorithm either makes substantial (multiplicative) progress in an appropriately regularized function $f(\boldsymbol{x}) + (\beta/2)\|\boldsymbol{x}\|^2_{\boldsymbol{w},2}$, or a significant fraction of the mass of $\boldsymbol{x}^2$ lies in $S^*$, which is the support of the target solution. This latter condition allows us to adapt the weights $\boldsymbol{w}$ in order to obtain a new regularization function that penalizes the target solution less. The proof of the lemma can be found in Appendix 9.5.3.

**Lemma 7.3.1** (Regularized IHT step progress)**.** *Let $f \in \mathbb{R}^n \to \mathbb{R}$ be a convex function that is $\beta$-smooth and $\alpha$-strongly convex, $\kappa = \beta/\alpha$ be its condition number, and $\boldsymbol{x}^*$ be any $s$-sparse solution.*

*Given any $s'$-sparse solution $\boldsymbol{x} \in \mathbb{R}^n$ where*

$$s' \geq (128\kappa + 2)s$$

*and a weight vector $\boldsymbol{w} \in (\{0\} \cup [1/2, 1])^n$ such that $\|\boldsymbol{w}\|_1 \geq n - s'/2$, we make the following update:*

$$\boldsymbol{x}' = H_{s'}\left((\mathbf{1} - 0.5\boldsymbol{w})\boldsymbol{x} - (2\beta)^{-1}\nabla f(\boldsymbol{x})\right) .$$

*Then, at least one of the following two conditions holds:*

- *Updating $\boldsymbol{x}$ makes regularized progress:*

  $$g(\boldsymbol{x}') \leq g(\boldsymbol{x}) - (16\kappa)^{-1}(g(\boldsymbol{x}) - f(\boldsymbol{x}^*)) ,$$

  *where*

  $$g(\boldsymbol{x}) := f(\boldsymbol{x}) + (\beta/2)\|\boldsymbol{x}\|^2_{\boldsymbol{w},2}$$

  *is the $\ell_2$-regularized version of $f$ with weights given by $\boldsymbol{w}$. Note: The regularized progress statement is true as long as $\boldsymbol{x}$ is suboptimal, i.e. $g(\boldsymbol{x}) > f(\boldsymbol{x}^*)$. Otherwise, we just have $g(\boldsymbol{x}') \leq g(\boldsymbol{x})$.*

- *$\boldsymbol{x}$ is significantly correlated to the optimal support $S^* := \operatorname{supp}(\boldsymbol{x}^*)$:*

  $$\|\boldsymbol{x}_{S^*}\|^2_{\boldsymbol{w}^2,2} \geq (4\kappa + 6)^{-1}\|\boldsymbol{x}\|^2_{\boldsymbol{w},2} ,$$

  *and the regularization term restricted to $S^*$ is non-negligible:*

  $$(\beta/2)\|\boldsymbol{x}_{S^*}\|^2_{\boldsymbol{w}^2,2} \geq (8\kappa + 8)^{-1}(g(\boldsymbol{x}) - f(\boldsymbol{x}^*)) .$$

**Comparison to ARHT.** The ARHT algorithm of [14] is also able to achieve a sparsity bound of $O(s\kappa)$. However, their algorithm is not practically desirable for a variety of reasons.

- First of all, it follows the OMP (more accurately, OMP with Removals) paradigm, which makes local changes to the support of the solution by inserting or removing a single element of the support, and then *fully re-optimizing* the function on its restriction to this support. Even though the support will generally be very small compared to the ambient dimension $n$, this is still a significant runtime overhead. In contrast, regularized IHT does not require re-optimization.

Additionally, the fact that in the ARHT only one new element is added at a time leads to an iteration count that scales with $s\kappa$, instead of the $\kappa$ of regularized IHT. This is a significant speedup, since both algorithms have to evaluate the gradient in each iteration. Therefore, regularized IHT will require $O(s)$ times fewer gradient evaluations.

- When faced with the non-progress condition, in which the regularized function value does not decrease sufficiently, ARHT moves by selecting a random index $i$ with probability proportional to $x_i^2$, and proceeds to *unregularize* this element, i.e. remove it from the sum of regularization terms. Instead, our algorithm is completely deterministic. This is achieved by allowing a *weighted* regularization term, and gradually reducing the regularization weights instead of dropping terms.

- ARHT requires knowledge of the optimal function value $f(\boldsymbol{x}^*)$. The reason is that in each iteration they need to gauge whether enough progress was made in reducing the value of the regularized function $g$, compared to how far it is from the optimal function value. If so, they would perform the unregularization step. In contrast, our analysis does not require these two cases (updates to $\boldsymbol{x}$ or $\boldsymbol{w}$) to be exclusive, and in fact simultaneously updates both, regardless of how much progress was made in $g$. Thus, our algorithm avoids the expensive overhead of an outer binary search over the optimal value $f(\boldsymbol{x}^*)$.

For all these reasons, as well as its striking simplicity, we believe that regularized IHT can prove to be a useful practical sparse optimization tool.

## 7.4 Low Rank Optimization Using Regularized Local Search

In this section we present a regularized local search algorithm for low rank optimization of convex functions, that returns an $\varepsilon$-optimal solution with rank $O\left(r\left(\kappa + \log\frac{f(\boldsymbol{O})-f(\boldsymbol{A}^*)}{\varepsilon}\right)\right)$, where $r$ is the target rank. The algorithm is based on the Local Search algorithm of [13], but also uses adaptive regularization, which leads to a lot new technical hurdles that are addressed in the analysis. This is presented in Theorem 7.1.2 and proved in Appendix 9.5.4.

**Theorem 7.1.2** (Adaptive Regularization for Low Rank Optimization)**.** *Let $f \in \mathbb{R}^{m\times n} \to \mathbb{R}$ be a convex function with condition number $\kappa$ and consider the low rank minimization problem*

$$\min_{\mathrm{rank}(\boldsymbol{A})\leq r} f(\boldsymbol{A})\,. \tag{7.8}$$

*For any error parameter $\varepsilon > 0$, there exists a polynomial time algorithm that returns a matrix $\boldsymbol{A}$ with $\mathrm{rank}(\boldsymbol{A}) \leq O\left(r\left(\kappa + \log\frac{f(\boldsymbol{O})-f(\boldsymbol{A}^*)}{\varepsilon}\right)\right)$ and $f(\boldsymbol{A}) \leq f(\boldsymbol{A}^*) + \varepsilon$, where $\boldsymbol{O}$ is the all-zero matrix and $\boldsymbol{A}^*$ is any rank-$r$ matrix.*

**Discussion about $\varepsilon$ dependence.** Some of the technical issues in the rank case have to do with operator *non-commutativity* and thus pose no issue in the sparsity case. In particular, the extra $\log\frac{f(\boldsymbol{O})-f(\boldsymbol{A}^*)}{\varepsilon}$ dependence in the rank comes exactly because of these issues. However, we think that it should be possible to completely remove this dependence in the future by a more careful analysis.

**Discussion about computational efficiency.** We note that the goal of this section is to show an improved rank bound, and not to argue about the computational efficiency of such an algorithm. It might be possible to derive an efficient algorithm by transforming the proof in Theorem 7.1.2 into

a proof for a matrix IHT algorithm, which might be significantly more efficient, as it will not require solving linear systems in each iteration. Still, there are a lot of remaining issues to be tackled, as currently the algorithm requires computing multiple singular value decompositions and orthogonal projections in each iteration. Therefore working on a computationally efficient algorithm that can guarantee a rank of $O(r\kappa)$ is a very interesting direction for future research.

**Matrix regularizer** Getting back into the main ingredients of Theorem 7.1.2, we describe the choice of our regularizer. As we are working over general rectangular matrices, we use *two* regularizers, one for the left singular vectors and one for the right singular vectors of $\boldsymbol{A}$. Concretely, given two weight matrices $\boldsymbol{Y}$, $\boldsymbol{W}$ such that $\boldsymbol{O} \preceq \boldsymbol{Y} \preceq \boldsymbol{I}$, $\boldsymbol{O} \preceq \boldsymbol{W} \preceq \boldsymbol{I}$, we define

$$\Phi(\boldsymbol{A}) = (\beta/4)\left(\langle \boldsymbol{W}, \boldsymbol{A}\boldsymbol{A}^\top\rangle + \langle \boldsymbol{Y}, \boldsymbol{A}^\top\boldsymbol{A}\rangle\right),$$

where $\beta$ is a bound on the smoothness of $f$. The gradient of the regularized function is

$$\nabla g(\boldsymbol{A}) = \nabla f(\boldsymbol{A}) + (\beta/2)\left(\boldsymbol{W}\boldsymbol{A} + \boldsymbol{A}\boldsymbol{Y}\right),$$

and the new solution $\bar{A}$ is defined as

$$\bar{A} = H_{s'-1}\left(\boldsymbol{A}\right) - \eta H_1\left(\nabla g(\boldsymbol{A})\right),$$

where we remind that the thresholding operator $H_r : \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$ that is used in the algorithm returns the top $r$ components of the singular value decomposition of a matrix, i.e. given $\boldsymbol{M} = \sum_{i=1}^{k} \lambda_i \boldsymbol{u}_i \boldsymbol{v}_i^\top$, where $\boldsymbol{\lambda}_1 \geq \cdots \geq \boldsymbol{\lambda}_k$ are the singular values and $r \leq k$, $H_r(\boldsymbol{M}) = \sum_{i=1}^{r} \lambda_i \boldsymbol{u}_i \boldsymbol{v}_i^\top$. In other words, we drop the bottom rank-1 component of $\boldsymbol{A}$ and add the top rank-1 component of the gradient.

After taking a step, we re-optimize over matrices with the current left and right singular space, also known as performing a fully corrective step, as in [144, 13]. To do this, we first compute the SVD $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ of $\bar{A}$ and then solve the optimization problem $\min_{\boldsymbol{A}=\boldsymbol{U}\boldsymbol{X}\boldsymbol{V}^\top} g^t(\boldsymbol{A})$. For simplicity we assume that this optimization problem can be solved exactly, but the analysis can be modified to account for the case when we have an approximate solution and we are only given a bound on the norm of the gradient (projected onto the relevant subspace), i.e. $\left\|\boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{U})}\nabla g^t(\boldsymbol{A})\boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V})}\right\|_F$.

Whenever there is not enough progress, we make the following updates on the weight matrices $\boldsymbol{W}$ and $\boldsymbol{Y}$:

$$\boldsymbol{W}' = \boldsymbol{W} - \boldsymbol{W}\boldsymbol{A}\boldsymbol{A}^\top\boldsymbol{W}/\langle \boldsymbol{W}, \boldsymbol{A}\boldsymbol{A}^\top\rangle$$
$$\boldsymbol{Y}' = \boldsymbol{Y} - \boldsymbol{Y}\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{Y}/\langle \boldsymbol{Y}, \boldsymbol{A}^\top\boldsymbol{A}\rangle.$$

The full algorithm description is in Algorithm 25, Appendix 9.5.4. In the algorithm description we assume that $f(\boldsymbol{A}^*)$ is known. This assumption can be removed by performing binary search over this value, as in [14].

## 7.5 Experiments

**Introduction.** In this section we present numerical experiments in order to compare the performance of IHT and regularized IHT (Algorithm 16) in training sparse linear models. In particular, we will look at the tasks of linear regression and logistic regression using both real and synthetic data.

In the former, we are given a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, where each row represents an example and each column a feature, and a vector $\boldsymbol{b} \in \mathbb{R}^m$ that represents the ground truth outputs, and our objective is to minimize the $\ell_2$ loss

$$(1/2) \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 .$$

In logistic regression, $\boldsymbol{b}$ has binary instead of real entries, and our objective is to minimize the logistic loss

$$\underbrace{-\sum_{i=1}^{m} (b_i \log \sigma(\boldsymbol{A}\boldsymbol{x})_i + (1 - b_i) \log(1 - \sigma(\boldsymbol{A}\boldsymbol{x})_i))}_{l(\boldsymbol{x})} ,$$

where $\sigma(z) = (1 + e^{-z})^{-1}$ is the sigmoid function. As is common, we look at the *regularized* logistic regression objective:

$$l(\boldsymbol{x}) + (\rho/2) \|\boldsymbol{x}\|_2^2 ,$$

for some $\rho > 0$. For our experiments we use $\rho = 0.1$.

**Preprocessing and choice of parameters.** The only preprocessing we perform is to center the columns of $\boldsymbol{A}$, i.e. we subtract the mean of each column from each entry of the column, and then scale the columns to unit $\ell_2$ norm. This ensures that for any sparsity parameter $s' \in [n]$, the function $f$ is $s'$-smooth when restricted to $s'$-sparse directions, or in other words the $s'$-restricted smoothness constant of $f$ is at most $s'$. Thus we set our smoothness estimate to $\beta := s'$. Our smoothness estimate $\beta$ influences the (regularized) IHT algorithm in two ways. First, as the step size of the algorithm is given by $1/\beta$, a value of $\beta$ that is too large can slow down the algorithm, or even get it stuck to a local minimum. Second, the strength of the regularization term in regularized IHT should be close to the $(s + s')$-restricted smoothness constant, as shown in the analysis of Theorem 7.1.1.

Even though having a perfectly accurate estimate of the smoothness constant is not necessary, a more accurate estimate improves the performance of the algorithm. In fact, the estimate $1/s'$ for the step size is generally too conservative. When used in practice, one should either tune this parameter or use a variable/adaptive step size to achieve the best results.

For the weight step size of regularized IHT, we set the weight step size to $c = s'/T$, but we also experiment with how changing $c$ affects the performance of the algorithm. The downside of this setting is that it requires knowing the number of iterations a priori. However, in practice one could tune $c$ and then run the algorithm for $O(s'/c)$ iterations. Note that ideally, based on the theoretical analysis, $T$ would be proportional to the restricted condition number of $f$, however this quantity is hard to compute in general. Another idea to avoid this in practice could be to let $c$ be a variable step size.

**Implementation.** Both the IHT and regularized IHT algorithms are incredibly simple, and can be described in a few lines of python code, as can be seen in Figure 9-1, Appendix 9.5.1. Note that in comparison to Algorithm 16 we do not perform the conditional assignment. All the experiments were run on a single 2.6GHz Intel Core i7 core of a 2019 MacBook Pro with 16GB DDR4 RAM using Python 3.9.10.

### 7.5.1 Real data

We first experiment with real data, specifically the *year* regression dataset from UCI [49] and the *rcv1* binary classification dataset [108], which have been previously used in the literature. Performance on other datasets was similar. In Figure 7-1 **(a)-(b)** we have a comparison between the error of the solution returned by IHT and regularized IHT for a fixed sparsity level. Specifically, if we let $\boldsymbol{x}^{**}$ be the (dense) global minimizer of $f$, we plot the logarithm of the (normalized) *excess loss* $(f(\boldsymbol{x}) - f(\boldsymbol{x}^{**}))/f(\boldsymbol{0})$ against the number of iterations. Note that $f(\boldsymbol{x}^{**})$ will typically be considerably lower than the loss of the sparse optimum $f(\boldsymbol{x}^*)$. In order to make a fair comparison, for each algorithm we pick the best fixed step size of the form $2^i/s$ for integer $i \geq 0$, where $s$ is the fixed sparsity level. The best step sizes of IHT and regularized IHT end up being $2/s, 4/s$ respectively for the linear regression example, and $8/s, 16/s$ respectively for the logistic regression example.

We notice that initially regularized IHT has a much higher error than IHT, but after some iterations it is lower than IHT. This phenomenon is to be expected, because the algorithm runs on a regularized function, and so tries to keep not just $f(\boldsymbol{x})$ but also $\|\boldsymbol{x}\|_2^2$ small. After some iterations, when the algorithm has learnt regularization weights that are closer to the optimal ones, it converges to sparser solutions than IHT (equivalently, lower error solutions with the same sparsity, which is what is shown in the plot).

In Figure 7-1 **(c)** we compare IHT and regularized IHT for different sparsity levels on the year dataset. If $e_1$ and $e_2$ are the excess errors of IHT and regularized IHT respectively, we plot $e_2/e_1$, which is the relative excess error of regularized IHT with respect to that of IHT. We notice a reduction of up to 40% on the excess error. In Figure 7-1 **(d)** we examine the effect of the choice of the weight step size $c$. We conclude that $c$ can give a tradeoff between runtime and accuracy, as setting it to a large value will lead to faster weight decay and thus resemble IHT, while a small value of $c$ will lead to slow weight decrease, which will lead to more iterations but also potentially recover an improved solution. Here we can see an interesting tradeoff between the number of iterations and the error of the solution that is eventually returned. In particular, the *larger* $c$ is, the *faster* the degradation of regularization weights. Thus, for $c \to \infty$, the algorithm tends to be the same as IHT. On the other hand, with *smaller* values of $c$, one can get an improved error rate, but at the cost of a larger number of iterations. This is because the regularization weights decrease slowly, and so in the early iterations of the algorithm the regularization term will account for a significant fraction of the objective function value.

### 7.5.2 Synthetic data

We now turn to synthetically generated linear regression instances. The first result presented in Figure 7-2 **(a)** is the hard IHT instance that we derived in our lower bound in Appendix 9.5.5. This experiment shows that there exist examples where, with bad initialization, IHT cannot decrease the objective at all (i.e. is stuck at a local minimum), while regularized IHT with the same initialization manages to reduce the loss by more than 70%.

The second result is a result in the well known setting of sparse signal recovery from linear measurements. We generate a matrix $\boldsymbol{A}$ with entries that are sampled i.i.d. from the standard normal distribution, an $s$-sparse signal $\boldsymbol{x}$ again with entries sampled i.i.d. from the standard normal distribution, and an observed vector $\boldsymbol{b} := \boldsymbol{A}\boldsymbol{x}$. The goal is to recover $\boldsymbol{x}$ by minimizing the objective

$$f(\boldsymbol{x}) = (1/2)\,\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2\,.$$

In Figure 7-2 **(b)**, we plot the normalized value of this objective, after running both IHT and regularized IHT for the same number of iterations. Here we pick the best step size per instance,

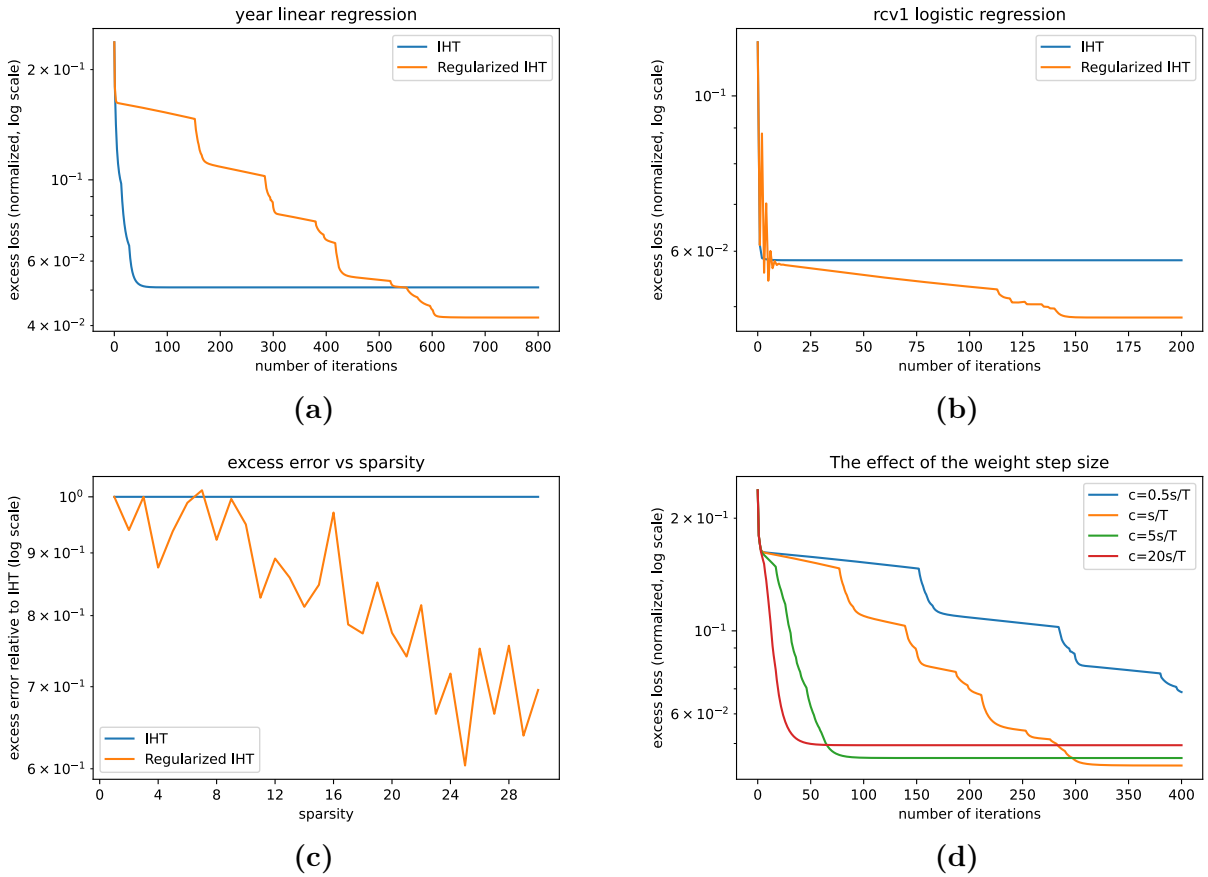
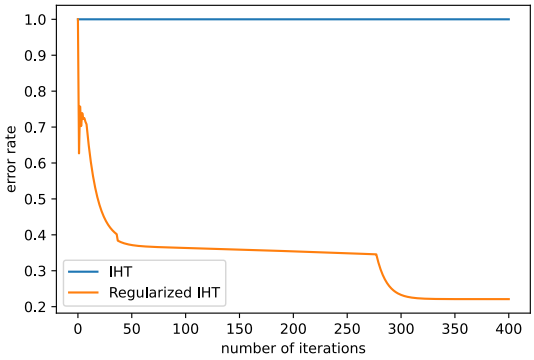
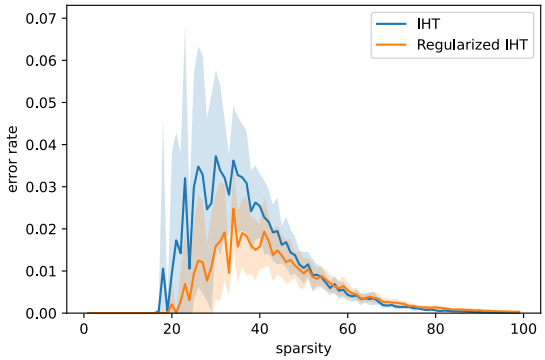



Figure 7-1: **(a)-(b)**: IHT vs Regularized IHT performance on the year and rcv1 datasets with fixed sparsity levels $s = 11$ and $s = 10$ respectively. On the $x$ axis we have number of iterations and on the $y$ axis we have the normalized excess loss (compared to the dense global optimum), in log scale. The excess loss of regularized IHT is less than that of IHT, specifically 17.3% and 17.2% respectively less in the two experiments. **(c):** Excess error of regularized IHT relative to IHT in the *year* dataset, where sparsity values range from 1 to 30. Both algorithms are run for $T = 800$ iterations. **(d):** Error rate vs number of iterations of regularized IHT on the *year* dataset with fixed sparsity $s = 11$ and step size $\eta = 4/s$, using different values for the weight step size $c$.




**(a)**




**(b)**

Figure 7-2: **(a):** A demonstration of a 80% decrease in loss by using regularized IHT instead of IHT on the hard instance for IHT presented in Section 9.5.5. We have generated the data with a condition number of $\kappa = 20$, and a planted sparse solution with sparsity $s = 2$. The dimension is $n = 842$. It can be observed that, for the given initialization vector, IHT never makes any progress on decreasing the error. In contrast, regularized IHT is able to decrease it by almost a factor of 5. **(b):** Sparse signal recovery, where $\boldsymbol{A}$ is an $100 \times 800$ measurement matrix, the sparsity level ranges from 1 to 100, and each algorithm is run for 240 iterations. Bands of 1 standard error are shown, after running each data point 20 times independently.

starting from $\eta = 1/s$ and increasing in multiples of 1.2. Also, for each fixed value of $s$ and algorithm, we run the experiments 20 times in order to account for the variance. The results show a superiority in the performance of regularized IHT for the sparse signal recovery task.

# Chapter 8

# Local Search Algorithms for Rank-Constrained Convex Optimization

## 8.1 Introduction

Given a real-valued convex function $R : \mathbb{R}^{m\times n} \to \mathbb{R}$ on real matrices and a parameter $r^* \in \mathbb{N}$, the *rank-constrained convex optimization* problem consists of finding a matrix $A \in \mathbb{R}^{m\times n}$ that minimizes $R(A)$ among all matrices of rank at most $r^*$:

$$\min_{\text{rank}(A)\leq r^*} R(A) \tag{8.1}$$

Even though $R$ is convex, the rank constraint makes this problem non-convex. Furthermore, it is known that this problem is NP-hard and even hard to approximate ([127, 63]).

In this work, we propose efficient greedy and local search algorithms for this problem. Our contribution is twofold:

1. We provide theoretical analyses that bound the rank and objective value of the solutions returned by the two algorithms in terms of the *rank-restricted condition number*, which is the natural generalization of the condition number for low rank subspaces. The results are significantly stronger than previous known bounds for this problem.

2. We experimentally demonstrate that, after careful performance adjustments, the proposed general-purpose greedy and local search algorithms have superior performance to other methods, even for some of those that are tailored to a particular problem. Thus, these algorithms can be considered as a general tool for rank-constrained convex optimization and a viable alternative to methods that use convex relaxations or alternating minimization.

**The rank-restricted condition number** Similarly to the work in sparse convex optimization, a restricted condition number quantity has been introduced as a reasonable assumption on $R$. If we let $\rho_r^+$ be the maximum smoothness bound and $\rho_r^-$ be the minimum strong convexity bound only along rank-$r$ directions of $R$ (these are called rank-restricted smoothness and strong convexity respectively), the rank-restricted condition number is defined as $\kappa_r = \frac{\rho_r^+}{\rho_r^-}$. If this quantity is bounded, one can efficiently find a solution $A$ with $R(A) \leq R(A^*) + \varepsilon$ and rank $r = O(r^* \cdot \kappa_{r+r^*} \frac{R(\mathbf{0})}{\varepsilon})$ using a greedy algorithm ([144]). However, this is not an ideal bound since the rank scales linearly with $\frac{R(\mathbf{0})}{\varepsilon}$, which can be particularly high in practice. Inspired by the analogous literature on sparse convex

optimization by [127, 146, 170, 89] and more recently [14], one would hope to achieve a logarithmic dependence or no dependence at all on $\frac{R(\mathbf{0})}{\varepsilon}$. In this chapter we achieve this goal by providing an improved analysis showing that the greedy algorithm of [144] in fact returns a matrix of rank of $r = O(r^* \cdot \kappa_{r+r^*} \log \frac{R(\mathbf{0})}{\varepsilon})$. We also provide a new local search algorithm together with an analysis guaranteeing a rank of $r = O(r^* \cdot \kappa^2_{r+r^*})$. Apart from significantly improving upon previous work on rank-restricted convex optimization, these results directly generalize a lot of work in sparse convex optimization, e.g. [127, 146, 89]. Our algorithms and theorem statements can be found in Section 8.2.

**Runtime improvements** Even though the rank bound guaranteed by our theoretical analyses is adequate, the algorithm runtimes leave much to be desired. In particular, both the greedy algorithm of [144] and our local search algorithm have to solve an optimization problem in each iteration in order to find the best possible linear combination of features added so far. Even for the case that $R(A) = \frac{1}{2} \sum_{(i,j)\in\Omega} (M - A)^2_{ij}$, this requires solving a least squares problem on $|\Omega|$ examples and $r^2$ variables. For practical implementations of these algorithms, we circumvent this issue by solving a related optimization problem that is usually much smaller. This instead requires solving $n$ least squares problems with *total* number of examples $|\Omega|$, each on $r$ variables. This not only reduces the size of the problem by a factor of $r$, but also allows for a straightforward distributed implementation. Interestingly, our theoretical analyses still hold for these variants. We propose an additional heuristic that reduces the runtime even more drastically, which is to only run a few (less than 10) iterations of the algorithm used for solving the inner optimization problem. Experimental results show that this modification not only does not significantly worsen results, but for machine learning applications also acts as a regularization method that can dramatically improve generalization. These matters, as well as additional improvements for making the local search algorithm more practical, are addressed in Section 8.2.3.

**Roadmap** In Section 8.2, we provide the descriptions and theoretical results for the algorithms used, along with several modifications to boost performance. In Section 8.3, we evaluate the proposed greedy and local search algorithms on optimization problems like robust PCA. Then, in Section 8.4 we evaluate their generalization performance in machine learning problems like matrix completion.

## 8.2 Algorithms & Theoretical Guarantees

In Sections 8.2.1 and 8.2.2 we state and provide theoretical performance guarantees for the basic greedy and local search algorithms respectively. Then in Section 8.2.3 we state the algorithmic adjustments that we propose in order to make the algorithms efficient in terms of runtime and generalization performance. A discussion regarding the tightness of the theoretical analysis is deferred to Appendix 9.6.

When the dimension is clear from context, we will denote the all-ones vector by $\mathbf{1}$, and the vector that is 0 everywhere and 1 at position $i$ by $\mathbf{1}_i$. Given a matrix $A$, we denote by $\mathrm{im}(A)$ its column span. One notion that we will find useful is that of *singular value thresholding*. More specifically, given a rank-$k$ matrix $A \in \mathbb{R}^{m\times n}$ with SVD $\sum_{i=1}^{k} \sigma_i u^i v^{i\top}$ such that $\sigma_1 \geq \cdots \geq \sigma_k$, as well as an integer parameter $r \geq 1$, we define $H_r(A) = \sum_{i=1}^{r} \sigma_i u^i v^{i\top}$ to be the operator that truncates to the $r$ highest singular values of $A$.

### 8.2.1 Greedy

Algorithm 17 (Greedy) was first introduced in [144] as the GECO algorithm. It works by iteratively adding a rank-1 matrix to the current solution. This matrix is chosen as the rank-1 matrix that best approximates the gradient, i.e. the pair of singular vectors corresponding to the maximum singular value of the gradient. In each iteration, an additional procedure is run to optimize the combination of previously chosen singular vectors.

In [144] guarantee on the rank of the solution returned by the algorithm is $r^*\kappa_{r+r^*}\frac{R(\mathbf{0})}{\varepsilon}$. The main bottleneck in order to improve on the $\frac{R(\mathbf{0})}{\varepsilon}$ factor is the fact that the analysis is done in terms of the squared nuclear norm of the optimal solution. As the worst-case discrepancy between the squared nuclear norm and the rank is $R(\mathbf{0})/\varepsilon$, their bounds inherit this factor. Our analysis works directly with the rank, in the spirit of sparse optimization results (e.g. [144, 89, 14]). A challenge compared to these works is the need for a suitable notion of "intersection" between two sets of vectors. The main technical contribution of this work is to show that the orthogonal projection of one set of vectors into the span of the other is such a notion, and, based on this, to define a decomposition of the optimal solution that is used in the analysis.

**Algorithm 17** Greedy

**procedure** GREEDY($r \in \mathbb{N}$ : target rank)
  function to be minimized $R : \mathbb{R}^{m\times n} \to \mathbb{R}$
  $U \in \mathbb{R}^{m\times 0}$ ▷ Initially rank is zero
  $V \in \mathbb{R}^{n\times 0}$
  **for** $t = 0 \dots r-1$ **do**
    $\sigma uv^\top \leftarrow H_1(\nabla R(UV^\top))$ ▷ Max singular value $\sigma$ and corresp. singular vectors $u, v$
    $U \leftarrow \begin{pmatrix} U & u \end{pmatrix}$ ▷ Append new vectors as columns
    $V \leftarrow \begin{pmatrix} V & v \end{pmatrix}$
    $U, V \leftarrow$ OPTIMIZE$(U, V)$
  **return** $UV^\top$
**procedure** OPTIMIZE($U \in \mathbb{R}^{m\times r}, V \in \mathbb{R}^{n\times r}$)
  $X \leftarrow \operatorname{argmin}_{X\in\mathbb{R}^{r\times r}} R(UXV^\top)$
  **return** $UX, V$

**Theorem 8.2.1** (Algorithm 17 (greedy) analysis)**.** *Let $A^*$ be any fixed optimal solution of (8.1) for some function $R$ and rank bound $r^*$, and let $\varepsilon > 0$ be an error parameter. For any integer $r \geq 2r^* \cdot \kappa_{r+r^*} \log \frac{R(\mathbf{0})-R(A^*)}{\varepsilon}$, if we let $A =$ GREEDY$(r)$ be the solution returned by Algorithm 17, then $R(A) \leq R(A^*) + \varepsilon$. The number of iterations is $r$.*

The proof of Theorem 8.2.1 can be found in Appendix 9.6.

### 8.2.2 Local Search

One drawback of Algorithm 1 is that it increases the rank in each iteration. Algorithm 2 is a modification of Algorithm 1, in which the rank is truncated in each iteration. The advantage of Algorithm 2 compared to Algorithm 1 is that it is able to make progress without increasing the rank of A, while Algorithm 1 necessarily increases the rank in each iteration. More specifically, because of the greedy nature of Algorithm 1, some rank-1 components that have been added to A might become obsolete or have reduced benefit after a number of iterations. Algorithm 2 is able to identify such candidates and remove them, thus allowing it to continue making progress.

**Algorithm 18** Local Search

**procedure** LOCAL_SEARCH($r \in \mathbb{N}$ : target rank)
  function to be minimized $R : \mathbb{R}^{m \times n} \to \mathbb{R}$
  $U \leftarrow \mathbf{0}_{m \times r}$ ▷ Initialize with all-zero solution
  $V \leftarrow \mathbf{0}_{n \times r}$
  **for** $t = 0 \dots L-1$ **do** ▷ Run for $L$ iterations
    $\sigma u v^\top \leftarrow H_1(\nabla R(UV^\top))$ ▷ Max singular value $\sigma$ and corresp. singular vectors $u, v$
    $U, V \leftarrow$ TRUNCATE$(U, V)$ ▷ Reduce rank of $UV^\top$ by one
    $U \leftarrow \begin{pmatrix} U & u \end{pmatrix}$ ▷ Append new vectors as columns
    $V \leftarrow \begin{pmatrix} V & v \end{pmatrix}$
    $U, V \leftarrow$ OPTIMIZE$(U, V)$
  **return** $UV^\top$
**procedure** TRUNCATE($U \in \mathbb{R}^{m \times r}, V \in \mathbb{R}^{n \times r}$)
  $U\Sigma V^\top \leftarrow \mathrm{SVD}(H_{r-1}(UV^\top))$ ▷ Keep all but minimum singular value
  **return** $U\Sigma, V$

**Theorem 8.2.2** (Algorithm 18 (local search) analysis)**.** *Let $A^*$ be any fixed optimal solution of (8.1) for some function $R$ and rank bound $r^*$, and let $\varepsilon > 0$ be an error parameter. For any integer $r \geq r^* \cdot (1 + 8\kappa^2_{r+r^*})$, if we let $A =$ LOCAL_SEARCH$(r)$ be the solution returned by Algorithm 18, then $R(A) \leq R(A^*) + \varepsilon$. The number of iterations is $O\left(r^* \kappa_{r+r^*} \log \frac{R(\mathbf{0}) - R(A^*)}{\varepsilon}\right)$.*

The proof of Theorem 8.2.2 can be found in Appendix 9.6.

### 8.2.3 Algorithmic adjustments

**Inner optimization problem** The inner optimization problem that is used in both greedy and local search is:

$$\min_{X \in \mathbb{R}^{r \times r}} R(UXV^\top)\,. \tag{8.2}$$

It essentially finds the choice of matrices $U'$ and $V'$, with columns in the column span of $U$ and $V$ respectively, that minimizes $R(U'V'^\top)$. We, however, consider the following problem instead:

$$\min_{V \in \mathbb{R}^{n \times r}} R(UV^\top)\,. \tag{8.3}$$

Note that the solution recovered from (8.3) will never have worse objective value than the one recovered from (8.2), and that nothing in the analysis of the algorithms breaks. Importantly, (8.3) can usually be solved much more efficiently than (8.2). As an example, consider the following objective that appears in matrix completion: $R(A) = \frac{1}{2} \sum_{(i,j) \in \Omega} (M - A)^2_{ij}$ for some $\Omega \subseteq [m] \times [n]$. If we let $\Pi_\Omega(\cdot)$ be an operator that zeroes out all positions in the matrix that are not in $\Omega$, we have $\nabla R(A) = -\Pi_\Omega(M - A)$. The optimality condition of (8.2) now is $U^\top \Pi_\Omega(M - UXV^\top)V = \mathbf{0}$ and that of (8.3) is $U^\top \Pi_\Omega(M - UV^\top) = \mathbf{0}$. The former corresponds to a least squares linear regression problem with $|\Omega|$ examples and $r^2$ variables, while the latter can be decomposed into $n$ independent systems $U^\top \left( \sum_{i:(i,j) \in \Omega} \mathbf{1}_i \mathbf{1}_i^\top \right) UV^j = U^\top \Pi_\Omega\left(M \mathbf{1}_j\right)$, where the variable is $V^j$ which is the $j$-th column of $V$. The $j$-th of these systems corresponds to a least squares linear regression problem with

$|\{i : (i,j) \in \Omega\}|$ examples and $r$ variables. Note that the total number of examples in all systems is $\sum_{j \in [n]} |\{i : (i,j) \in \Omega\}| = |\Omega|$. The choice of $V$ here as the variable to be optimized is arbitrary. In particular, as can be seen in Algorithm 19, in practice we alternate between optimizing $U$ and $V$ in each iteration. It is worthy of mention that the OPTIMIZE_FAST procedure is basically the same as one step of the popular alternating minimization procedure for solving low rank problems. As a matter of fact, when our proposed algorithms are viewed from this lens, they can be seen as alternating minimization interleaved with rank-1 insertion and/or removal steps.

**Algorithm 19** Fast inner Optimization

**procedure** OPTIMIZE_FAST($U \in \mathbb{R}^{m \times r}, V \in \mathbb{R}^{n \times r}, t \in \mathbb{N}$ : iteration index of algorithm)
  **if** $t \bmod 2 = 0$ **then**
    $X \leftarrow \operatorname{argmin}_{X \in \mathbb{R}^{m \times r}} R(XV^\top)$
    **return** $X, V$
  **else**
    $X \leftarrow \operatorname{argmin}_{X \in \mathbb{R}^{n \times r}} R(UX^\top)$
    **return** $U, X$

**Singular value decomposition** As modern methods for computing the top entries of a singular value decomposition scale very well even for large sparse matrices ([120, 159, 163]), the "insertion" step of greedy and local search, in which the top entry of the SVD of the gradient is determined, is quite fast in practice. However, these methods are not suited for computing the *smallest* singular values and corresponding singular vectors, a step required for the local search algorithm that we propose. Therefore, in our practical implementations we opt to perfom the alternative step of directly removing one pair of vectors from the representation $UV^\top$. A simple approach is to go over all $r$ possible removals and pick the one that increases the objective by the least amount. A variation of this approach has been used by [144]. However, a much faster approach is to just pick the pair of vectors $U\mathbf{1}_i$, $V\mathbf{1}_i$ that minimizes $\|U\mathbf{1}_i\|_2 \|V\mathbf{1}_i\|_2$. This is the approach that we use, as can be seen in Algorithm 20.

**Algorithm 20** Fast rank reduction

**procedure** TRUNCATE_FAST($U \in \mathbb{R}^{m \times r}, V \in \mathbb{R}^{n \times r}$)
  $i \leftarrow \operatorname{argmin}_{i \in [r]} \|U\mathbf{1}_i\|_2 \|V\mathbf{1}_i\|_2$
  **return** $\left(U_{[m],[1,i-1]} \quad U_{[m],[i+1,r]}\right), \left(V_{[n],[1,i-1]} \quad V_{[n],[i+1,r]}\right)$ ▷ Remove column $i$

After the previous discussion, we are ready to state the fast versions of Algorithm 17 and Algorithm 18 that we use for our experiments. These are Algorithm 8.2.3 and Algorithm 21. Notice that we initialize Algorithm 21 with the solution of Algorithm 8.2.3 and we run it until the value of $R(\cdot)$ stops decreasing rather than for a fixed number of iterations.

**Algorithm 8.2.3** (Fast Greedy)**.** *The Fast Greedy algorithm is defined identically as Algorithm 17, with the only difference that it uses the* OPTIMIZE_FAST *routine as opposed to the* OPTIMIZE *routine.*

**Algorithm 21** Fast Local Search

**procedure** FAST_LOCAL_SEARCH($r \in \mathbb{N}$ : target rank)
  function to be minimized $R : \mathbb{R}^{m \times n} \to \mathbb{R}$
  $U, V \leftarrow$ solution returned by FAST_GREEDY($r$)
  **do**
    $U_{\text{prev}}, V_{\text{prev}} \leftarrow U, V$
    $\sigma u v^\top \leftarrow H_1(\nabla R(UV^\top))$ ▷ Max singular value $\sigma$ and corresp. singular vectors $u, v$
    $U, V \leftarrow$ TRUNCATE_FAST($U, V$) ▷ Reduce rank of $UV^\top$ by one
    $U \leftarrow \begin{pmatrix} U & u \end{pmatrix}$ ▷ Append new vectors as columns
    $V \leftarrow \begin{pmatrix} V & v \end{pmatrix}$
    $U, V \leftarrow$ OPTIMIZE_FAST($U, V, t$)
  **while** $R(UV^\top) < R(U_{\text{prev}} V_{\text{prev}}^\top)$
  **return** $U_{\text{prev}} V_{\text{prev}}^\top$

## 8.3 Optimization Applications

An immediate application of the above algorithms is in the problem of *low rank matrix recovery*. Given any convex distance measure between matrices $d : \mathbb{R}_{m \times n} \times \mathbb{R}_{m \times n} \to \mathbb{R}_{\geq 0}$, the goal is to find a low-rank matrix $A$ that matches a target matrix $M$ as well as possible in terms of $d$: $\min_{\text{rank}(A) \leq r^*} d(M, A)$ This problem captures a lot of different applications, some of which we go over in the following sections.

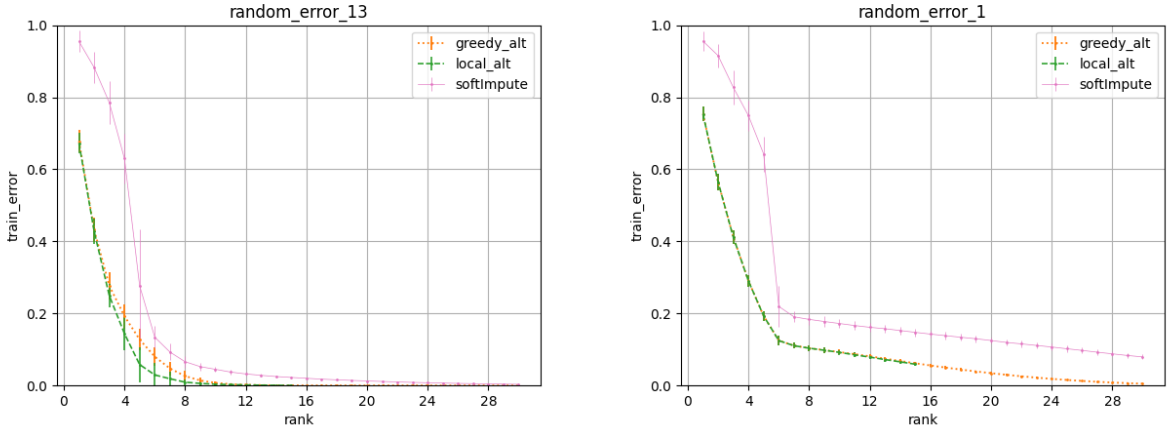
### 8.3.1 Low-rank approximation on observed set

A particular case of interest is when $d(M, A)$ is the Frobenius norm of $M - A$, but only applied to entries belonging to some set $\Omega$. In other words, $d(M, A) = \frac{1}{2}\|\Pi_\Omega(M - A)\|_F^2$ . We have compared our Fast Greedy and Fast Local Search algorithms with the SoftImpute algorithm of [121] as implemented by [141], on the same experiments as in [121]. We have solved the inner optimization problem required by our algorithms by the LSQR algorithm [134]. More specifically, $M = UV^\top + \eta \in \mathbb{R}^{100 \times 100}$, where $\eta$ is some noise vector. We let every entry of $U, V, \eta$ be i.i.d. normal with mean 0 and the entries of $\Omega$ are chosen i.i.d. uniformly at random over the set $[100] \times [100]$. The experiments have three parameters: The true rank $r^*$ (of $UV^\top$), the percentage of observed entries $p = |\Omega|/10^4$, and the signal-to-noise ratio SNR. We measure the normalized MSE, i.e. $\|\Pi_\Omega(M - A)\|_F^2 / \|\Pi_\Omega(M)\|_F^2$. The results can be seen in Figure 8-1, where it is illustrated that Fast Local Search sometimes returns significantly more accurate and lower-rank solutions than Fast Greedy, and Fast Greedy generally returns significantly more accurate and lower-rank solutions than SoftImpute.

### 8.3.2 Robust principal component analysis (RPCA)

The robust PCA paradigm asks one to decompose a given matrix $M$ as $L + S$, where $L$ is a low-rank matrix and $S$ is a sparse matrix. This is useful for applications with outliers where directly computing the principal components of $M$ is significantly affected by them. For a comprehensive survey on Robust PCA survey one can look at [24]. The following optimization problem encodes the above-stated requirements:

$$\min_{\text{rank}(L) \leq r^*} \|M - L\|_0 \tag{8.4}$$




(a) $r^* = 5$, $p = 0.2$, $SNR = 10$ (b) $r^* = 6$, $p = 0.5$, $SNR = 1$

Figure 8-1: Objective value error vs rank in the problem of Section 8.3.1.

where $\|X\|_0$ is the sparsity (i.e. number of non-zeros) of $X$. As neither the rank constraint or the $\ell_0$ function are convex, [34] replaced them by their usual convex relaxations, i.e. the nuclear norm $\|\cdot\|_*$ and $\ell_1$ norm respectively. However, we opt to only relax the $\ell_0$ function but not the rank constraint, leaving us with the problem:

$$\min_{\text{rank}(L)\leq r^*} \|M - L\|_1 \tag{8.5}$$

In order to make the objective differentiable and thus more well-behaved, we further replace the $\ell_1$ norm by the Huber loss $H_\delta(x) = \begin{cases} x^2/2 & \text{if } |x| \leq \delta \\ \delta|x| - \delta^2/2 & \text{otherwise} \end{cases}$, thus getting: $\min_{\text{rank}(L)\leq r^*} \sum_{ij} H_\delta(M-L)_{ij}$. This is a problem on which we can directly apply our algorithms. We solve the inner optimization problem by applying 10 L-BFGS iterations.

In Figure 8-2 one can see an example of foreground-background separation from video using robust PCA. The video is from the BMC 2012 dataset [164]. In this problem, the low-rank part corresponds to the background and the sparse part to the foreground. We compare three algorithms: Our Fast Greedy algorithm, standard PCA with 1 component (the choice of 1 was picked to get the best outcome), and the standard Principal Component Pursuit (PCP) algorithm ([34]), as implemented in [111], where we tuned the regularization parameter $\lambda$ to achieve the best result. We find that Fast Greedy has the best performance out of the three algorithms in this sample task.

## 8.4 Machine Learning Applications

### 8.4.1 Regularization techniques

In the previous section we showed that our proposed algorithms bring down different optimization objectives aggressively. However, in applications where the goal is to obtain a low generalization error, regularization is needed. We considered two different kinds of regularization. The first method is to run the inner optimization algorithm for less iterations, usually 2-3. Usually this is straightforward since an iterative method is used. For example, in the case $R(A) = \frac{1}{2}\|\Pi_\Omega(M-A)\|_F^2$ the inner optimization is a least squares linear regression problem that we solve using the LSQR algorithm. The second one is to add an $\ell_2$ regularizer to the objective function. However, this option did not

Figure 8-2: Foreground-background separation from video. From left to right: Fast Greedy with rank=3 and Huber loss with $\delta = 20$. Standard PCA with rank=1. Principal Component Pursuit (PCP) with $\lambda = 0.002$. Both PCA and PCP have visible "shadows" in the foreground that appear as "smudges" in the background. These are less obvious in a still frame but more apparent in a video.

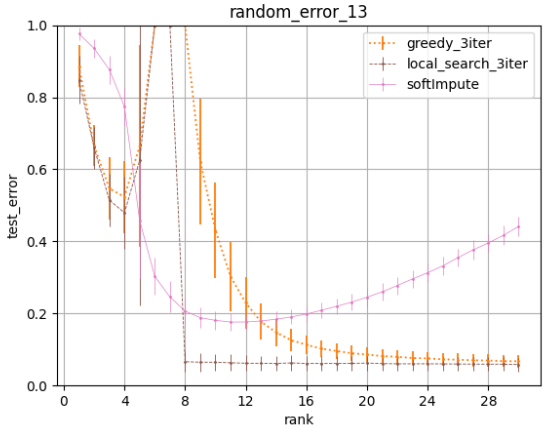
provide a substantial performance boost in our experiments, and so we have not implemented it.

### 8.4.2 Matrix Completion with Random Noise

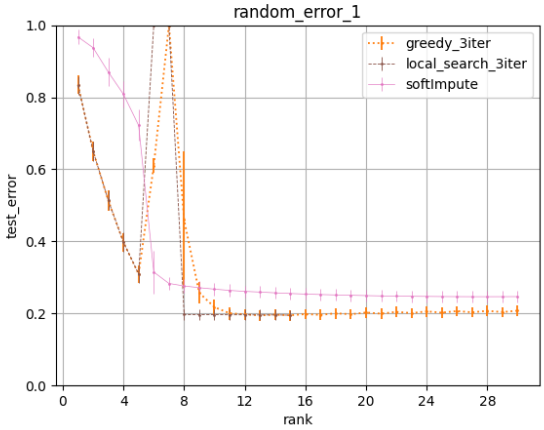
In this section we evaluate our algorithms on the task of recovering a low rank matrix $UV^\top$ after observing $\Pi_\Omega(UV^\top + \eta)$, i.e. a fraction of its entries with added noise. As in Section 8.3.1, we use the setting of [121] and compare with the SoftImpute method. The evaluation metric is the normalized MSE, defined as $(\sum_{(i,j)\notin\Omega}(UV^\top - A)_{ij}^2)/(\sum_{(i,j)\notin\Omega}(UV^\top)_{ij}^2)$, where $A$ is the predicted matrix and $UV^\top$ the true low rank matrix. A few example plots can be seen in Figure 8-3 and a table of results in Table 8.1. We have implemented the Fast Greedy and Fast Local Search algorithms with 3 inner optimization iterations. In the first few iterations there is a spike in the relative MSE of the algorithms that use the OPTIMIZE_FAST routine. We attribute this to the aggressive alternating minimization steps of this procedure and conjecture that adding a regularization term to the objective might smoothen the spike. However, the Fast Local Search algorithm still gives the best overall performance in terms of how well it approximates the true low rank matrix $UV^\top$, and in particular with a very small rank—practically the same as the true underlying rank.

### 8.4.3 Recommender Systems

In this section we compare our algorithms on the task of movie recommendation on the Movielens datasets [80]. In order to evaluate the algorithms, we perform random 80%-20% train-test splits that are the same for all algorithms and measure the mean RMSE in the test set. If we let $\Omega \subseteq [m] \times [n]$ be the set of user-movie pairs in the training set, we assume that the true user-movie matrix is low rank, and thus pose (8.1) with $R(A) = \frac{1}{2}\|\Pi_\Omega(M - A)\|_F^2$. We make the following slight modification




(a) $k = 5$, $p = 0.2$, $SNR = 10$




(b) $k = 6$, $p = 0.5$, $SNR = 1$

Figure 8-3: Test error vs rank in the matrix completion problem of Section 8.4.2. Bands of $\pm 1$ standard error are shown. Note that SoftImpute starts to overfit for ranks larger than 12 in (a). The "jumps" at around rank 5-7 happen because of overshooting (taking too large a step) during the insertion the rank-1 component in both Fast Greedy and Fast Local Search. More specifically, these implementations only apply 3 iterations of the inner optimization step, which in some cases are too few to amend the overshooting. However, after a few more iterations of the algorithm the overshooting issue is resolved (i.e. the algorithm has had enough iterations to scale down the rank-1 component that caused the overshooting).

| Algorithm | random_error_13 | random_error_1 | random_error_2 |
|---|---|---|---|
| SoftImpute ([121]) | 0.1759/10 | 0.2465/28 | 0.2284/30 |
| Fast Greedy (Algorithm 8.2.3) | 0.0673/30 | 0.1948/13 | 0.1826/21 |
| Fast Local Search (Algorithm 21) | 0.0613/14 | 0.1952/15 | 0.1811/15 |

Table 8.1: Lowest test error for any rank in the matrix completion problem of Section 8.4.2, and associated rank returned by each algorithm. In the form error/rank.

in order to take into account the range of the ratings $[1, 5]$: We clip the entries of $A$ between 1 and 5 when computing $\nabla R(A)$ in Algorithm 8.2.3 and Algorithm 21. In other words, instead of $\Pi_\Omega(A - M)$ we compute the gradient as $\Pi_\Omega(\text{clip}(A, 1, 5) - M)$. This is similar to replacing our objective by a Huber loss, with the difference that we only do so in the steps that we mentioned and not the inner optimization step, mainly for runtime efficiency reasons.

The results can be seen in Table 8.2. We do not compare with Fast Local Search, as we found that it only provides an advantage for small ranks ($< 30$), and otherwise matches Fast Greedy. For the inner optimization steps we have used the LSQR algorithm with 2 iterations in the 100K and 1M datasets, and with 3 iterations in the 10M dataset. Note that even though the SVD algorithm by [101] as implemented by [83] (with no user/movie bias terms) is a highly tuned algorithm for recommender systems that was one of the top solutions in the famous Netflix prize, it has comparable performance to our general-purpose Algorithm 8.2.3.

Finally, Table 8.3 demonstrates the speedup achieved by our algorithms over the basic greedy implementation. It should be noted that the speedup compared to the basic greedy of [144] (Algorithm 17) is larger as rank increases, since the fast algorithms scale linearly with rank, but the

| Algorithm | MovieLens 100K | MovieLens 1M | MovieLens 10M |
|---|---|---|---|
| NMF ([104]) | 0.9659 | 0.9166 | 0.8960 |
| SoftImpute | 1.0106 | 0.9599 | 0.957 |
| Alternating Minimization | **0.9355** | 0.8732 | 0.8410 |
| SVD ([101]) | 0.9533 | 0.8743 | **0.8315** |
| Fast Greedy (Algorithm 8.2.3) | 0.9451 | **0.8714** | 0.8330 |

Table 8.2: Mean RMSE and standard error among 5 random splits for 100K and 1M with standard errors $< 0.01$, and 3 random splits for 10M with standard errors $< 0.001$. The rank of the prediction is set to 100 except for NMF where it is 15 and Fast Greedy in the 10M dataset where it is chosen to be 35 by cross-validation. Alternating Minimization is a well known algorithm (e.g. [157]) that alternatively minimizes the left and right subspace, and also uses Frobenius norm regularization. For SoftImpute and Alternating Minimization we have found the best choice of parameters by performing a grid search over the rank and the multiplier of the regularization term. We have found the best choice of parameters by performing a grid search over the rank and the multiplier of the regularization term. We ran 20 iterations of Alternating Minimization in each case.

basic greedy scales quadratically.

| Algorithm | Figure 8-3 (a) | Movielens 100K | Movielens 1M |
|---|---|---|---|
| SoftImpute | 10.6 | 9.4 | 40.6 |
| Alternating Minimization | 18.9 | 252.0 | 1141.4 |
| Greedy ([144]) | 18.8 | 418.4 | 4087.3 |
| Fast Greedy | 10.2 | 43.4 | 244.2 |
| Fast Local Search | 10.8 | 46.1 | 263.0 |

Table 8.3: Runtimes (in seconds) of different algorithms for fitting a rank=30 solution in various experiments. Code written in python and tested on an Intel Skylake CPU with 16 vCPUs.

It is important to note that our goal here is not to be competitive with the best known algorithms for matrix completion, but rather to propose a general yet practically applicable method for rank-constrained convex optimization. For a recent survey on the best performing algorithms in the Movielens datasets see [139]. It should be noted that a lot of these algorithms have significant performance boost compared to our methods because they use additional features (meta information about each user, movie, timestamp of a rating, etc.) or stronger models (user/movie biases, "implicit" ratings). A runtime comparison with these recent approches is an interesting avenue for future work. As a rule of thumb, however, Fast Greedy has roughly the same runtime as SVD ([101]) in each iteration, i.e. $O(|\Omega|r)$, where $\Omega$ is the set of observable elements and $r$ is the rank. As some better performing approaches have been reported to be much slower than SVD (e.g. SVD++ is reported to be 50-100x slower than SVD in the Movielens 100K and 1M datasets ([83]), this might also suggest a runtime advantage of our approach compared to some better performing methods.

## 8.5 Conclusions

We presented simple algorithms with strong theoretical guarantees for optimizing a convex function under a rank constraint. Although the basic versions of these algorithms have appeared before, through a series of careful runtime, optimization, and generalization performance improvements

that we introduced, we have managed to reshape the performance of these algorithms in all fronts. Via our experimental validation on a host of practical problems such as low-rank matrix recovery with missing data, robust principal component analysis, and recommender systems, we have shown that the performance in terms of the solution quality matches or exceeds other widely used and even specialized solutions, thus making the argument that our Fast Greedy and Fast Local Search routines can be regarded as strong and practical general purpose tools for rank-constrained convex optimization. Interesting directions for further research include the exploration of different kinds of regularization and tuning for machine learning applications, as well as a competitive implementation and extensive runtime comparison of our algorithms.

# Bibliography


[1] Deeksha Adil, Rasmus Kyng, Richard Peng, and Sushant Sachdeva. "Iterative Refinement for $\ell_p$-norm Regression". In: *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM. 2019, pp. 1405–1424.

[2] Deeksha Adil and Sushant Sachdeva. "Faster $p$-norm minimizing flows, via smoothed $q$-norm problems". In: *Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM. 2020, pp. 892–910.

[3] Pankaj K. Agarwal, Micha Sharir, and Sivan Toledo. "Applications of Parametric Searching in Geometric Optimization". In: *J. Algorithms* 17.3 (1994), pp. 292–318. DOI: 10.1006/jagm.1994.1038. URL: https://doi.org/10.1006/jagm.1994.1038.

[4] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. *Network Flows*. Tech. rep. Alfred P. Sloan School of Management, Cambridge, MA, 1988.

[5] Zeyuan Allen-Zhu, Elad Hazan, Wei Hu, and Yuanzhi Li. "Linear convergence of a frank-wolfe type algorithm over trace-norm balls". In: *Advances in Neural Information Processing Systems* 30 (2017).

[6] Jason Altschuler, Aditya Bhaskara, Gang Fu, Vahab Mirrokni, Afshin Rostamizadeh, and Morteza Zadimoghaddam. "Greedy Column Subset Selection: New Bounds and Distributed Algorithms". In: *International Conference on Machine Learning*. 2016, pp. 2539–2548.

[7] Joel Andersson and Jan-Olov Strömberg. "On the theorem of uniform recovery of random sampling matrices". In: *IEEE Transactions on Information Theory* 60.3 (2014), pp. 1700–1710.

[8] Chetan Arora, Subhashis Banerjee, Prem Kalra, and SN Maheshwari. "Generic cuts: An efficient algorithm for optimal inference in higher order MRF-MAP". In: *European Conference on Computer Vision*. Springer. 2012, pp. 17–30.

[9] Kyriakos Axiotis, Adam Karczmarz, Anish Mukherjee, Piotr Sankowski, and Adrian Vladu. "Decomposable submodular function minimization via maximum flow". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 446–456.

[10] Kyriakos Axiotis, Aleksander Mądry, and Adrian Vladu. "Circulation control for faster minimum cost flow in unit-capacity graphs". In: *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2020, pp. 93–104.

[11] Kyriakos Axiotis, Aleksander Mądry, and Adrian Vladu. "Faster sparse minimum cost flow by electrical flow localization". In: *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2022, pp. 528–539.

[12] Kyriakos Axiotis and Maxim Sviridenko. "Iterative Hard Thresholding with Adaptive Regularization: Sparser Solutions Without Sacrificing Runtime". In: *International Conference on Machine Learning* (2022).

[13] Kyriakos Axiotis and Maxim Sviridenko. "Local Search Algorithms for Rank-Constrained Convex Optimization". In: *International Conference on Learning Representations*. 2021.

[14] Kyriakos Axiotis and Maxim Sviridenko. "Sparse Convex Optimization via Adaptively Regularized Hard Thresholding". In: *Journal of Machine Learning Research* 22 (2021), pp. 1–47.

[15] Francis Bach et al. "Learning with submodular functions: A convex optimization perspective". In: *Foundations and Trends® in Machine Learning* 6.2-3 (2013), pp. 145–373.

[16] Sohail Bahmani, Bhiksha Raj, and Petros T Boufounos. "Greedy sparsity-constrained optimization". In: *Journal of Machine Learning Research* 14.Mar (2013), pp. 807–841.

[17] Marshall Bern, John R Gilbert, Bruce Hendrickson, Nhat Nguyen, and Sivan Toledo. "Support-graph preconditioners". In: *SIAM Journal on Matrix Analysis and Applications* 27.4 (2006), pp. 930–951.

[18] Aaron Bernstein, Jan van den Brand, Maximilian Probst Gutenberg, Danupon Nanongkai, Thatchaphol Saranurak, Aaron Sidford, and He Sun. "Fully-dynamic graph sparsifiers against an adaptive adversary". In: *arXiv preprint arXiv:2004.08432* (2020).

[19] Thomas Blumensath and Mike E Davies. "Iterative hard thresholding for compressed sensing". In: *Applied and computational harmonic analysis* 27.3 (2009), pp. 265–274.

[20] Thomas Blumensath and Mike E Davies. *On the difference between orthogonal matching pursuit and orthogonal least squares*. Tech. rep. 2007.

[21] Holger Boche, Robert Calderbank, Gitta Kutyniok, and Jan Vybiral. "A survey of compressed sensing". In: *Compressed sensing and its applications*. Springer, 2015, pp. 1–39.

[22] Glencora Borradaile and Philip N. Klein. "An $O(n \log n)$ algorithm for maximum $st$-flow in a directed planar graph". In: *J. ACM* 56.2 (2009), 9:1–9:30. DOI: 10.1145/1502793.1502798. URL: https://doi.org/10.1145/1502793.1502798.

[23] Jonathan Borwein and Adrian S Lewis. *Convex analysis and nonlinear optimization: theory and examples*. Springer Science & Business Media, 2010.

[24] Thierry Bouwmans, Sajid Javed, Hongyang Zhang, Zhouchen Lin, and Ricardo Otazo. "On the applications of robust PCA in image and video processing". In: *Proceedings of the IEEE* 106.8 (2018), pp. 1427–1457.

[25] Jan van den Brand, Yin Tat Lee, Yang P Liu, Thatchaphol Saranurak, Aaron Sidford, Zhao Song, and Di Wang. "Minimum cost flows, mdps, and $\ell_1$-regression in nearly linear time for dense instances". In: *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*. 2021, pp. 859–869.

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

[27] Sébastien Bubeck, Michael B Cohen, Yin Tat Lee, James R Lee, and Aleksander Mądry. "K-server via multiscale entropic regularization". In: *Proceedings of the 50th annual ACM SIGACT symposium on theory of computing*. 2018, pp. 3–16.

[28] Sébastien Bubeck, Michael B Cohen, Yin Tat Lee, and Yuanzhi Li. "An homotopy method for $\ell_p$ regression provably beyond self-concordance and in input-sparsity time". In: *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*. 2018, pp. 1130–1137.

[29] T Tony Cai, Lie Wang, and Guangwu Xu. "New bounds for restricted isometry constants". In: *IEEE Transactions on Information Theory* 56.9 (2010), pp. 4388–4394.

[30] T Tony Cai, Lie Wang, and Guangwu Xu. "Shifting inequality and recovery of sparse signals". In: *IEEE Transactions on Signal processing* 58.3 (2009), pp. 1300–1308.

[31] Emmanuel J Candes. "The restricted isometry property and its implications for compressed sensing". In: *Comptes rendus mathematique* 346.9-10 (2008), pp. 589–592.

[32] Emmanuel J Candes, Justin K Romberg, and Terence Tao. "Stable signal recovery from incomplete and inaccurate measurements". In: *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences* 59.8 (2006), pp. 1207–1223.

[33] Emmanuel J Candes and Terence Tao. "Decoding by linear programming". In: *IEEE transactions on information theory* 51.12 (2005), pp. 4203–4215.

[34] Emmanuel J Candès, Xiaodong Li, Yi Ma, and John Wright. "Robust principal component analysis?" In: *Journal of the ACM (JACM)* 58.3 (2011), pp. 1–37.

[35] Deeparnab Chakrabarty, Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. "Subquadratic submodular function minimization". In: *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. 2017, pp. 1220–1231.

[36] Antonin Chambolle and Jérôme Darbon. "On total variation minimization and surface evolution using parametric maximum flows". In: *International journal of computer vision* 84.3 (2009), p. 288.

[37] Li Chen, Rasmus Kyng, Yang P Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. "Maximum flow and minimum-cost flow in almost-linear time". In: *arXiv preprint arXiv:2203.00671* (2022).

[38] Lin Chen, Moran Feldman, and Amin Karbasi. "Weakly Submodular Maximization Beyond Cardinality Constraints: Does Randomization Help Greedy?" In: *International Conference on Machine Learning*. 2018, pp. 804–813.

[39] Paul Christiano, Jonathan A Kelner, Aleksander Mądry, 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*. 2011, pp. 273–282.

[40] Fan Chung and Linyuan Lu. "Concentration inequalities and martingale inequalities: a survey". In: *Internet Mathematics* 3.1 (2006), pp. 79–127.

[41] Michael B Cohen, Rasmus Kyng, Gary L Miller, Jakub W Pachocki, Richard Peng, Anup B Rao, and Shen Chen Xu. "Solving SDD linear systems in nearly $m \log^{1/2} n$ time". In: *Proceedings of the 46th Annual ACM Symposium on Theory of computing*. 2014, pp. 343–352.

[42] Michael B Cohen, Aleksander Mądry, Piotr Sankowski, and Adrian Vladu. "Negative-Weight Shortest Paths and Unit Capacity Minimum Cost Flow in $\widetilde{O}(m^{10/7} \log W)$ Time". In: *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM. 2017, pp. 752–771.

[43] Daniel Dadush, László A Végh, and Giacomo Zambelli. "Geometric rescaling algorithms for submodular function minimization". In: *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM. 2018, pp. 832–848.

[44] 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*. 2008, pp. 451–460.

[45] Nikhil R Devanur, Shaddin Dughmi, Roy Schwartz, Ankit Sharma, and Mohit Singh. "On the approximation of submodular functions". In: *arXiv preprint arXiv:1304.4948* (2013).

[46] David L Donoho. "Compressed sensing". In: *IEEE Transactions on information theory* 52.4 (2006), pp. 1289–1306.

[47] David L Donoho and Michael Elad. "Optimally sparse representation in general (nonorthogonal) dictionaries via l1 minimization". In: *Proceedings of the National Academy of Sciences* 100.5 (2003), pp. 2197–2202.

[48] David L Donoho, Arian Maleki, and Andrea Montanari. "Message-passing algorithms for compressed sensing". In: *Proceedings of the National Academy of Sciences* 106.45 (2009), pp. 18914–18919.

[49] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2017. URL: `http://archive.ics.uci.edu/ml`.

[50] John Duchi, Elad Hazan, and Yoram Singer. "Adaptive subgradient methods for online learning and stochastic optimization." In: *Journal of machine learning research* 12.7 (2011).

[51] David Durfee, Yu Gao, Gramoz Goranci, and Richard Peng. "Fully dynamic spectral vertex sparsifiers and applications". In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. 2019, pp. 914–925.

[52] David Durfee, Kevin A Lai, and Saurabh Sawlani. "$\ell_1$ Regression using Lewis Weights Preconditioning and Stochastic Gradient Descent". In: *Conference on Learning Theory*. 2018, pp. 1626–1656.

[53] Jack Edmonds. "Matroids and the greedy algorithm". In: *Mathematical programming* 1.1 (1971), pp. 127–136.

[54] Jack Edmonds. "Submodular functions, matroids, and certain polyhedra". In: *Combinatorial Structures and Their Applications*. 1970, pp. 69–87.

[55] Ethan Elenberg, Alexandros G Dimakis, Moran Feldman, and Amin Karbasi. "Streaming weak submodularity: Interpreting neural networks on the fly". In: *Advances in Neural Information Processing Systems*. 2017, pp. 4044–4054.

[56] Alina Ene and Huy Nguyen. "Random coordinate descent methods for minimizing decomposable submodular functions". In: *International Conference on Machine Learning*. PMLR. 2015, pp. 787–795.

[57] Alina Ene, Huy L Nguyen, and László A Végh. "Decomposable Submodular Function Minimization: Discrete and Continuous". In: *NIPS*. 2017.

[58] Jeff Erickson. "Maximum Flows and Parametric Shortest Paths in Planar Graphs". In: *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*. Ed. by Moses Charikar. SIAM, 2010, pp. 794–804. DOI: `10.1137/1.9781611973075.65`.

[59] Steve Fisk. "A note on weyl's inequality". In: (1996).

[60] Alexander Fix, Thorsten Joachims, Sung Min Park, and Ramin Zabih. "Structured learning of sum-of-submodular higher order energy functions". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2013, pp. 3104–3111.

[61] Lisa Fleischer and Satoru Iwata. "A push-relabel framework for submodular function minimization and applications to parametric optimization". In: *Discrete Applied Mathematics* 131.2 (2003), pp. 311–322.

[62] L. R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, 1962. ISBN: 9780691625393.

[63] Dean Foster, Howard Karloff, and Justin Thaler. "Variable selection is hard". In: *Conference on Learning Theory*. 2015, pp. 696–709.

[64] Simon Foucart. "A note on guaranteed sparse recovery via $\ell_1$-minimization". In: *Applied and Computational Harmonic Analysis* 29.1 (2010), pp. 97–103.

[65] Simon Foucart. "Hard thresholding pursuit: an algorithm for compressive sensing". In: *SIAM Journal on Numerical Analysis* 49.6 (2011), pp. 2543–2563.

[66] Simon Foucart. "Sparse recovery algorithms: sufficient conditions in terms of restricted isometry constants". In: *Approximation Theory XIII: San Antonio 2010*. Springer, 2012, pp. 65–77.

[67] Simon Foucart and Ming-Jun Lai. "Sparsest solutions of underdetermined linear systems via $\ell_q$-minimization for $0 < q \leq 1$". In: *Applied and Computational Harmonic Analysis* 26.3 (2009), pp. 395–407.

[68] Simon Foucart and Holger Rauhut. "A mathematical introduction to compressive sensing". In: *Bull. Am. Math* 54 (2017), pp. 151–165.

[69] Marguerite Frank, Philip Wolfe, et al. "An algorithm for quadratic programming". In: *Naval research logistics quarterly* 3.1-2 (1956), pp. 95–110.

[70] Satoru Fujishige. "Lexicographically optimal base of a polymatroid with respect to a weight vector". In: *Mathematics of Operations Research* 5.2 (1980), pp. 186–196.

[71] Harold N Gabow. "Scaling algorithms for network problems". In: *24th Annual Symposium on Foundations of Computer Science (SFCS 1983)*. IEEE. 1983, pp. 248–258.

[72] Giorgio Gallo, Michael D. Grigoriadis, and Robert Endre Tarjan. "A Fast Parametric Maximum Flow Algorithm and Applications". In: *SIAM J. Comput.* 18.1 (1989), pp. 30–55. DOI: 10.1137/0218003. URL: https://doi.org/10.1137/0218003.

[73] Yu Gao, Yang P Liu, and Richard Peng. "Fully dynamic electrical flows: Sparse maxflow faster than goldberg-rao". In: *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2022, pp. 516–527.

[74] Andrew V Goldberg, Sagi Hed, Haim Kaplan, and Robert E Tarjan. "Minimum-cost flows in unit-capacity networks". In: *Theory of Computing Systems* 61.4 (2017), pp. 987–1010.

[75] Andrew V Goldberg and Satish Rao. "Beyond the flow decomposition barrier". In: *Journal of the ACM (JACM)* 45.5 (1998), pp. 783–797.

[76] Andrew V Goldberg and Robert E Tarjan. "A new approach to the maximum-flow problem". In: *Journal of the ACM (JACM)* 35.4 (1988), pp. 921–940.

[77] Frieda Granot, S. Thomas McCormick, Maurice Queyranne, and Fabio Tardella. "Structural and algorithmic properties for parametric minimum cuts". In: *Math. Program.* 135.1-2 (2012), pp. 337–367. DOI: 10.1007/s10107-011-0463-1.

[78] Martin Grötschel, László Lovász, and Alexander Schrijver. "The ellipsoid method and its consequences in combinatorial optimization". In: *Combinatorica* 1.2 (1981), pp. 169–197.

[79] Dan Gusfield and Charles U. Martel. "A Fast Algorithm for the Generalized Parametric Minimum Cut Problem and Applications". In: *Algorithmica* 7.5&6 (1992), pp. 499–519. DOI: 10.1007/BF01758775.

[80] F Maxwell Harper and Joseph A Konstan. "The movielens datasets: History and context". In: *Acm transactions on interactive intelligent systems (tiis)* 5.4 (2015), pp. 1–19.

[81] Nicholas J Higham. *Accuracy and stability of numerical algorithms*. Vol. 80. SIAM, 2002.

[82] R. A. Horn and C.R. Johnson. *Topics in Matrix Analysis*. 1991.

[83] Nicolas Hug. "Surprise: A Python library for recommender systems". In: *Journal of Open Source Software* 5.52 (2020), p. 2174. DOI: 10.21105/joss.02174. URL: https://doi.org/10.21105/joss.02174.

[84] Satoru Iwata. "A faster scaling algorithm for minimizing submodular functions". In: *SIAM Journal on Computing* 32.4 (2003), pp. 833–840.

[85] Satoru Iwata, Lisa Fleischer, and Satoru Fujishige. "A combinatorial strongly polynomial algorithm for minimizing submodular functions". In: *Journal of the ACM (JACM)* 48.4 (2001), pp. 761–777.

[86] Martin Jaggi. "Revisiting Frank-Wolfe: Projection-free sparse convex optimization". In: *International Conference on Machine Learning*. PMLR. 2013, pp. 427–435.

[87] Prateek Jain, Ambuj Tewari, and Inderjit S Dhillon. "Orthogonal matching pursuit with replacement". In: *Advances in neural information processing systems*. 2011, pp. 1215–1223.

[88] Prateek Jain, Ambuj Tewari, and Inderjit S Dhillon. "Partial hard thresholding". In: *IEEE Transactions on Information Theory* 63.5 (2017), pp. 3029–3038.

[89] Prateek Jain, Ambuj Tewari, and Purushottam Kar. "On iterative hard thresholding methods for high-dimensional m-estimation". In: *Advances in Neural Information Processing Systems*. 2014, pp. 685–693.

[90] Stefanie Jegelka, Francis Bach, and Suvrit Sra. "Reflection methods for user-friendly submodular optimization". In: *Advances in Neural Information Processing Systems* 26 (2013).

[91] Haotian Jiang, Yin Tat Lee, Zhao Song, and Sam Chiu-wai Wong. "An improved cutting plane method for convex optimization, convex-concave games, and its applications". In: *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*. 2020, pp. 944–953.

[92] Sham M Kakade, Shai Shalev-Shwartz, and Ambuj Tewari. "Regularization techniques for learning with matrices". In: *The Journal of Machine Learning Research* 13.1 (2012), pp. 1865–1890.

[93] Adam Karczmarz and Piotr Sankowski. "A Deterministic Parallel APSP Algorithm and its Applications". In: *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 255–272. DOI: 10.1137/1.9781611976465.17. URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611976465.17.

[94] Senanayak Sesh Kumar Karri, Francis Bach, and Thomas Pock. "Fast decomposable submodular function minimization using constrained total variation". In: *Advances in Neural Information Processing Systems* 32 (2019).

[95] Tarun Kathuria, Yang P Liu, and Aaron Sidford. "Unit Capacity Maxflow in Almost $O(m^{4/3})$ Time". In: *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2020, pp. 119–130.

[96] Jonathan Kelner and Petar Maymounkov. "Electric routing and concurrent flow cutting". In: *International Symposium on Algorithms and Computation*. Springer. 2009, pp. 792–801.

[97] 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 25th Annual ACM-SIAM Symposium on Discrete algorithms*. SIAM. 2014, pp. 217–226.

[98] Jonathan A Kelner, Lorenzo Orecchia, Aaron Sidford, and Zeyuan Allen Zhu. "A simple, combinatorial algorithm for solving SDD systems in nearly-linear time". In: *Proceedings of the 45th Annual ACM Symposium on Theory of Computing*. 2013, pp. 911–920.

[99] Pushmeet Kohli, Philip HS Torr, et al. "Robust higher order potentials for enforcing label consistency". In: *International Journal of Computer Vision* 82.3 (2009), pp. 302–324.

[100] Vladimir Kolmogorov. "Minimizing a sum of submodular functions". In: *Discrete Applied Mathematics* 160.15 (2012), pp. 2246–2258.

[101] Yehuda Koren, Robert Bell, and Chris Volinsky. "Matrix factorization techniques for recommender systems". In: *Computer* 42.8 (2009), pp. 30–37.

[102] Ioannis Koutis, Gary L Miller, and David Tolliver. "Combinatorial preconditioners and multilevel solvers for problems in computer vision and image processing". In: *International Symposium on Visual Computing*. Springer. 2009, pp. 1067–1078.

[103] Rasmus Kyng, Richard Peng, Sushant Sachdeva, and Di Wang. "Flows in almost linear time via adaptive preconditioning". In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. 2019, pp. 902–913.

[104] Daniel D Lee and H Sebastian Seung. "Algorithms for non-negative matrix factorization". In: *Advances in neural information processing systems*. 2001, pp. 556–562.

[105] Yin Tat Lee, Satish Rao, and Nikhil Srivastava. "A new approach to computing maximum flows using electrical flows". In: *Proceedings of the 45th Annual ACM Symposium on Theory of Computing*. 2013, pp. 755–764.

[106] Yin Tat Lee and Aaron Sidford. "Path finding methods for linear programming: Solving linear programs in $O(\sqrt{rank})$ iterations and faster algorithms for maximum flow". In: *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*. IEEE. 2014, pp. 424–433.

[107] Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. "A faster cutting plane method and its implications for combinatorial and convex optimization". In: *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*. IEEE. 2015, pp. 1049–1065.

[108] David D Lewis, Yiming Yang, Tony Russell-Rose, and Fan Li. "Rcv1: A new benchmark collection for text categorization research". In: *Journal of machine learning research* 5.Apr (2004), pp. 361–397.

[109] Edo Liberty and Maxim Sviridenko. "Greedy minimization of weakly supermodular set functions". In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik. 2017.

[110] Hui Lin and Jeff Bilmes. "Optimal selection of limited vocabulary speech corpora". In: *Twelfth Annual Conference of the International Speech Communication Association*. 2011.

[111] Zhouchen Lin, Minming Chen, and Yi Ma. "The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices". In: *arXiv preprint arXiv:1009.5055* (2010).

[112] Haoyang Liu and Rina Foygel Barber. "Between hard and soft thresholding: optimal iterative thresholding algorithms". In: *Information and Inference: A Journal of the IMA* 9.4 (2020), pp. 899–933.

[113] Ji Liu, Jieping Ye, and Ryohei Fujimaki. "Forward-backward greedy algorithms for general convex smooth functions over a cardinality constraint". In: *International Conference on Machine Learning*. 2014, pp. 503–511.

[114] Yang P Liu and Aaron Sidford. "Faster Divergence Maximization for Faster Maximum Flow". In: *arXiv:2003.08929* (2020).

[115] Yang P Liu and Aaron Sidford. "Faster Energy Maximization for Faster Maximum Flow". In: *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*. 2020.

[116] László Lovász. "Submodular functions and convexity". In: *Mathematical programming the state of the art*. Springer, 1983, pp. 235–257.

[117] Aleksander Mądry. "Computing maximum flow with augmenting electrical flows". In: *57th Annual Symposium on Foundations of Computer Science*. IEEE. 2016, pp. 593–602.

[118] Aleksander Mądry. "Navigating central path with electrical flows: From flows to matchings, and back". In: *54th Annual Symposium on Foundations of Computer Science*. IEEE. 2013, pp. 253–262.

[119] K. Mahler. "An inequality for the discriminant of a polynomial." In: *Michigan Math. J.* 11.3 (Sept. 1964), pp. 257–262. DOI: 10.1307/mmj/1028999140. URL: https://doi.org/10.1307/mmj/1028999140.

[120] Per-Gunnar Martinsson, Vladimir Rokhlin, and Mark Tygert. "A randomized algorithm for the decomposition of matrices". In: *Applied and Computational Harmonic Analysis* 30.1 (2011), pp. 47–68.

[121] Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. "Spectral regularization algorithms for learning large incomplete matrices". In: *The Journal of Machine Learning Research* 11 (2010), pp. 2287–2322.

[122] S. Thomas McCormick. "Fast Algorithms for Parametric Scheduling Come From Extensions to Parametric Maximum Flow". In: *Oper. Res.* 47.5 (1999), pp. 744–756. DOI: 10.1287/opre.47.5.744.

[123] Nimrod Megiddo. "Applying Parallel Computation Algorithms in the Design of Serial Algorithms". In: *J. ACM* 30.4 (1983), pp. 852–865. DOI: 10.1145/2157.322410. URL: https://doi.org/10.1145/2157.322410.

[124] Qun Mo and Song Li. "New bounds on the restricted isometry constant $\delta_{2k}$". In: *Applied and Computational Harmonic Analysis* 31.3 (2011), pp. 460–468.

[125] Ahmad Mousavi, Mehdi Rezaee, and Ramin Ayanzadeh. "A survey on compressive sensing: classical results and recent advancements". In: *Journal of Mathematical Modeling* 8.3 (2020), pp. 309–344.

[126] Mukund Narasimhan and Jeff A Bilmes. "Local Search for Balanced Submodular Clusterings." In: *IJCAI*. 2007, pp. 981–986.

[127] Balas Kausik Natarajan. "Sparse approximate solutions to linear systems". In: *SIAM journal on computing* 24.2 (1995), pp. 227–234.

[128] Deanna Needell and Joel A Tropp. "CoSaMP: Iterative signal recovery from incomplete and inaccurate samples". In: *Applied and computational harmonic analysis* 26.3 (2009), pp. 301–321.

[129] Deanna Needell and Roman Vershynin. "Signal recovery from incomplete and inaccurate measurements via regularized orthogonal matching pursuit". In: *IEEE Journal of selected topics in signal processing* 4.2 (2010), pp. 310–316.

[130] Deanna Needell and Roman Vershynin. "Uniform uncertainty principle and signal recovery via regularized orthogonal matching pursuit". In: *Foundations of computational mathematics* 9.3 (2009), pp. 317–334.

[131] Robert Nishihara, Stefanie Jegelka, and Michael I Jordan. "On the Convergence Rate of Decomposable Submodular Function Minimization". In: *Advances in Neural Information Processing Systems* 27 (2014), pp. 640–648.

[132] James B Orlin. "A faster strongly polynomial time algorithm for submodular function minimization". In: *Mathematical Programming* 118.2 (2009), pp. 237–251.

[133] James B Orlin. "A polynomial time primal network simplex algorithm for minimum cost flows". In: *Mathematical Programming* 78.2 (1997), pp. 109–129.

[134] Christopher C Paige and Michael A Saunders. "LSQR: An algorithm for sparse linear equations and sparse least squares". In: *ACM Transactions on Mathematical Software (TOMS)* 8.1 (1982), pp. 43–71.

[135] Yagyensh Chandra Pati, Ramin Rezaiifar, and Perinkulam Sambamurthy Krishnaprasad. "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition". In: *Proceedings of 27th Asilomar conference on signals, systems and computers*. IEEE. 1993, pp. 40–44.

[136] Richard Peng. "Approximate undirected maximum flows in $O(m$ polylog $n)$ time". In: *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete algorithms*. SIAM. 2016, pp. 1862–1867.

[137] 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*. 2014, pp. 333–342.

[138] Alexandra Peste, Eugenia Iofinova, Adrian Vladu, and Dan Alistarh. "AC/DC: Alternating Compressed/DeCompressed Training of Deep Neural Networks". In: *Thirty-Fifth Conference on Neural Information Processing Systems*. 2021.

[139] Steffen Rendle, Li Zhang, and Yehuda Koren. "On the difficulty of evaluating baselines: A study on recommender systems". In: *arXiv preprint arXiv:1905.01395* (2019).

[140] R Tyrrell Rockafellar. *Convex analysis*. Vol. 36. Princeton University Press, 1970.

[141] Alex Rubinsteyn and Sergey Feldman. *fancyimpute: Version 0.0.16*. Version v0.0.16. May 2016. DOI: 10.5281/zenodo.51773. URL: https://doi.org/10.5281/zenodo.51773.

[142] Ludwig Schmidt. "Algorithms above the noise floor". PhD thesis. Massachusetts Institute of Technology, Cambridge, USA, 2018. URL: http://hdl.handle.net/1721.1/118098.

[143] Alexander Schrijver. "A combinatorial algorithm minimizing submodular functions in strongly polynomial time". In: *Journal of Combinatorial Theory, Series B* 80.2 (2000), pp. 346–355.

[144] Shai Shalev-Shwartz, Alon Gonen, and Ohad Shamir. "Large-scale convex minimization with a low-rank constraint". In: *Proceedings of the 28th International Conference on International Conference on Machine Learning*. 2011, pp. 329–336.

[145] Shai Shalev-Shwartz and Yoram Singer. "Convex repeated games and Fenchel duality". In: *NIPS*. Vol. 6. 2006, pp. 1265–1272.

[146] Shai Shalev-Shwartz, Nathan Srebro, and Tong Zhang. "Trading accuracy for sparsity in optimization problems with sparsity constraints". In: *SIAM Journal on Optimization* 20.6 (2010), pp. 2807–2832.

[147] Ishant Shanu, Chetan Arora, and Parag Singla. "Min norm point algorithm for higher order MRF-MAP inference". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 5365–5374.

[148] Jie Shen and Ping Li. "On the iteration complexity of support recovery via hard thresholding pursuit". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 3115–3124.

[149] Jie Shen and Ping Li. "Partial hard thresholding: Towards a principled analysis of support recovery". In: *Advances in Neural Information Processing Systems*. 2017, pp. 3124–3134.

[150] Jonah Sherman. "Area-convexity, $\ell_\infty$ regularization, and undirected multicommodity flow". In: *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. 2017, pp. 452–460.

[151] Jonah Sherman. "Generalized preconditioning and undirected minimum-cost flow". In: *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM. 2017, pp. 772–780.

[152] Jonah Sherman. "Nearly maximum flows in nearly linear time". In: *54th Annual Symposium on Foundations of Computer Science*. IEEE. 2013, pp. 263–269.

[153] Aaron Sidford and Kevin Tian. "Coordinate methods for accelerating $\ell_\infty$ regression and faster approximate maximum flow". In: *59th Annual Symposium on Foundations of Computer Science*. IEEE. 2018, pp. 922–933.

[154] Raghav Somani, Chirag Gupta, Prateek Jain, and Praneeth Netrapalli. "Support recovery for orthogonal matching pursuit: upper and lower bounds". In: *Advances in Neural Information Processing Systems*. 2018, pp. 10814–10824.

[155] Daniel A Spielman and Shang-Hua Teng. "Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems". In: *Proceedings of the 36th annual ACM symposium on Theory of computing*. 2004, pp. 81–90.

[156] Daniel A Spielman and Shang-Hua Teng. "Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time". In: *Journal of the ACM (JACM)* 51.3 (2004), pp. 385–463.

[157] Nathan Srebro, Jason Rennie, and Tommi Jaakkola. "Maximum-margin matrix factorization". In: *Advances in neural information processing systems* 17 (2004), pp. 1329–1336.

[158] Peter Stobbe and Andreas Krause. "Efficient minimization of decomposable submodular functions". In: *Advances in Neural Information Processing Systems* 23 (2010).

[159] Arthur Szlam, Yuval Kluger, and Mark Tygert. "An implementation of a randomized algorithm for principal component analysis". In: *arXiv preprint arXiv:1412.3510* (2014).

[160] Robert Endre Tarjan, Julie Ward, Bin Zhang, Yunhong Zhou, and Jia Mao. "Balancing Applied to Maximum Network Flow Problems". In: *Algorithms - ESA 2006, 14th Annual European Symposium, Zurich, Switzerland, September 11-13, 2006, Proceedings*. 2006, pp. 612–623. DOI: 10.1007/11841036_55. URL: https://doi.org/10.1007/11841036%5C_55.

[161] Robert Tibshirani. "Regression shrinkage and selection via the lasso". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (1996), pp. 267–288.

[162] Sivan Toledo and Haim Avron. "Combinatorial preconditioners". In: *Combinatorial Scientific Computing* (2010), pp. 69–93.

[163] Andrew Tulloch. *Fast Randomized SVD*. `https://research.fb.com/blog/2014/09/fast-randomized-svd/`. 2014.

[164] Antoine Vacavant, Thierry Chateau, Alexis Wilhelm, and Laurent Lequièvre. "A benchmark dataset for outdoor foreground/background extraction". In: *Asian Conference on Computer Vision*. Springer. 2012, pp. 291–300.

[165] Nate Veldt, Austin R Benson, and Jon Kleinberg. "Minimizing Localized Ratio Cut Objectives in Hypergraphs". In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2020, pp. 1708–1718.

[166] Sara Vicente, Vladimir Kolmogorov, and Carsten Rother. "Joint optimization of segmentation and appearance models". In: *2009 IEEE 12th International Conference on Computer Vision*. IEEE. 2009, pp. 755–762.

[167] James Hardy Wilkinson. *Rounding errors in algebraic processes*. Courier Corporation, 1994.

[168] Chee-Keng Yap et al. *Fundamental problems of algorithmic algebra*. Vol. 49. Oxford University Press Oxford, 2000.

[169] Xiaotong Yuan, Ping Li, and Tong Zhang. "Exact recovery of hard thresholding pursuit". In: *Advances in Neural Information Processing Systems*. 2016, pp. 3558–3566.

[170] Tong Zhang. "Sparse recovery with orthogonal matching pursuit under RIP". In: *IEEE Transactions on Information Theory* 57.9 (2011), pp. 6215–6221.

[171] Le Zheng, Arian Maleki, Haolei Weng, Xiaodong Wang, and Teng Long. "Does $\ell_p$-Minimization Outperform $\ell_1$-Minimization?" In: *IEEE Transactions on Information Theory* 63.11 (2017), pp. 6896–6935.

# Chapter 9

# Appendix

## 9.1 Appendix for Chapter 3

### 9.1.1 Initializing the Interior Point Method

We require finding an initial $\boldsymbol{f}_0$ for which the solution can be easily brought to centrality. To do so, we modify the original graph by adding a set of $O(m)$ arcs with high cost $c_\infty = (m+1)\|\boldsymbol{c}\|_\infty$ such that the flow which pushes exactly $1/2$ units on every arc in the modified graph routes the original demand $\boldsymbol{d}$.

The consequence of adding such these edges is that, while the solution to the original problem remains unchanged, since it will never be beneficial to use an arc with cost $c_\infty$ in the optimal solution. Meanwhile initializing $\boldsymbol{f}_0 = \mathbf{1}/2$ ensures that the flow on each arc is equally far from the upper and the lower barrier, and therefore their contributions to the gradient will cancel. This will make centering trivially easy.

First let us show that such an augmentation is indeed possible.

**Lemma 9.1.1.** *Let $G = (V, E, \boldsymbol{c})$ be a directed graph with $|E| = m$ arcs with unit capacity, $|V| = n$ vertices, costs on arcs $\boldsymbol{c} \geq 0$, and let $\boldsymbol{d} \in \mathbb{Z}^n$ be a demand vector $\sum_{i=1}^n d_i = 0$.*

*Then there exists a graph $G' = (V', E', \boldsymbol{c}')$ with at most $2m$ unit-capacity arcs, and a demand vector $\boldsymbol{d}'$ such that the flow $\boldsymbol{f}' = \mathbf{1}/2$ routes the demand $\boldsymbol{d}'$ in $G'$.*

*Furthermore if $(\boldsymbol{f}')^*$ is a flow in $G'$ with $\mathbf{0} \leq (\boldsymbol{f}')^* \leq \mathbf{1}$ and which routes $\boldsymbol{d}'$ such that the cost $\langle \boldsymbol{c}', \boldsymbol{f}' \rangle$ is minimized, then one can convert it in $O(m)$ time into a flow $\boldsymbol{f}^*$ which routes $\boldsymbol{d}$ in $G$, such that $\mathbf{0} \leq \boldsymbol{f}^* \leq \mathbf{1}$ and the cost $\langle \boldsymbol{c}, \boldsymbol{f} \rangle$ is minimized, or certify that no such flow exists.*

*Proof.* First we construct the graph $G'$. Let $V' = V \cup \{v_0\}$, where $v_0$ is a new vertex. Initialize $S = \emptyset$. For each vertex $v \in V$, let $\ell(v) = d_v - \frac{1}{2}(|E^-(v)| - |E^+(v)|)$, representing the excess flow at vertex $v$ after routing $\boldsymbol{f} = 1/2$ on each arc. Let $c_\infty = (m+1)\|\boldsymbol{c}\|_\infty$. For each $v$ where $\ell(v) > 0$, create $2\ell(v)$ arcs $(v, v_0)$ with cost $c_\infty$, and add them to $S$. Similarly, for each $v$ where $\ell(v) < 0$ create $-2\ell(v)$ arcs $(v_0, v)$ with cost $c_\infty$ and add them to $S$. Note that $2\ell(v)$ is an integer, since $\boldsymbol{d}$ is integral. Let $E' = E \cup S$ and $\boldsymbol{c}'$ be the corresponding cost vector where arcs in $E$ preserve their original cost $\boldsymbol{c}$, while those in $S$ have cost $c_\infty$. Let $d'_v = d_v$ for all $v \in V$ and $d'_{v_0} = 0$.

Now let $(\boldsymbol{f}')^*$ be the minimum cost flow in $G'$ which satisfies capacity constraints. If $(\boldsymbol{f}')^*$ is not supported on any arcs in $S$, then $(\boldsymbol{f}')^*$ is also a feasible flow in $G$. Furthermore it must be the optimal flow in $G$, since otherwise $(\boldsymbol{f}')^*$ would not have been optimal in $G'$. If $(\boldsymbol{f}')^*$ has nonzero flow on some arc in $S$, then we must conclude that it is impossible to route $\boldsymbol{d}$ in $G$ while satisfying capacity constraints.

Suppose it were possible to do so using a flow $\boldsymbol{f}^*$. Then, consider the circulation $\boldsymbol{g} = \boldsymbol{f}^* - (\boldsymbol{f}')^*$. Now convert $\boldsymbol{g}$ into a *minimal* circulation $\widetilde{\boldsymbol{g}}$ which preserves the flows on $S$ as follows: while $\boldsymbol{g}$ contains a cycle without any arcs in $S$, decrease the value of all flows along that cycle by the minimum value of the flow along it. This operation always zeroes out the flow on at least one edge, so the process must finish. Furthermore note that the cost of any such cycle must be non-positive, since otherwise we contradict the optimality of $\boldsymbol{f}^*$.

At this point we are left with a circulation $\widetilde{\boldsymbol{g}}$ which does not contain any cycles supported only in $E$, and whose cost is at at least $\langle \boldsymbol{c}, \boldsymbol{g} \rangle \geq 0$; the latter inequality follows from the optimality of $(\boldsymbol{f}')^*$.

Using the fact that $\widetilde{\boldsymbol{g}}$ does not contain cycles supported in $E$, we get that the restriction to the arcs in $E$, $\widetilde{\boldsymbol{g}}_{|E}$ satisfies $\|\widetilde{\boldsymbol{g}}_{|E}\|_1 \leq m \cdot \|\widetilde{\boldsymbol{g}}_{|S}\|_1$, where $\widetilde{\boldsymbol{g}}_{|S}$ is the corresponding restriction to $S$. Therefore $|\langle \boldsymbol{c}, \widetilde{\boldsymbol{g}}_{|E} \rangle| \leq \|\boldsymbol{c}\|_\infty \cdot \|\widetilde{\boldsymbol{g}}_{|E}\|_1 \leq m \cdot \|\boldsymbol{c}\|_\infty \cdot \|\widetilde{\boldsymbol{g}}_{|S}\|_1$. Since by definition $\langle \boldsymbol{c}, \widetilde{\boldsymbol{g}}_{|S} \rangle = -c_\infty \cdot \|\widetilde{\boldsymbol{g}}_{|S}\|_1$, we conclude that $\langle \boldsymbol{c}, \widetilde{\boldsymbol{g}} \rangle = \langle \boldsymbol{c}, \widetilde{\boldsymbol{g}}_{|E} \rangle + \langle \boldsymbol{c}, \widetilde{\boldsymbol{g}}_{|S} \rangle \leq m \cdot \|\boldsymbol{c}\|_\infty \cdot \|\widetilde{\boldsymbol{g}}_{|S}\|_1 - c_\infty \cdot \|\widetilde{\boldsymbol{g}}_{|S}\|_1 = \|\widetilde{\boldsymbol{g}}_{|S}\|_1 \cdot (m\|\boldsymbol{c}\|_\infty - c_\infty) < 0$. This yields a contradiction, so a feasible $\boldsymbol{f}^*$ can not possibly exist. □

At this point, using the reduction given above, we can assume without loss of generality that the flow $\boldsymbol{f} = 1/2 \cdot \mathbf{1}$ routes the demand $\boldsymbol{d}$.

**Lemma 9.1.2.** *If $\boldsymbol{f} = 1/2 \cdot \mathbf{1}$ routes the input demand $\boldsymbol{d}$, then in the time dominated by $O(\log \log m)$ residual correction steps, we can produce a solution $\boldsymbol{f} = \boldsymbol{C}\boldsymbol{x}$ and a set of weights $\boldsymbol{w} \geq \mathbf{1}$ such that $\|\boldsymbol{w}\|_1 \leq 2m + 1$ and $\nabla F_\mu^{\boldsymbol{w}}(\boldsymbol{x}) = \mathbf{0}$, for $\mu \leq 2\|\boldsymbol{c}\|_2$.*

*Proof.* For this flow $\boldsymbol{f} = \boldsymbol{C}\boldsymbol{x}$ we have that $\boldsymbol{s}^+ = \boldsymbol{s}^- = \mathbf{1}/2$ and the corresponding residual satisfies $\nabla F_\mu^{\mathbf{1}}(\boldsymbol{x}) = \frac{\boldsymbol{C}^\top \boldsymbol{c}}{\mu}$. Per Definition 3.3.3 we can certify an upper bound on $\mathcal{E}_{\mathbf{1},\boldsymbol{s}}(\nabla F_\mu^{\mathbf{1}}(\boldsymbol{x}))$ using $\boldsymbol{y} = (\mathbf{0}; \boldsymbol{c}/\mu)$ which shows that

$$\mathcal{E}_{\mathbf{1},\boldsymbol{s}}(\nabla F_\mu^{\mathbf{1}}(\boldsymbol{x})) \leq \frac{1}{2} \cdot \left\langle (\boldsymbol{s}^+; \boldsymbol{s}^-)^2, (\mathbf{0}; \boldsymbol{c}/\mu)^2 \right\rangle = \frac{1}{2} \cdot \frac{1}{4\mu^2} \cdot \|\boldsymbol{c}\|_2^2 .$$

Therefore setting $\mu = \|\boldsymbol{c}\|_2$ we have that the energy is at most $1/8$, which fulfills the conditions required by Corollary 3.3.7 and Lemma 3.3.8 to produce in the time dominated by $O(\log \log m)$ residual correction steps a solution $\boldsymbol{x}$ and a weight vector $\boldsymbol{w} \geq \mathbf{1}$ such that $\|\boldsymbol{w} - \mathbf{1}\|_1 \leq 2m^{-9} \leq 1$ for which $\nabla F_{\mu'}^{\boldsymbol{w}}(\boldsymbol{x}) = \mathbf{0}$, where $\mu' \leq \|\boldsymbol{c}\|_2(1 + m^{-10}) \leq 2\|\boldsymbol{c}\|_2$. □

### 9.1.2 Preconditioning Proof

In this section we provide the proof for Lemma 3.4.7.

*Proof of Lemma 3.4.7.* Let $\widetilde{\boldsymbol{f}}_\star = \widetilde{\boldsymbol{f}} + \widetilde{\boldsymbol{f}}'$. Writing the optimality condition for (3.18) we obtain

$$\boldsymbol{C}_\star^\top (\boldsymbol{r}\widetilde{\boldsymbol{f}}_\star - \boldsymbol{h}) = \mathbf{0} , \tag{9.1}$$

where

$$\boldsymbol{r} = \begin{cases} \frac{\boldsymbol{w}^+}{(\boldsymbol{s}^+)^2} + \frac{\boldsymbol{w}^-}{(\boldsymbol{s}^-)^2} + R_p \cdot (\widetilde{\boldsymbol{f}})^{p-2} & \text{for edges in } E, \\ R_\star + R_p \cdot (\widetilde{\boldsymbol{f}}')^{p-2} & \text{for edges in } E'. \end{cases}$$

Since $\boldsymbol{C}_\star$ is a cycle basis for $G_\star$, the condition (9.1) implies that along any cycle in $G_\star$ the (appropriately signed) sum of weights $\boldsymbol{r}\widetilde{\boldsymbol{f}}_\star - \boldsymbol{h}$ is exactly 0. This means that these weights are determined by an embedding of the vertices of $G_\star$ onto the line, c.f. Lemma 9.1.3. Hence there exists a vector

$\boldsymbol{\phi} \in \mathbb{R}^{|V \cup \{v_\star\}|}$ such that for every edge $(u, v) \in E \cup E'$:

$$r_e(\widetilde{f}_\star)_e - h_e = \phi_u - \phi_v \,. \tag{9.2}$$

Also, to shorten notation, let us define

$$\bar{\boldsymbol{w}} = \boldsymbol{w}^+ + \boldsymbol{w}^- \,. \tag{9.3}$$

Next we will prove that the coordinates of $\boldsymbol{\phi}$ must lie within a short interval. The intuition here relies on the fact that the preconditioning edges make the graph $G_\star$ have good enough expansion; in turn, using an argument similar to [96] which was subsequently employed under various forms in [97, 118, 42] we argue that good expansion means that all the vertices are close to each other in the potential embedding.

Given a scalar $t$, let $S_t$ be the set of edges $(u, v)$ for which $t \in (\min\{\phi_u, \phi_v\}, \max\{\phi_u, \phi_v\})$. Our proof proceeds by lower bounding for each $t$:

$$\sum_{(u,v)\in S_t} \frac{\bar{w}_{uv}}{|\phi_u - \phi_v|} \geq \frac{\left(\sum_{(u,v)\in S_t} \frac{\sqrt{\bar{w}_{uv}}}{\sqrt{r_{uv}}}\right)^2}{\sum_{(u,v)\in S_t} \frac{|\phi_u - \phi_v|}{r_{uv}}} \,, \tag{9.4}$$

which we obtained using Cauchy-Schwarz.

Next we observe that the quantity in the denominator is upper bounded by

$$\sum_{(u,v)\in S_t} \frac{|\phi_u - \phi_v|}{r_{uv}} \leq \sum_{(u,v)\in S_t} \frac{|h_{uv}|}{r_{uv}} \,. \tag{9.5}$$

We can see why this is true by using the fact that $\widetilde{\boldsymbol{f}}_\star$ is a circulation, therefore along any cut $S_t$ one has

$$\sum_{\substack{(u,v)\in S_t \\ \phi_u > \phi_v}} (\widetilde{f}_\star)_{uv} = \sum_{\substack{(u,v)\in S_t \\ \phi_u \leq \phi_v}} (\widetilde{f}_\star)_{uv} \,,$$

i.e. the sum of the values of flows going from left to right is equal to the sum of values of flows going from right to left in the embedding.

By substituting $(\widetilde{f}_\star)_{uv}$ with the value determined from (9.2) we equivalently obtain that

$$\sum_{\substack{(u,v)\in S_t \\ \phi_u > \phi_v}} \frac{h_{uv} + |\phi_u - \phi_v|}{r_{uv}} = \sum_{\substack{(u,v)\in S_t \\ \phi_u \leq \phi_v}} \frac{h_{uv} - |\phi_u - \phi_v|}{r_{uv}} \,,$$

and by rearranging

$$\sum_{(u,v)\in S_t} \frac{|\phi_u - \phi_v|}{r_{uv}} = \sum_{\substack{(u,v)\in S_t \\ \phi_u > \phi_v}} \frac{-h_{uv}}{r_{uv}} + \sum_{\substack{(u,v)\in S_t \\ \phi_u \leq \phi_v}} \frac{h_{uv}}{r_{uv}} \leq \sum_{(u,v)\in S_t} \frac{|h_{uv}|}{r_{uv}} \,.$$

Therefore, plugging (9.5) into (9.4), we obtain

$$\sum_{(u,v)\in S_t} \frac{\bar{w}_{uv}}{|\phi_u - \phi_v|} \geq \frac{\left(\sum_{(u,v)\in S_t} \frac{\sqrt{\bar{w}_{uv}}}{\sqrt{r_{uv}}}\right)^2}{\sum_{(u,v)\in S_t} \frac{|h_{uv}|}{r_{uv}}} \geq \frac{\sum_{(u,v)\in S_t} \frac{\sqrt{\bar{w}_{uv}}}{\sqrt{r_{uv}}}}{\max \frac{|h_{uv}|/r_{uv}}{\sqrt{\bar{w}_{uv}}/\sqrt{r_{uv}}}} = \left(\sum_{(u,v)\in S_t} \frac{\sqrt{\bar{w}_{uv}}}{\sqrt{r_{uv}}}\right) \cdot \frac{1}{\|\boldsymbol{h}\bar{\boldsymbol{w}}^{-1/2}\boldsymbol{r}^{-1/2}\|_\infty} \,. \tag{9.6}$$

At this point we can prove that the all the values in $\boldsymbol{\phi}$ lie within a small interval. In order to do so we will crucially use the augmenting edges, which endow $G_\star$ with better expansion properties. Suppose w.l.o.g. that $\phi_{v_\star} = 0$. For every edge $e = (u, v) \in E \cup E'$, let $m_e = \min\{\phi_u, \phi_v\}$, $M_e = \max\{\phi_u, \phi_v\}$. For all edges in $e = (u, v_\star) \in E'$ (of which multiple copies can occur) we define

$$\bar{w}_e = 1 \,.$$

Hence we may define for $t \geq 0$:

$$F(t) = \sum_{\substack{(u,v)\in E\cup E' \\ M_{uv}\geq t}} \bar{w}_{uv} \cdot \frac{M_{uv} - \max\{m_{uv}, t\}}{|\phi_u - \phi_v|} \,, \tag{9.7}$$

which represents the sum of weighted fractions of edges that are on the right side of the cut at position $t$ on the real line.

Our goal is to show that since $F(t)$ decreases very fast, we do not need to increase $t$ very much before we run out of edges i.e. $F(t)$ becomes 0. Indeed, (9.6) offers a lower bound on the instantaneous decrease of $F(t)$, as $t$ increases. The reason is that all the augmenting edges $(v_\star, u)$ for $\phi_u > t$ appear in the cut. This also means that $F(t) > 0$ if $S_t \neq \emptyset$ and $F(t) = 0$ otherwise.

Intuitively (9.6) states that when slightly increasing $t$, $F(t)$ must decrease by a constant factor, scaled by the change in $t$. To formalize this we simply use the fact that

$$\sum_{(u,v)\in S_t} \frac{\sqrt{\bar{w}_{uv}}}{\sqrt{r_{uv}}} = \sum_{(u,v)\in S_t} \frac{\bar{w}_{uv}}{\sqrt{\bar{w}_{uv} r_{uv}}} \geq \sum_{(u,v_\star)\in S_t} \frac{\bar{w}_{uv_\star}}{\sqrt{\bar{w}_{uv_\star} r_{uv_\star}}} \geq \frac{1}{\max_{e\in E'} \sqrt{\bar{w}_e r_e}} \cdot \frac{F(t)}{2} \,, \tag{9.8}$$

which follows from the inequality

$$F(t) \leq \sum_{e\in S_t} \bar{w}_e \leq 2 \cdot \sum_{e\in S_t \cap E'} \bar{w}_e \,.$$

The latter is ensured by the fact that by definition each vertex $v \in V$ is incident to at least $\sum_{e\in E: e\sim v} \bar{w}_e$ augmenting edges in $E'$. Furthermore, even if $F(t)$ is very small but $S_t$ is still nonempty, we can use the lower bound

$$\sum_{(u,v)\in S_t} \frac{\sqrt{\bar{w}_{uv}}}{\sqrt{r_{uv}}} = \sum_{(u,v)\in S_t} \frac{\bar{w}_{uv}}{\sqrt{\bar{w}_{uv} r_{uv}}} \geq \frac{|S_t|}{\max_{e\in E'} \sqrt{\bar{w}_e r_e}} \geq \frac{1}{\max_{e\in E'} \sqrt{\bar{w}_e r_e}} \,. \tag{9.9}$$

Combining (9.6), (9.8), and (9.9) we obtain that if $S_t$ is nonempty, or equivalently $F(t) > 0$, then

$$\sum_{(u,v)\in S_t} \frac{\bar{w}_{uv}}{|\phi_u - \phi_v|} \geq \frac{1}{\|\boldsymbol{h}\bar{\boldsymbol{w}}^{-1/2}\boldsymbol{r}^{-1/2}\|_\infty \cdot \max_{e\in E'} \sqrt{\bar{w}_e r_e}} \cdot \max\{F(t)/2, 1\} \,. \tag{9.10}$$

As a matter of fact, this tells us that $F$ must decrease very fast, since from the definition (9.7) we have that $F$ is a continuous function, differentiable almost everywhere, such that at all points $t \geq 0$ where it is differentiable it satisfies

$$\frac{d}{dt}F(t) = -\sum_{\substack{(u,v)\in E\cup E' \\ M_{uv}\geq t}} \frac{\bar{w}_{uv}}{|\phi_u - \phi_v|} \leq -\sum_{(u,v)\in S_t} \frac{\bar{w}_{uv}}{|\phi_u - \phi_v|}\,.$$

Thus using (9.10) and Lemma 9.1.4 we obtain that $F(T) = 0$ for

$$\begin{aligned} T &= \|\boldsymbol{h}\bar{\boldsymbol{w}}^{-1/2}\boldsymbol{r}^{-1/2}\|_\infty \cdot \max_{e\in E'} \sqrt{\bar{w}_e r_e} \cdot (1 + 2\log F(0)) \\ &\leq \|\boldsymbol{h}\bar{\boldsymbol{w}}^{-1/2}\boldsymbol{r}^{-1/2}\|_\infty \cdot \max_{e\in E'} \sqrt{\bar{w}_e r_e} \cdot (1 + 2\log(4\|\boldsymbol{w}\|_1))\,, \end{aligned} \tag{9.11}$$

where the last inequality follows from accounting the weights of the augmenting edges in $E'$.

Using the identical argument for vertices embedded to the left of $v_\star$, we conclude that (9.11) yields an upper bound on $M_e - m_e$, and therefore, for all edges $e \in E$

$$\begin{aligned} |r_e\widetilde{f}_e - h_e| &\leq \max_{e\in E'} \sqrt{\bar{w}_e r_e} \cdot \|\boldsymbol{h}\bar{\boldsymbol{w}}^{-1/2}\boldsymbol{r}^{-1/2}\|_\infty \cdot 2(1 + 2\log(4\|\boldsymbol{w}\|_1)) \\ &\leq \max_{e\in E'} \sqrt{\bar{w}_e r_e} \cdot \|\boldsymbol{h}\bar{\boldsymbol{w}}^{-1/2}\boldsymbol{r}^{-1/2}\|_\infty \cdot 32\log\|\boldsymbol{w}\|_1\,. \end{aligned}$$

Now we use Lemma 3.4.6 to obtain that for all $(u, v_\star) = e \in E'$:

$$\bar{w}_e r_e = R_\star + R_p \cdot (\widetilde{f}'_e)^{p-2} \leq R_\star + R_p \cdot \|\widetilde{\boldsymbol{f}}'\|_\infty^{p-2}\,,$$

which ensures that

$$\left|\left(\frac{w_e^+}{(s_e^+)^2} + \frac{w_e^-}{(s_e^-)^2} + R_p(\widetilde{f}_e)^{p-2}\right)\widetilde{f}_e - h_e\right| \leq \left(R_\star + R_p \cdot \|\widetilde{\boldsymbol{f}}_\star\|_\infty^{p-2}\right)^{1/2} \cdot \left\|\frac{\boldsymbol{h}}{\sqrt{\bar{\boldsymbol{w}}\boldsymbol{r}}}\right\|_\infty \cdot 32\log\|\boldsymbol{w}\|_1\,, \tag{9.12}$$

and therefore that

$$\left|\left(\frac{w_e^+}{(s_e^+)^2} + \frac{w_e^-}{(s_e^-)^2}\right)\widetilde{f}_e - h_e\right| \leq \left|R_p \cdot (\widetilde{f}_e)^{p-1}\right| + \left(R_\star + R_p \cdot \|\widetilde{\boldsymbol{f}}_\star\|_\infty^{p-2}\right)^{1/2} \tag{9.13}$$

$$\cdot \left\|\frac{\boldsymbol{h}}{\sqrt{(\boldsymbol{w}^+ + \boldsymbol{w}^-)\left(\frac{\boldsymbol{w}^+}{(\boldsymbol{s}^+)^2} + \frac{\boldsymbol{w}^-}{(\boldsymbol{s}^-)^2}\right)}}\right\|_\infty \cdot 32\log\|\boldsymbol{w}\|_1\,, \tag{9.14}$$

which is what we needed.

We can furthermore establish a similar upper bound on

$$\begin{aligned}\left|\frac{w_e^+}{(s_e^+)^2}\cdot\widetilde{f}_e\right| &\le \left|\left(\frac{w_e^+}{(s_e^+)^2}+\frac{w_e^-}{(s_e^-)^2}+R_p\cdot\left(\widetilde{f}_e\right)^{p-2}\right)\widetilde{f}_e\right| \\ &\le |h_e|+\max_{e\in E'}\sqrt{\bar{w}_e r_e}\cdot\|\boldsymbol{h}\bar{\boldsymbol{w}}^{-1/2}\boldsymbol{r}^{-1/2}\|_\infty\cdot 32\log\|\boldsymbol{w}\|_1 \\ &\le |h_e|+\left(R_\star+R_p\cdot\|\widetilde{\boldsymbol{f}}_\star\|_\infty^{p-2}\right)^{1/2} \\ &\quad\cdot\left\|\frac{\boldsymbol{h}}{\sqrt{(\boldsymbol{w}^+-\boldsymbol{w}^-)\left(\frac{\boldsymbol{w}^+}{(\boldsymbol{s}^+)^2}+\frac{\boldsymbol{w}^-}{(\boldsymbol{s}^-)^2}\right)}}\right\|_\infty\cdot 32\log\|\boldsymbol{w}\|_1\,,\end{aligned}$$

and an identical upper bound on $\left|\frac{w_e^-}{(s_e^-)^2}\cdot\widetilde{f}_e\right|$.

□

**Lemma 9.1.3.** *Let $G$ be a graph with $n$ vertices and $m$ edges, and $\boldsymbol{C}$ be a matrix that encodes a cycle basis for $G$, in the sense that for any $\boldsymbol{x}$, $\boldsymbol{C}\boldsymbol{x}$ is a circulation in $G$ and for any circulation $\boldsymbol{f}$ in $G$, there exists a vector $\boldsymbol{x}$ such that $\boldsymbol{f}=\boldsymbol{C}\boldsymbol{x}$. Suppose that $\boldsymbol{y}\in\mathbb{R}^m$ is a vector such that $\boldsymbol{C}^\top\boldsymbol{y}=0$. Then there exists a vector $\boldsymbol{\phi}\in\mathbb{R}^n$ such that for all $(u,v)\in E$ one has that $y_{uv}=\phi_u-\phi_v$.*

*Proof.* By definition, the image of $\boldsymbol{C}$ is the space of circulations in $G$. Therefore the kernel of $\boldsymbol{C}^\top$ is orthogonal to the space of circulations in $G$, and therefore so is the vector $\boldsymbol{y}$. Now consider the incidence matrix $\boldsymbol{B}\in\mathbb{R}^{m\times n}$, which is constructed as follows: for each edge $(u,v)\in E$, add a row in with two nonzero entries, $+1$ at position $u$, and $-1$ at position $v$. One can easily see that $\ker\boldsymbol{B}^\top$ is exactly the space of circulations, as the $\boldsymbol{B}^\top$ operator acts on flows by returning the vector of demands that they route. Hence $\boldsymbol{y}$ lies in the image of $\boldsymbol{B}$, i.e. $\boldsymbol{y}=\boldsymbol{B}\boldsymbol{\phi}$. By the definition of $\boldsymbol{B}$, the conclusion follows. □

**Lemma 9.1.4.** *Let $F:\mathbb{R}_{\ge 0}\to\mathbb{R}_{\ge 0}$ be a continuous function, differentiable almost everywhere, such that $F(0)\ge 0$. Suppose that for all $t\ge 0$ where $F(t)>0$ and $dF(t)/dt$ exists, we have*

$$\frac{d}{dt}F(t)\le-\frac{1}{\alpha}\max\left\{\frac{F(t)}{2},1\right\},$$

*for some $\alpha>0$. Then*

$$F(\alpha(1+2\log F(0)))=0\,.$$

*Proof.* Let $T>0$ be any point for which $F(T)>0$. From the hypothesis we know that the instantaneous decrease in $F$ at all points $t\in[0,T]$ is at least $dt/\alpha$. Hence we have that:

$$F(T)\le\min_{0\le t\le T}F(t)-\frac{T-t}{\alpha}\,.$$

Furthermore, using the fact that the instantaneous decrease in $F(t)$ is at least $F(t)/(2\alpha)$, we solve the corresponding ODE to obtain that for all $t$,

$$F(t)\le F(0)\exp\left(-\frac{t}{2\alpha}\right).$$

Combining the two inequalities, and setting $t = 2\alpha \log F(0)$, we get that

$$F(T) \leq 1 - \frac{T - 2\alpha \log F(0)}{\alpha} = (1 + 2\log F(0)) - \frac{T}{\alpha}.$$

which implies that $T \leq \alpha(1 + 2\log F(0))$, since $F(T) \geq 0$. □

Finally, we discuss how to adapt the proof of Lemma 3.4.7 to obtain the Lemma 3.6.4.

*Proof of Lemma 3.6.4.* This proof is identical to that of Lemma 3.4.7. The main difference consists of including the $\boldsymbol{\alpha}$ factors in the definition of $\boldsymbol{r}$ for edges in $e$, more specifically, we let:

$$\boldsymbol{r} = \begin{cases} \frac{\boldsymbol{\alpha}^+ \boldsymbol{w}^+}{(\boldsymbol{s}^+)^2} + \frac{\boldsymbol{\alpha}^- \boldsymbol{w}^-}{(\boldsymbol{s}^-)^2} + R_p \cdot (\widetilde{\boldsymbol{f}})^{p-2} & \text{for edges in } E, \\ R_\star + R_p \cdot (\widetilde{\boldsymbol{f}}')^{p-2} & \text{for edges in } E'. \end{cases}$$

Using this new definition for $\boldsymbol{r}$, the remaining proof carries over by following the exact same steps as before. □

### 9.1.3 Solving the Mixed Objective

#### Invoking the Solver

Producing a high precision solution to the regularized objective in (3.18) can be done efficiently in our particular setting, where we aim to optimize a mixed $\ell_2$-$\ell_p$ objective in the space of circulations. To this extent we use the following result from [103], also restated and improved in [2].

**Theorem 9.1.5.** *For any $p \geq 2$, given weights $\boldsymbol{r} \in \mathbb{R}^{|E|}_{\geq 0}$, and a cost vector $\boldsymbol{g} \in \mathbb{R}^{|E|}$ define the function defined over circulations $\boldsymbol{f}$ in $G$:*

$$val(\boldsymbol{f}) = \sum_e g_e f_e + r_e f_e^2 + |f_e|^p.$$

*Given any circulation $\boldsymbol{f}$ for which all the parameters are bounded by $2^{(\log n)^{O(1)}}$ we can compute a circulation $\widetilde{\boldsymbol{f}}$ such that*

$$val(\widetilde{\boldsymbol{f}}) - OPT \leq \frac{1}{2^{(\log m)^{O(1)}}}(val(\boldsymbol{f}) - OPT) + \frac{1}{2^{(\log m)^{O(1)}}}$$

*in $2^{O(p^{3/2})} m^{1+O(1/\sqrt{p})}$ time.*

For large values of $p$ this solves the regularized objective defined in (3.18) to high precision in almost linear time $O\left(m^{1+o(1)}\right)$, which is comparably fast to the time required to minimize a convex quadratic function in the space of circulations via fast Laplacian solvers.

#### Discussion on Solver Errors

Throughout Chapter 3 we assume that the solutions to the regularized objective are exact. This is not exactly true due to the approximate nature of the solver specified in Theorem 9.1.5. Instead, we argue that the entire analysis we showed carries over even if the solver returns a solution which carries some small error.

Consider the equivalent problem of minimizing the negative of the objective $\Phi(\boldsymbol{x})$ stated in (3.18). We verify that at a point $\boldsymbol{x}$ such that the current flow $\widetilde{\boldsymbol{f}} = \boldsymbol{C}_\star \boldsymbol{x}$, its Hessian is

$$\nabla^2 \Phi(\boldsymbol{x}) = \boldsymbol{C}_\star^\top \boldsymbol{D} \boldsymbol{C}_\star$$

where $\boldsymbol{D}$ is a diagonal matrix whose entries are defined such that

$$(\boldsymbol{D})_{e,e} = \begin{cases} \frac{w_e^+}{(s_e^+)^2} + \frac{w_e^-}{(s_e^-)^2} + (p-1) R_p (\widetilde{f}_e)^{p-2} & \text{if } e \in E\,, \\ R_\star + R_p (p-1)(\widetilde{f}_e)^{p-2} & \text{if } e \in E'\,. \end{cases}$$

Our choice of regularization parameters yield an upper bound on $\|\widetilde{\boldsymbol{f}}\|_\infty$ as we showed in the proof of Lemma 3.4.6 (Equation (3.43)), which together with our choice of regularization parameters (Section 3.4.2) and the invariants that $\mathbf{1} \le \boldsymbol{w}$ and $\|\boldsymbol{w}\|_1 = O(m)$ ensure that $R_p(\widetilde{f}_e)^{p-2} \le \widetilde{O}((\delta^2 m)^3) = o(m)$. Assuming that we always maintain all our slacks large enough i.e. for all $e$: $\tau^{-1} \le f_e \le 1 - \tau^{-1}$, for $\tau = m^{O(1)}$, we see that $\boldsymbol{D}$ is always well-conditioned in the sense that all of its diagonal entries are between $\tau^{-2}$ and $M = O(m\tau^2)$. We will discuss later how to maintain this property.[1]

Under this assumption we can therefore use basic tools from convex analysis to argue that after optimizing $\Phi$ to high precision, the gradient of the returned solution is small. Let $\boldsymbol{x}$ be the solution returned by the high-precision solver such that $\Phi(\boldsymbol{x}) \le \Phi(\boldsymbol{x}^*) + \varepsilon$. Letting $\nabla \Phi(\boldsymbol{x}) = \boldsymbol{C}_\star^\top \boldsymbol{h}$ for some vector $\boldsymbol{h}$, we can write

$$\begin{aligned} \|\nabla \Phi(\boldsymbol{x})\|^*_{\boldsymbol{C}_\star^\top \boldsymbol{C}_\star} &= \left\| \int_0^1 \nabla^2 \Phi((1-t)\boldsymbol{x}^* + t\boldsymbol{x})(\boldsymbol{x} - \boldsymbol{x}^*) dt \right\|^*_{\boldsymbol{C}_\star^\top \boldsymbol{C}_\star} \\ &\le \max_{t \in [0,1]} \left\| \nabla^2 \Phi((1-t)\boldsymbol{x}^* + t\boldsymbol{x}) \right\|_{\|\cdot\|_{\boldsymbol{C}_\star^\top \boldsymbol{C}_\star} \to \|\cdot\|^*_{\boldsymbol{C}_\star^\top \boldsymbol{C}_\star}} \cdot \|\boldsymbol{x} - \boldsymbol{x}^*\|_{\boldsymbol{C}_\star^\top \boldsymbol{C}_\star} \\ &\le \max_{\substack{\boldsymbol{D} \text{ diagonal} \\ \boldsymbol{D}_{ii} \in [M^{-1}, M]}} \left\| \boldsymbol{C}_\star^\top \boldsymbol{D} \boldsymbol{C}_\star \right\|_{\|\cdot\|_{\boldsymbol{C}_\star^\top \boldsymbol{C}_\star} \to \|\cdot\|^*_{\boldsymbol{C}_\star^\top \boldsymbol{C}_\star}} \cdot \|\boldsymbol{x} - \boldsymbol{x}^*\|_{\boldsymbol{C}_\star^\top \boldsymbol{C}_\star}\,. \end{aligned}$$

Here we use the notation

$$\begin{aligned} \|\boldsymbol{x}\|_{\boldsymbol{C}_\star^\top \boldsymbol{C}_\star} &= \|\boldsymbol{C}_\star \boldsymbol{x}\|_2\,, \\ \|\boldsymbol{f}\|^*_{\boldsymbol{C}_\star^\top \boldsymbol{C}_\star} &= \max_{\boldsymbol{x}: \|\boldsymbol{x}\|_{\boldsymbol{C}_\star^\top \boldsymbol{C}_\star} \le 1} \langle \boldsymbol{f}, \boldsymbol{x} \rangle\,, \end{aligned}$$

and

$$\|\boldsymbol{A}\|_{\|\cdot\|_{\boldsymbol{C}_\star^\top \boldsymbol{C}_\star} \to \|\cdot\|^*_{\boldsymbol{C}_\star^\top \boldsymbol{C}_\star}} = \max_{\boldsymbol{x}: \|\boldsymbol{x}\|_{\boldsymbol{C}_\star^\top \boldsymbol{C}_\star} \le 1} \|\boldsymbol{A}\boldsymbol{x}\|^*_{\boldsymbol{C}_\star^\top \boldsymbol{C}_\star}\,.$$

Using these definitions we can further write

$$\begin{aligned} \|\boldsymbol{C}_\star^\top \boldsymbol{D} \boldsymbol{C}_\star\|_{\|\cdot\|_{\boldsymbol{C}_\star^\top \boldsymbol{C}_\star} \to \|\cdot\|^*_{\boldsymbol{C}_\star^\top \boldsymbol{C}_\star}} &= \max_{\boldsymbol{x}: \|\boldsymbol{x}\|_{\boldsymbol{C}_\star^\top \boldsymbol{C}_\star} \le 1} \|\boldsymbol{C}_\star^\top \boldsymbol{D} \boldsymbol{C}_\star \boldsymbol{x}\|^*_{\boldsymbol{C}_\star^\top \boldsymbol{C}_\star} \\ &= \max_{\boldsymbol{x}: \|\boldsymbol{C}_\star \boldsymbol{x}\|_2 \le 1} \max_{\boldsymbol{y}: \|\boldsymbol{C}_\star \boldsymbol{y}\|_2 \le 1} \langle \boldsymbol{C}_\star^\top \boldsymbol{D} \boldsymbol{C}_\star \boldsymbol{x}, \boldsymbol{y} \rangle \\ &= \max_{\boldsymbol{f}, \boldsymbol{g}: \|\boldsymbol{f}\|_2 \le 1, \|\boldsymbol{g}\|_2 \le 1} \langle \boldsymbol{D}\boldsymbol{f}, \boldsymbol{g} \rangle \\ &\le \max_i \boldsymbol{D}_{ii}\,. \end{aligned}$$

[1] Alternatively, we can enforce lower and upper bounds on the entries of $\boldsymbol{D}$ by adding an additional small quadratic regularizer to the edges in $E$.

Next we bound the distance $\|\boldsymbol{x}-\boldsymbol{x}^*\|_{\boldsymbol{C}_\star{}^\top \boldsymbol{C}_\star}$. To do so, we Taylor expand:

$$\begin{aligned}\Phi(\boldsymbol{x})-\Phi(\boldsymbol{x}^*) &= \frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}^*)^\top\left(\int_0^1 \nabla^2\Phi((1-t)\boldsymbol{x}^*+t\boldsymbol{x})dt\right)(\boldsymbol{x}-\boldsymbol{x}^*)\\ &= \frac{1}{2}\left(\left\|\int_0^1 \nabla^2\Phi((1-t)\boldsymbol{x}^*+t\boldsymbol{x})dt\right\|_{\|\cdot\|_{\boldsymbol{C}_\star{}^\top\boldsymbol{C}_\star}\to\|\cdot\|^*_{\boldsymbol{C}_\star{}^\top\boldsymbol{C}_\star}}\|\boldsymbol{x}-\boldsymbol{x}^*\|_{\boldsymbol{C}_\star{}^\top\boldsymbol{C}_\star}\right)^2\\ &\geq \frac{1}{2}\left(\min_i \boldsymbol{D}_{ii}\cdot\|\boldsymbol{x}-\boldsymbol{x}^*\|_{\boldsymbol{C}_\star{}^\top\boldsymbol{C}_\star}\right)^2.\end{aligned}$$

Since the solution we obtain satisfies $\Phi(\boldsymbol{x})-\Phi(\boldsymbol{x}^*)\leq\varepsilon$, we thus infer that

$$\|\boldsymbol{x}-\boldsymbol{x}^*\|_{\boldsymbol{C}_\star{}^\top\boldsymbol{C}_\star}\leq\frac{\sqrt{2\varepsilon}}{\min_i \boldsymbol{D}_{ii}}.$$

Plugging into the upper bound on the gradient norm, we obtain that:

$$\|\nabla\Phi(\boldsymbol{x})\|^*_{\boldsymbol{C}_\star{}^\top\boldsymbol{C}_\star}\leq\frac{\max_i \boldsymbol{D}_{ii}}{\min_i \boldsymbol{D}_{ii}}\cdot\sqrt{2\varepsilon}.$$

Since we have that $\nabla\Phi(\boldsymbol{x})=\boldsymbol{C}_\star{}^\top\boldsymbol{h}$, this implies an upper bound on the norm of $\boldsymbol{h}$. Indeed we have that $\|\boldsymbol{C}_\star{}^\top\boldsymbol{h}\|^*_{\boldsymbol{C}_\star{}^\top\boldsymbol{C}_\star}=\max_{\boldsymbol{y}:\|\boldsymbol{C}_\star\boldsymbol{y}\|\leq1}\langle\boldsymbol{C}_\star{}^\top\boldsymbol{h},\boldsymbol{y}\rangle=\max_{\boldsymbol{y}:\|\boldsymbol{C}_\star\boldsymbol{y}\|\leq1}\langle\boldsymbol{h},\boldsymbol{C}_\star\boldsymbol{y}\rangle=\|\boldsymbol{h}\|_2$. Therefore the same upper bound also holds for $\|\boldsymbol{h}\|_2$. Hence under the assumption that all slacks are lower bounded by a small polynomial, we can obtain that $\|\boldsymbol{h}\|_2\leq M^{-1}$ by setting $\varepsilon$ to an appropriately small polynomial.

Therefore after (approximately) solving the regularized objective (3.18) we can set its gradient exactly to 0 only by slightly perturbing the linear term. This extra error, in turn, gets passed to Lemma 3.4.20 which removes the error by slightly modifying the weight vector $\boldsymbol{w}$. Although this operation may enable some weights to decrease by a tiny amount below 1, this can be prevented simply by uniformly upscaling all the weights. The centrality property will be then achieved for a slightly smaller parameter $\mu$ and the weight increase will be upper bounded by an arbitrary inverse polynomial in $m$ thus negligible.

Finally, Lemma 9.1.6 ensures that as a matter of fact all slacks are always polynomially lower bounded, and so our claim holds. Furthermore, all calls to the solver in Theorem 9.1.5 are made on instances where all parameters are well-conditioned, as required.

**Lemma 9.1.6** (Slack lower bound)**.** *When invoking the mixed objective solver during the procedure described in Theorem 3.4.26, at all times we have that* $\min\{s_e^+,s_e^-\}\geq 1/m^{O(1)}$ *for all edges* $e\in E$.

*Proof.* We note that the stretch condition from Lemma 3.4.7 holds for any $\boldsymbol{h}$. We apply it once for $\boldsymbol{h}=\frac{\boldsymbol{w}^+}{\boldsymbol{s}^+}-\frac{\boldsymbol{w}^-}{\boldsymbol{s}^-}$ and once for $\boldsymbol{h}=\frac{\boldsymbol{c}}{\mu}$. For the former, since

$$\left\|\frac{\left|\frac{\boldsymbol{w}^+}{\boldsymbol{s}^+}-\frac{\boldsymbol{w}^-}{\boldsymbol{s}^-}\right|}{\sqrt{(\boldsymbol{w}^++\boldsymbol{w}^-)\left(\frac{\boldsymbol{w}^+}{(\boldsymbol{s}^+)^2}+\frac{\boldsymbol{w}^-}{(\boldsymbol{s}^-)^2}\right)}}\right\|_\infty\leq 1,$$

we get

$$\left\| \left( \frac{\boldsymbol{w}^+}{(\boldsymbol{s}^+)^2} + \frac{\boldsymbol{w}^-}{(\boldsymbol{s}^-)^2} + R_p \cdot \widetilde{\boldsymbol{f}}^{p-2} \right) \widetilde{\boldsymbol{f}} - \left( \frac{\boldsymbol{w}^+}{\boldsymbol{s}^+} - \frac{\boldsymbol{w}^-}{\boldsymbol{s}^-} \right) \right\|_\infty = O(\delta \|\boldsymbol{w}\|_1 \cdot \log \|\boldsymbol{w}\|_1) = O(m)\,, \quad (9.15)$$

and for the latter, since $\left\| \frac{\frac{\boldsymbol{c}}{\mu}}{\sqrt{(\boldsymbol{w}^+ + \boldsymbol{w}^-)\left(\frac{\boldsymbol{w}^+}{(\boldsymbol{s}^+)^2} + \frac{\boldsymbol{w}^-}{(\boldsymbol{s}^-)^2}\right)}} \right\|_\infty \leq \frac{W \cdot m^4}{2}$, we get

$$\left\| \left( \frac{\boldsymbol{w}^+}{(\boldsymbol{s}^+)^2} + \frac{\boldsymbol{w}^-}{(\boldsymbol{s}^-)^2} + R_p \cdot \widetilde{\boldsymbol{f}}^{p-2} \right) \widetilde{\boldsymbol{f}} - \frac{\boldsymbol{c}}{\mu} \right\|_\infty = O(\delta \|\boldsymbol{w}\|_1 \cdot W \cdot m^4 \cdot \log \|\boldsymbol{w}\|_1) = m^{O(1)}\,. \quad (9.16)$$

Combining (9.15) and (9.16) and using the triangle inequality we get that

$$\left\| \frac{\boldsymbol{w}^+}{\boldsymbol{s}^+} - \frac{\boldsymbol{w}^-}{\boldsymbol{s}^-} \right\|_\infty \leq \left\| \frac{\boldsymbol{c}}{\mu} \right\|_\infty + m^{O(1)} = m^{O(1)}\,.$$

Since $\max\{s_e^+, s_e^-\} \geq \frac{1}{2}$, using the triangle inequality and the fact that $1 \leq w_e^+, w_e^- \leq O(m)$ in the above inequality implies that $\min\{s_e^+, s_e^-\} \geq 1/m^{O(1)}$. □

### 9.1.4 Strengthening the Mixed Objective Solver

In this section we prove that the $\ell_2$-$\ell_p$ solver from [103] can be extended to handle a broader class of optimization problems on graphs. This will be useful in order to solve the optimization problem required by the improved method we present in Section 3.6. We next state the main lemma we prove in this section.

**Lemma 9.1.7.** *Let a graph $G = (V, E)$ with $m$ edges, and a family of functions $\{g_e\}_{e \in E}$, $g_e : \mathbb{R} \to \mathbb{R}$ such that each function satisfies $\left| \ln \frac{g_e''(x)}{g_e''(0)} \right| \leq \alpha$, for all $x \in \mathbb{R}$. Let the function defined over circulations $\boldsymbol{f}$ in $G$:*

$$\widetilde{val}(\boldsymbol{f}) = \sum_e g_e(f_e) + |f_e|^p\,.$$

*We can compute a circulation $\widetilde{\boldsymbol{f}}$ such that*

$$\widetilde{val}(\widetilde{\boldsymbol{f}}) - OPT \leq \frac{1}{2^{(\log m)^{O(1)}}} (\widetilde{val}(\boldsymbol{f}) - OPT) + \frac{1}{2^{(\log m)^{O(1)}}}$$

*in $2^{O(\alpha + p^{3/2})} m^{1 + O(1/\sqrt{p})}$ time.*

The proof closely follows the lines of the iterative refinement proofs from the original paper [1]. Intuitively, since $g$ has bounded second order derivatives, it is always well approximated by a quadratic. Therefore the solver from Theorem 9.1.5 can be used iteratively to improve the error of the current solution.

Iterative refinement is based on the following basic statement.

**Lemma 9.1.8.** *Let a linear subspace $\mathcal{X} \subseteq \mathbb{R}^m$ , let $h, k : \mathcal{X} \to \mathbb{R}$ be convex twice-differentiable functions, and let $\boldsymbol{x}, \boldsymbol{x}^* \in \mathbb{R}^m$ such that $\boldsymbol{x}^*$ minimizes $h$. Suppose that $k$ satisfies*

$$k(c\boldsymbol{\delta}) \leq c^2 k(\boldsymbol{\delta})\,,$$

*for all* $c \in \mathbb{R}$, $\boldsymbol{\delta} \in \mathcal{X}$, *such that*

$$k(\boldsymbol{\delta}) \leq h(\boldsymbol{x}+\boldsymbol{\delta}) - h(\boldsymbol{x}) - \langle \nabla h(\boldsymbol{x}), \boldsymbol{\delta} \rangle \leq \beta \cdot k(\boldsymbol{\delta}),$$

*for any* $\boldsymbol{\delta} \in \mathcal{X}$. *Then letting*

$$\boldsymbol{\delta}^* = \arg\min_{\boldsymbol{\delta} \in \mathcal{X}} \langle \nabla h(\boldsymbol{x}), \boldsymbol{\delta} \rangle + k(\boldsymbol{\delta})$$

*and* $\boldsymbol{\delta}^\sharp \in \mathcal{X}$ *such that*

$$\langle \nabla h(\boldsymbol{x}), \boldsymbol{\delta}^\sharp \rangle + k(\boldsymbol{\delta}^\sharp) \leq \frac{1}{\gamma} \left( \langle \nabla h(\boldsymbol{x}), \boldsymbol{\delta}^* \rangle + k(\boldsymbol{\delta}^*) \right) + \varepsilon,$$

*one has that*

$$h(\boldsymbol{x}) - h(\boldsymbol{x}+\boldsymbol{\delta}^\sharp) \geq \frac{1}{\beta\gamma} \left( h(\boldsymbol{x}) - h(\boldsymbol{x}^*) \right) - \frac{\varepsilon}{\beta}.$$

*Proof.* Using the hypothesis, we have that

$$\begin{aligned} h(\boldsymbol{x}+\boldsymbol{\delta}^\sharp/\beta) &\leq h(\boldsymbol{x}) + \langle \nabla h(\boldsymbol{x}), \boldsymbol{\delta}^\sharp/\beta \rangle + \beta \cdot k(\boldsymbol{\delta}^\sharp/\beta) \\ &\leq h(\boldsymbol{x}) + \frac{1}{\beta} \left( \langle \nabla h(\boldsymbol{x}), \boldsymbol{\delta}^\sharp \rangle + k(\boldsymbol{\delta}^\sharp) \right), \end{aligned}$$

where we used the right hand side of the sandwiching inequality from the hypothesis, and the fact that $k(\boldsymbol{\delta}^\sharp/\beta) \leq k(\boldsymbol{\delta}^\sharp)/\beta^2$.

Next we plug in the relation between $\boldsymbol{\delta}^*$ and $\boldsymbol{\delta}^\sharp$ to obtain:

$$\begin{aligned} h(\boldsymbol{x}+\boldsymbol{\delta}^\sharp/\beta) &\leq h(\boldsymbol{x}) + \frac{1}{\beta\gamma} \left( \langle \nabla h(\boldsymbol{x}), \boldsymbol{\delta}^* \rangle + k(\boldsymbol{\delta}^*) \right) + \frac{\varepsilon}{\beta} \\ &\leq h(\boldsymbol{x}) + \frac{1}{\beta\gamma} \left( \langle \nabla h(\boldsymbol{x}), \boldsymbol{x}^* - \boldsymbol{x} \rangle + k(\boldsymbol{x}^* - \boldsymbol{x}) \right) + \frac{\varepsilon}{\beta}, \end{aligned}$$

where the latter inequality follows from the fact that $\boldsymbol{\delta}^*$ minimizes $\langle \nabla h(\boldsymbol{x}), \boldsymbol{\delta}^* \rangle + k(\boldsymbol{\delta}^*)$, so plugging in $\boldsymbol{x}^* - \boldsymbol{x}$ as an argument can only increase the sum of these two terms. Finally, we plug in the left hand side of the sandwiching inequality, i.e.

$$\langle \nabla h(\boldsymbol{x}), \boldsymbol{x}^* - \boldsymbol{x} \rangle + k(\boldsymbol{x}^* - \boldsymbol{x}) \leq \boldsymbol{h}(\boldsymbol{x}^*) - \boldsymbol{h}(\boldsymbol{x}),$$

which combined with the previous inequality yields,

$$h(\boldsymbol{x}) - h(\boldsymbol{x}+\boldsymbol{\delta}^\sharp/\beta) \geq \frac{1}{\beta\gamma} \left( h(\boldsymbol{x}) - h(\boldsymbol{x}^*) \right) - \frac{\varepsilon}{\beta},$$

which is what we needed. □

We apply Lemma 9.1.8 for a custom choice of $k$, which is dictated by the specific family of functions $\{g_e\}_{e \in E}$..

In order to do so, we also require a sandwiching inequality for the $\sum f_e^p$ term. We use the following inequality from [103].

**Lemma 9.1.9** ([103], p. 11)**.** *Let* $x, \delta \in \mathbb{R}$ *and* $p \geq 2$. *Then*

$$2^{-O(p)}(x^{p-2}\delta^2 + \delta^p) \leq (x+\delta)^p - x^p - px^{p-1} \leq 2^{O(p)}(x^{p-2}\delta^2 + \delta^p).$$

Using Lemmas 9.1.8 and 9.1.9, we can now define an appropriate function $k$ for each circulation $\boldsymbol{f}$. In order to do so, we use the following simple lemma.

**Lemma 9.1.10.** *Let $\widetilde{val}(\boldsymbol{f})$ specified as in Lemma 9.1.7, and let a circulation $\boldsymbol{f}$. Then one has that for any circulation $\boldsymbol{\delta}$:*

$$\sum_e \left( \frac{e^{-\alpha} g_e''(0)}{2} \delta_e^2 + 2^{-O(p)} (f_e^{p-2} \delta_e^2 + \delta_e^p) \right) \leq \widetilde{val}(\boldsymbol{f} + \boldsymbol{\delta}) - \widetilde{val}(\boldsymbol{f}) - \langle \nabla \widetilde{val}(\boldsymbol{f}), \boldsymbol{\delta} \rangle$$
$$\leq \sum_e \left( \frac{e^{\alpha} g_e''(0)}{2} \delta_e^2 + 2^{O(p)} (f_e^{p-2} \delta_e^2 + \delta_e^p) \right) .$$

*Proof.* We require lower bounding and upper bounding the terms of order higher than 1, after expanding $\widetilde{val}(\boldsymbol{f} + \boldsymbol{\delta})$ around $\widetilde{val}(\boldsymbol{f})$. To do so, we first notice that by the hypothesis in Lemma 9.1.7 we have

$$\frac{e^{-\alpha} g_e''(0)}{2} \delta^2 \leq g_e(x + \delta) - g_e(x) - g_e'(x)\delta \leq \frac{e^{\alpha} g_e''(0)}{2} \delta^2 .$$

Similarly, we use Lemma 9.1.9 to lower and upper bound the higher order terms of $f_e^p$ for each $e$. Combining, we obtain the desired claim. □

Now we can prove that we can decrease $\widetilde{val}(f) - OPT$ very fast, which in turn enables us to prove the main lemma in this section.

*Proof of Lemma 9.1.7.* We use Lemma 9.1.8 where we define the functions $k_e$ based on Lemma 9.1.10. More precisely, for each $e \in E$ we let

$$k(\boldsymbol{\delta}) = \sum_e k_e(\delta_e) ,$$

where

$$k_e(\delta_e) = \frac{e^{-\alpha} g_e''(0)}{2} \delta_e^2 + 2^{-O(p)} (f_e^{p-2} \delta_e^2 + \delta_e^p) .$$

Using Lemma 9.1.10 we verify that

$$k(\boldsymbol{\delta}) \leq \widetilde{val}(\boldsymbol{f} + \boldsymbol{\delta}) - \widetilde{val}(\boldsymbol{f}) - \langle \nabla \widetilde{val}(\boldsymbol{f}), \boldsymbol{\delta} \rangle \leq \max\{e^{2\alpha}, 2^{O(p)}\} \cdot k(\boldsymbol{\delta}) .$$

We use the solver from Theorem 9.1.5 to approximately minimize $k(\boldsymbol{\delta})$ plus the corresponding linear term. Then, for our specific setting, we can apply Lemma 9.1.8 with

$$\gamma = 1 + \frac{1}{2^{(\log m)^{O(1)}}} ,$$
$$\varepsilon = \frac{1}{2^{(\log m)^{O(1)}}} ,$$
$$\beta = \max\{e^{2\alpha}, 2^{O(p)}\} ,$$

to get that the newly obtained iterate $\boldsymbol{f}'$ satisfies

$$\widetilde{val}(\boldsymbol{f}) - \widetilde{val}(\boldsymbol{f}') \geq \frac{1}{\beta\gamma} (\widetilde{val}(\boldsymbol{f}) - OPT) - \frac{\varepsilon}{\beta} .$$

Therefore executing this step for $T$ iterations, we obtain $\boldsymbol{f}_T$ such that

$$\begin{aligned}\widetilde{val}(\boldsymbol{f}_T) - OPT &\leq \left(1 - \frac{1}{\beta\gamma}\right)^T \left(\widetilde{val}(\boldsymbol{f}) - OPT\right) + \frac{\varepsilon}{\beta}\left(\sum_{t=0}^{T-1}\left(1 - \frac{1}{\beta\gamma}\right)^t\right) \\ &\leq \left(1 - \frac{1}{\beta\gamma}\right)^T \left(\widetilde{val}(\boldsymbol{f}) - OPT\right) + \varepsilon\gamma\,.\end{aligned}$$

Hence setting $T = \beta\gamma(\log m)^{O(1)}$ we make

$$\widetilde{val}(\boldsymbol{f}_T) - OPT \leq \frac{1}{2^{(\log m)^{O(1)}}}(\widetilde{val}(\boldsymbol{f}) - OPT) + \frac{1}{2^{(\log m)^{O(1)}}}\,.$$

Hence we require $T = (e^{2\alpha} + 2^{O(p)})\log^{O(1)} m$ iterations to obtain the target accuracy. Together with the running time guarantee from Theorem 9.1.5 we obtain the claim. □

### 9.1.5 Deferred Proofs

**Proof of Lemma 3.3.1**

*Proof.* Writing first order optimality conditions for (3.3), we have

$$\nabla F_\mu^{\boldsymbol{w}}(\boldsymbol{x}) = \frac{\boldsymbol{C}^\top \boldsymbol{c}}{\mu} + \boldsymbol{C}^\top\left(\frac{\boldsymbol{w}^+}{\boldsymbol{s}^+} - \frac{\boldsymbol{w}^-}{\boldsymbol{s}^-}\right).$$

Setting this to $\mathbf{0}$ yields the conclusion. Now letting $\boldsymbol{y} = \mu \cdot \boldsymbol{w}/\boldsymbol{s}$, we see that $\boldsymbol{y} \geq \mathbf{0}$ since both weights $\boldsymbol{w}$ and slacks $\boldsymbol{s}$ are non-negative, and $\boldsymbol{C}^\top(\boldsymbol{y}^+ - \boldsymbol{y}^-) = -\boldsymbol{C}^\top c$, from which we conclude that $\boldsymbol{y}$ is a feasible dual vector. Finally, we can write the duality gap as

$$\begin{aligned}&\langle \boldsymbol{c}, \boldsymbol{C}\boldsymbol{x}\rangle + \langle \mathbf{1} - \boldsymbol{f}_0, \boldsymbol{y}^+\rangle + \langle \boldsymbol{f}_0, \boldsymbol{y}^-\rangle = -\langle \boldsymbol{C}^\top(\boldsymbol{y}^+ - \boldsymbol{y}^-), \boldsymbol{x}\rangle + \langle \mathbf{1} - \boldsymbol{f}_0, \boldsymbol{y}^+\rangle + \langle \boldsymbol{f}_0, \boldsymbol{y}^-\rangle \\ &= \langle \mathbf{1} - \boldsymbol{f}_0 - \boldsymbol{C}\boldsymbol{x}, \boldsymbol{y}^+\rangle + \langle \boldsymbol{f}_0 + \boldsymbol{C}\boldsymbol{x}, \boldsymbol{y}^-\rangle = \langle \boldsymbol{s}^+, \mu \cdot \boldsymbol{w}^+/\boldsymbol{s}^+\rangle + \langle \boldsymbol{s}^-, \mu \cdot \boldsymbol{w}^-/\boldsymbol{s}^-\rangle \\ &= \mu\|\boldsymbol{w}\|_1\,.\end{aligned}$$

□

**Proof of Lemma 3.3.4**

*Proof.* We write

$$\begin{aligned}\mathcal{E}_{\boldsymbol{w},\boldsymbol{s}}(\boldsymbol{h}) &= \min_{\widetilde{\boldsymbol{y}}:\boldsymbol{C}^\top(\widetilde{\boldsymbol{y}}+\boldsymbol{h})=\mathbf{0}} \frac{1}{2}\sum_{e\in E}\left(\frac{w_e^+}{(s_e^+)^2} + \frac{w_e^-}{(s_e^-)^2}\right)^{-1}(\widetilde{y}_e)^2 \\ &= \min_{\widetilde{\boldsymbol{y}}}\max_{\widetilde{\boldsymbol{x}}} \frac{1}{2}\sum_{e\in E}\left(\frac{w_e^+}{(s_e^+)^2} + \frac{w_e^-}{(s_e^-)^2}\right)^{-1}(\widetilde{y}_e)^2 - \langle \widetilde{\boldsymbol{x}}, \boldsymbol{C}^\top(\widetilde{\boldsymbol{y}} + \boldsymbol{h})\rangle \\ &= \max_{\widetilde{\boldsymbol{x}}}\min_{\widetilde{\boldsymbol{y}}} \frac{1}{2}\sum_{e\in E}\left(\frac{w_e^+}{(s_e^+)^2} + \frac{w_e^-}{(s_e^-)^2}\right)^{-1}(\widetilde{y}_e)^2 - \langle \boldsymbol{C}\widetilde{\boldsymbol{x}}, \widetilde{\boldsymbol{y}} + \boldsymbol{h}\rangle\end{aligned}$$

The solution to the inner optimization problem is $\widetilde{\boldsymbol{y}} = \left(\frac{\boldsymbol{w}^+}{(\boldsymbol{s}^+)^2} + \frac{\boldsymbol{w}^-}{(\boldsymbol{s}^-)^2}\right) \boldsymbol{C}\widetilde{\boldsymbol{x}}$, therefore by substituting we get

$$\begin{aligned}\mathcal{E}_{\boldsymbol{w},\boldsymbol{s}}(\boldsymbol{h}) &= \max_{\widetilde{\boldsymbol{x}}} \langle \boldsymbol{h}, \boldsymbol{C}\widetilde{\boldsymbol{x}}\rangle - \frac{1}{2}\sum_{e\in E}\left(\frac{w_e^+}{(s_e^+)^2} + \frac{w_e^-}{(s_e^-)^2}\right)(\boldsymbol{C}\widetilde{\boldsymbol{x}})_e^2 \\ &= \max_{\widetilde{\boldsymbol{f}}=\boldsymbol{C}\widetilde{\boldsymbol{x}}} \langle \boldsymbol{h}, \widetilde{\boldsymbol{f}}\rangle - \frac{1}{2}\sum_{e\in E}\left(\frac{w_e^+}{(s_e^+)^2} + \frac{w_e^-}{(s_e^-)^2}\right)(\widetilde{f}_e)^2\,,\end{aligned}$$

proving (3.9). Now the first order optimality condition of this problem is given by

$$\boldsymbol{C}^\top\left(\frac{\boldsymbol{w}^+}{(\boldsymbol{s}^+)^2} + \frac{\boldsymbol{w}^-}{(\boldsymbol{s}^-)^2}\right)\cdot(\boldsymbol{C}\widetilde{\boldsymbol{x}}) = \boldsymbol{C}^\top\boldsymbol{h}$$

or equivalently, by setting $\widetilde{\boldsymbol{f}} = \boldsymbol{C}\widetilde{\boldsymbol{x}}$,

$$\boldsymbol{C}^\top\left(\frac{\boldsymbol{w}^+}{(\boldsymbol{s}^+)^2} + \frac{\boldsymbol{w}^-}{(\boldsymbol{s}^-)^2}\right)\cdot\widetilde{\boldsymbol{f}} = \boldsymbol{C}^\top\boldsymbol{h}\,.$$

If $\widetilde{\boldsymbol{x}}, \widetilde{\boldsymbol{f}}$ are solutions to the above linear system, this implies

$$\begin{aligned}\mathcal{E}_{\boldsymbol{w},\boldsymbol{s}}(\boldsymbol{h}) &= \langle \boldsymbol{h}, \boldsymbol{C}\widetilde{\boldsymbol{x}}\rangle - \frac{1}{2}\sum_{e\in E}\left(\frac{w_e^+}{(s_e^+)^2} + \frac{w_e^-}{(s_e^-)^2}\right)(\boldsymbol{C}\widetilde{\boldsymbol{x}})_e^2 \\ &= \langle \boldsymbol{C}^\top\boldsymbol{h}, \widetilde{\boldsymbol{x}}\rangle - \frac{1}{2}\sum_{e\in E}\left(\frac{w_e^+}{(s_e^+)^2} + \frac{w_e^-}{(s_e^-)^2}\right)(\boldsymbol{C}\widetilde{\boldsymbol{x}})_e^2 \\ &= \frac{1}{2}\sum_{e\in E}\left(\frac{w_e^+}{(s_e^+)^2} + \frac{w_e^-}{(s_e^-)^2}\right)(\boldsymbol{C}\widetilde{\boldsymbol{x}})_e^2 \\ &= \frac{1}{2}\sum_{e\in E}\left(\frac{w_e^+}{(s_e^+)^2} + \frac{w_e^-}{(s_e^-)^2}\right)(\widetilde{f}_e)^2\,.\end{aligned}$$

The equations in terms of $\boldsymbol{\rho}$ follow by simple substitution. □

**Proof of Lemma 3.3.6**

*Proof.* We explicitly write the new residual after performing the update. We have:

$$\begin{aligned}-\boldsymbol{C}^\top\boldsymbol{h}' := \nabla F_\mu^{\boldsymbol{w}}(\boldsymbol{x}') &= \boldsymbol{C}^\top\left(\frac{\boldsymbol{w}^+}{(\boldsymbol{s}^+)'} - \frac{\boldsymbol{w}^-}{(\boldsymbol{s}^-)'} + \frac{\boldsymbol{c}}{\mu}\right) \\ &= \boldsymbol{C}^\top\left(\frac{\boldsymbol{w}^+}{\boldsymbol{s}^+} - \frac{\boldsymbol{w}^-}{\boldsymbol{s}^-} + \frac{\boldsymbol{c}}{\mu}\right) + \boldsymbol{C}^\top\left(\frac{\boldsymbol{w}^+}{(\boldsymbol{s}^+)'} - \frac{\boldsymbol{w}^+}{\boldsymbol{s}^+} - \frac{\boldsymbol{w}^-}{(\boldsymbol{s}^-)'} + \frac{\boldsymbol{w}^-}{\boldsymbol{s}^-}\right) \\ &\overset{(1)}{=} -\boldsymbol{C}^\top\left(\frac{\boldsymbol{w}^+\boldsymbol{\rho}^+}{\boldsymbol{s}^+} - \frac{\boldsymbol{w}^-\boldsymbol{\rho}^-}{\boldsymbol{s}^-}\right) + \boldsymbol{C}^\top\left(\frac{\boldsymbol{w}^+\boldsymbol{\rho}^+}{\boldsymbol{s}^+(\mathbf{1}-\boldsymbol{\rho}^+)} - \frac{\boldsymbol{w}^-\boldsymbol{\rho}^-}{\boldsymbol{s}^-(\mathbf{1}-\boldsymbol{\rho}^-)}\right) \\ &= \boldsymbol{C}^\top\left(\frac{\boldsymbol{w}^+(\boldsymbol{\rho}^+)^2}{\boldsymbol{s}^+(\mathbf{1}-\boldsymbol{\rho}^+)} - \frac{\boldsymbol{w}^-(\boldsymbol{\rho}^-)^2}{\boldsymbol{s}^-(\mathbf{1}-\boldsymbol{\rho}^-)}\right) \\ &= \boldsymbol{C}^\top\left(\frac{\boldsymbol{w}^+(\boldsymbol{\rho}^+)^2}{(\boldsymbol{s}^+)'} - \frac{\boldsymbol{w}^-(\boldsymbol{\rho}^-)^2}{(\boldsymbol{s}^-)'}\right),\end{aligned}$$

where (1) follows from Equations (3.11) and (3.12) and from writing

$$(\boldsymbol{s}^+)' = \boldsymbol{s}^+ - \boldsymbol{C}\widetilde{\boldsymbol{x}} = \boldsymbol{s}^+ \left(\mathbf{1} - \frac{\boldsymbol{C}\widetilde{\boldsymbol{x}}}{\boldsymbol{s}^+}\right) = \boldsymbol{s}\left(\mathbf{1} - \boldsymbol{\rho}^+\right)$$
$$(\boldsymbol{s}^-)' = \boldsymbol{s}^- + \boldsymbol{C}\widetilde{\boldsymbol{x}} = \boldsymbol{s}^- \left(\mathbf{1} + \frac{\boldsymbol{C}\widetilde{\boldsymbol{x}}}{\boldsymbol{s}^+}\right) = \boldsymbol{s}\left(\mathbf{1} - \boldsymbol{\rho}^-\right) .$$

Now we can upper bound the energy required to route the new residual with resistances determined by $(\boldsymbol{w}, \boldsymbol{s}')$, by substituting

$$\widetilde{\boldsymbol{y}} = \frac{\boldsymbol{w}^+(\rho^+)^2}{(\boldsymbol{s}^+)'} - \frac{\boldsymbol{w}^-(\rho^-)^2}{(\boldsymbol{s}^-)'}$$

into Definition 3.3.3, after noting that

$$\boldsymbol{C}^\top \widetilde{\boldsymbol{y}} = \boldsymbol{C}^\top \left(\frac{\boldsymbol{w}^+(\boldsymbol{\rho}^+)^2}{(\boldsymbol{s}^+)'} - \frac{\boldsymbol{w}^-(\boldsymbol{\rho}^-)^2}{(\boldsymbol{s}^-)'}\right) = -\boldsymbol{C}^\top \boldsymbol{h}' .$$

We thus obtain

$$\begin{aligned}\mathcal{E}_{\boldsymbol{w},\boldsymbol{s}'}(\boldsymbol{h}') &\leq \frac{1}{2}\sum_{e\in E} \frac{\left(\frac{w_e^+(\rho_e^+)^2}{(s_e^+)'} - \frac{w_e^-(\rho_e^-)^2}{(s_e^-)'}\right)^2}{\frac{w_e^+}{(s_e^+)'^2} + \frac{w_e^-}{(s_e^-)'^2}} \leq \frac{1}{2}\sum_{e\in E} \frac{\left(\frac{w_e^+(\rho_e^+)^2}{(s_e^+)'}\right)^2 + \left(\frac{w_e^-(\rho_e^-)^2}{(s_e^-)'}\right)^2}{\frac{w_e^+}{(s_e^+)'^2} + \frac{w_e^-}{(s_e^-)'^2}} \\ &\leq \frac{1}{2}\sum_{e\in E} \frac{\left(\frac{w_e^+(\rho_e^+)^2}{(s_e^+)'}\right)^2}{\frac{w_e^+}{(s_e^+)'^2}} + \frac{\left(\frac{w_e^-(\rho_e^-)^2}{(s_e^-)'}\right)^2}{\frac{w_e^-}{(s_e^-)'^2}} = \frac{1}{2}\sum_{e\in E}\left(w_e^+(\rho_e^+)^4 + w_e^-(\rho_e^-)^4\right) .\end{aligned}$$

□

**Proof of Corollary 3.3.7**

*Proof.* From Lemma 3.3.4 we have that

$$\mathcal{E}_{\boldsymbol{w},\boldsymbol{s}}(\boldsymbol{h}) = \frac{1}{2}\sum_{e\in E}\left(w_e^+(\rho_e^+)^2 + w_e^-(\rho_e^-)^2\right) .$$

Since $\boldsymbol{w} \geq 1$ we have that

$$\|\boldsymbol{\rho}\|_\infty^2 \leq 2 \cdot \mathcal{E}_{\boldsymbol{w},\boldsymbol{s}}(\boldsymbol{h}) .$$

Using Lemma 3.3.6 we upper bound

$$\mathcal{E}_{\boldsymbol{w},\boldsymbol{s}'}(\boldsymbol{h}') \leq \frac{1}{2}\sum_{e\in E}\left(w_e^+(\rho_e^+)^4 + w_e^-(\rho_e^-)^4\right) \leq \frac{1}{2}\|\boldsymbol{\rho}\|_\infty^2 \cdot \sum_{e\in E}\left(w_e^+(\rho_e^+)^2 + w_e^-(\rho_e^-)^2\right) \leq 2 \cdot \mathcal{E}_{\boldsymbol{w},\boldsymbol{s}}(\boldsymbol{h})^2 .$$

□

**Proof Lemma 3.3.8**

*Proof.* By Lemma 3.3.4 we know that there exists a vector $\boldsymbol{\rho}$ such that

$$\boldsymbol{C}^\top \boldsymbol{h} = \boldsymbol{C}^\top \left(\frac{\boldsymbol{w}^+\boldsymbol{\rho}^+}{\boldsymbol{s}^+} - \frac{\boldsymbol{w}^-\boldsymbol{\rho}^-}{\boldsymbol{s}^-}\right)$$

and

$$\mathcal{E}_{\boldsymbol{w},\boldsymbol{s}}(\boldsymbol{h}) = \frac{1}{2}\sum_{e\in E}\left(w_e^+(\rho_e^+)^2 + w_e^-(\rho_e^-)^2\right) \le \varepsilon\,.$$

Therefore $\|\boldsymbol{\rho}\|_\infty \le \sqrt{2\varepsilon}$. Now let $\boldsymbol{w}' = \frac{\boldsymbol{w}(\mathbf{1}+\boldsymbol{\rho})}{1-\|\boldsymbol{\rho}\|_\infty}$ and $\mu' = \frac{\mu}{1-\|\boldsymbol{\rho}\|_\infty}$. We have that

$$\begin{aligned}
\nabla F_{\mu'}^{\boldsymbol{w}'}(\boldsymbol{x}) &= \boldsymbol{C}^\top\left(\frac{(\boldsymbol{w}^+)'}{\boldsymbol{s}^+} - \frac{(\boldsymbol{w}^-)'}{\boldsymbol{s}^-} - \frac{\boldsymbol{c}}{\mu'}\right)\\
&= \frac{1}{1-\|\boldsymbol{\rho}\|_\infty}\cdot\left(\boldsymbol{C}^\top\left(\frac{\boldsymbol{w}^+}{\boldsymbol{s}^+} - \frac{\boldsymbol{w}^-}{\boldsymbol{s}^-} - \frac{\boldsymbol{c}}{\mu}\right) + \boldsymbol{C}^\top\left(\frac{\boldsymbol{w}^+\boldsymbol{\rho}^+}{\boldsymbol{s}^+} - \frac{\boldsymbol{w}^-\boldsymbol{\rho}^-}{\boldsymbol{s}^-}\right)\right)\\
&= \frac{1}{1-\|\boldsymbol{\rho}\|_\infty}\cdot\left(-\boldsymbol{C}^\top\boldsymbol{h} + \boldsymbol{C}^\top\boldsymbol{h}\right)\\
&= \mathbf{0}\,.
\end{aligned}$$

Furthermore, by this construction we see that $\boldsymbol{w} \le \boldsymbol{w}'$, and that

$$\boldsymbol{w}' \le \boldsymbol{w}\cdot\frac{1+\|\boldsymbol{\rho}\|_\infty}{1-\|\boldsymbol{\rho}\|_\infty} \le \boldsymbol{w}\cdot\frac{1+\sqrt{2\varepsilon}}{1-\sqrt{2\varepsilon}} \le \boldsymbol{w}\cdot(1+4\sqrt{\varepsilon})\,,$$

whenever $\varepsilon \le 1/100$. Therefore the total increase in weight is at most $4\sqrt{\varepsilon}\|\boldsymbol{w}\|_1$. Furthermore the loss of duality gap is determined by $\mu' \le \mu(1+2\sqrt{\varepsilon})$. □

**Proof of Lemma 3.3.9**

*Proof.* We first perform $O(\log\log\|\boldsymbol{w}\|_1)$ vanilla residual correction steps as described in Definition 3.3.5 to obtain a new solution $\boldsymbol{f}' = \boldsymbol{f}_0 + \boldsymbol{C}\boldsymbol{x}'$ with residual $\nabla F_\mu^{\boldsymbol{w}}(\boldsymbol{x}') = -\boldsymbol{C}^\top\boldsymbol{g}'$ and low energy $\mathcal{E}_{\boldsymbol{w},\boldsymbol{s}'}(\boldsymbol{g}') \le \|\boldsymbol{w}\|_1^{-22}/16$. Then we apply the perfect correction step from Lemma 3.3.8 to eliminate the residual $-\boldsymbol{C}^\top\boldsymbol{g}'$ by obtaining a new set of weights $\boldsymbol{w}' \ge \boldsymbol{w}$ such that $\|\boldsymbol{w}'-\boldsymbol{w}\|_1 \le \|\boldsymbol{w}\|^{-10}$) and $\nabla F_{\mu'}^{\boldsymbol{w}'}(\boldsymbol{x}') = \mathbf{0}$ where $\mu' \le \mu(1+\frac{1}{2}\|\boldsymbol{w}\|_1^{-11})$. □

**Proof of Lemma 3.3.10**

*Proof.* Since $\boldsymbol{f}$ is $\mu$-central, we can write:

$$\begin{aligned}
\nabla F_{\mu'}^{\boldsymbol{w}}(\boldsymbol{x}) &= \boldsymbol{C}^\top\left(\frac{\boldsymbol{w}^+}{\boldsymbol{s}^+} - \frac{\boldsymbol{w}^-}{\boldsymbol{s}^-} + \frac{\boldsymbol{c}}{\mu'}\right) = (1+\delta)\,\boldsymbol{C}^\top\left(\frac{\boldsymbol{w}^+}{\boldsymbol{s}^+} - \frac{\boldsymbol{w}^-}{\boldsymbol{s}^-} + \frac{\boldsymbol{c}}{\mu}\right) - \delta\,\boldsymbol{C}^\top\left(\frac{\boldsymbol{w}^+}{\boldsymbol{s}^+} - \frac{\boldsymbol{w}^-}{\boldsymbol{s}^-}\right)\\
&= -\delta\,\boldsymbol{C}^\top\left(\frac{\boldsymbol{w}^+}{\boldsymbol{s}^+} - \frac{\boldsymbol{w}^-}{\boldsymbol{s}^-}\right)\,,
\end{aligned}$$

and so we can set $\boldsymbol{h}' = \delta\left(\frac{\boldsymbol{w}^+}{\boldsymbol{s}^+} - \frac{\boldsymbol{w}^-}{\boldsymbol{s}^-}\right)$. Using Definition 3.3.3 we can upper bound the energy required to route this residual by exhibiting the solution $\widetilde{\boldsymbol{y}} = -\delta\left(\frac{\boldsymbol{w}^+}{\boldsymbol{s}^+} - \frac{\boldsymbol{w}^-}{\boldsymbol{s}^-}\right)$ after noting that

$\boldsymbol{C}^\top\left(\widetilde{\boldsymbol{y}}+\boldsymbol{h}'\right)=\mathbf{0}$:

$$\begin{aligned}\mathcal{E}_{\boldsymbol{w},\boldsymbol{s}}(\boldsymbol{h}') &\leq \frac{1}{2}\delta^2\sum_{e\in E}\frac{\left(\frac{w_e^+}{s_e^+}-\frac{w_e^-}{s_e^-}\right)^2}{\frac{w_e^+}{(s_e^+)^2}+\frac{w_e^-}{(s_e^-)^2}} \leq \frac{1}{2}\delta^2\sum_{e\in E}\frac{\left(\frac{w_e^+}{s_e^+}\right)^2+\left(\frac{w_e^-}{s_e^-}\right)^2}{\frac{w_e^+}{(s_e^+)^2}+\frac{w_e^-}{(s_e^-)^2}} \leq \frac{1}{2}\delta^2\left(\sum_{e\in E}\frac{\left(\frac{w_e^+}{s_e^+}\right)^2}{\frac{w_e^+}{(s_e^+)^2}}+\sum_{e\in E}\frac{\left(\frac{w_e^-}{s_e^-}\right)^2}{\frac{w_e^-}{(s_e^-)^2}}\right)\\ &= \frac{1}{2}\delta^2\left\|w\right\|_1,\end{aligned}$$

which is at most $1/4$ as long as $\delta\leq\frac{1}{(2\|\boldsymbol{w}\|_1)^{1/2}}$. □

**Proof of Lemma 3.3.11**

*Proof.* Given a $\mu$-central flow and using Lemma 3.3.10 we see that setting $\delta=\frac{1}{(2\|\boldsymbol{w}\|_1)^{1/2}}$ and $\mu'=\mu(1-\delta)$ we obtain $\mathcal{E}_{\boldsymbol{w},\boldsymbol{s}}(\boldsymbol{h})\leq 1/4$, where $-\boldsymbol{C}^\top\boldsymbol{h}=\nabla F^{\boldsymbol{w}}_{\mu'}(\boldsymbol{x})$. Hence applying Corollary 3.3.7 for $O(\log\log m)$ iterations we obtain a new flow $\boldsymbol{f}'=\boldsymbol{f}_0+\boldsymbol{C}\boldsymbol{x}'$ with slacks $\boldsymbol{s}'$ and residual $-\boldsymbol{C}^\top\boldsymbol{h}'=\nabla F^{\boldsymbol{w}}_{\mu'}(\boldsymbol{x}')$ such that $\mathcal{E}_{\boldsymbol{w},\boldsymbol{s}'}(\boldsymbol{h}')\leq m^{-20}/4$.

Finally, applying the perfect correction step from Lemma 3.3.8 we obtain a new set of weights $\boldsymbol{w}'\geq\boldsymbol{w}$, such that $\boldsymbol{w}'\leq\boldsymbol{w}(1+m^{-10})$ and $\nabla F^{\boldsymbol{w}'}_{\mu''}(\boldsymbol{x}')=\mathbf{0}$ for $\mu''\leq\mu'(1+m^{-10})$. In other words, $\boldsymbol{f}'$ is $\mu''$-central with respect to $\boldsymbol{w}'$.

Since the increase in weights is very small, iterating this procedure for $O(m^{1/2}\log m)$ steps maintains the invariant that $\|\boldsymbol{w}\|_1\leq 2m+1$. Furthermore, in each iteration the parameter $\mu$ gets scaled down by a factor of $1+\frac{1}{(4\|\boldsymbol{w}\|_1)^{1/2}}\geq 1+\frac{1}{3m^{1/2}}$, after which it gets slightly scaled up by at most $1+m^{-10}$ due to the perfect correction step. Hence within $O(m^{1/2}\log\frac{m\mu^0}{\varepsilon})=O(m^{1/2}\log m)$ iterations, the parameter $\mu$ gets scaled down by a factor of $\Omega(m\mu^0/\varepsilon)$, which thus implies that the final duality gap will be $O\left(\frac{3m\cdot\mu^0}{m\mu^0/\varepsilon}\right)=O(\varepsilon)$. □

**Proof of Lemma 3.4.5**

*Proof.* For the first inequality, we use the test vector $\widetilde{\boldsymbol{y}}=-\boldsymbol{h}$ into Definition 3.3.3. For the second one, we have

$$\begin{aligned}\mathcal{E}^{\max}(\boldsymbol{h},\boldsymbol{w},\boldsymbol{s}) &= \frac{1}{2}\sum_{e\in E}\frac{\delta^2\left(\frac{w_e^+}{s_e^+}-\frac{w_e^-}{s_e^-}\right)^2}{\frac{w_e^+}{(s_e^+)^2}+\frac{w_e^-}{(s_e^-)^2}} \leq \frac{1}{2}\delta^2\sum_{e\in E}\frac{\left(\frac{w_e^+}{s_e^+}\right)^2+\left(\frac{w_e^-}{s_e^-}\right)^2}{\frac{w_e^+}{(s_e^+)^2}+\frac{w_e^-}{(s_e^-)^2}}\\ &\leq \frac{1}{2}\delta^2\sum_{e\in E}\left(\frac{\left(\frac{w_e^+}{s_e^+}\right)^2}{\frac{w_e^+}{(s_e^+)^2}}+\frac{\left(\frac{w_e^-}{s_e^-}\right)^2}{\frac{w_e^-}{(s_e^-)^2}}\right)=\frac{1}{2}\delta^2\sum_{e\in E}(w_e^++w_e^-)\\ &= \frac{1}{2}\delta^2\left\|\boldsymbol{w}\right\|_1.\end{aligned}$$

□

**Proof of Corollary 3.4.10**

*Proof.* We can upper bound using Lemma 3.4.6:

$$\begin{aligned}\|\widetilde{\boldsymbol{f}}_\star\|_p &\le \left(\frac{p\cdot\mathcal{E}^{\max}(\boldsymbol{h},\boldsymbol{w},\boldsymbol{s})}{R_p}\right)^{1/p} \le \left(\frac{p\cdot\frac{1}{2}\delta^2\|\boldsymbol{w}\|_1}{p\cdot(10^6\cdot\delta^2\|\boldsymbol{w}\|_1\cdot\log\|\boldsymbol{w}\|_1)^{p+1}}\right)^{1/p}\\ &\le \frac{1}{10^6\cdot\delta^2\|\boldsymbol{w}\|_1\cdot\log\|\boldsymbol{w}\|_1}\,. \end{aligned} \tag{9.17}$$

□

**Proof of Corollary 3.4.11**

*Proof.* We have

$$\hat{\gamma} = \left(R_\star + R_p\cdot\|\widetilde{\boldsymbol{f}}_\star\|_\infty^{p-2}\right)^{1/2}\cdot\left\|\frac{\boldsymbol{h}}{\sqrt{(\boldsymbol{w}^+ + \boldsymbol{w}^-)\left(\frac{\boldsymbol{w}^+}{(\boldsymbol{s}^+)^2}+\frac{\boldsymbol{w}^-}{(\boldsymbol{s}^-)^2}\right)}}\right\|_\infty \cdot 32\log\|\boldsymbol{w}\|_1\,.$$

In particular, for sufficiently large $m$ and since $\|\widetilde{\boldsymbol{f}}_\star\|_\infty \le \|\widetilde{\boldsymbol{f}}_\star\|_p$ we have

$$\begin{aligned} R_p\left\|\widetilde{\boldsymbol{f}}_\star\right\|_\infty^{p-2} &\le \frac{p\left(10^6\cdot\delta^2\left\|\boldsymbol{w}\right\|_1\cdot\log\left\|\boldsymbol{w}\right\|_1\right)^{p+1}}{\left(10^6\cdot\delta^2\left\|\boldsymbol{w}\right\|_1\cdot\log\left\|\boldsymbol{w}\right\|_1\right)^{p-2}} \le p\cdot\left(10^6\cdot\delta^2\left\|\boldsymbol{w}\right\|_1\cdot\log\left\|\boldsymbol{w}\right\|_1\right)^3\\ &= \left(\delta^2\|\boldsymbol{w}\|_1\right)^3\cdot p\cdot\left(10^6\cdot\log\|\boldsymbol{w}\|_1\right)^3 < 3\delta^2\left\|\boldsymbol{w}\right\|_1^2 = R_\star\,, \end{aligned} \tag{9.18}$$

where the last inequality follows from the assumption on $\delta$. Furthermore, we bound the term under the $\ell_\infty$ norm as:

$$\begin{aligned}\frac{|h_e|}{\sqrt{(w_e^+ + w_e^-)\left(\frac{w_e^+}{(s_e^+)^2}+\frac{w_e^-}{(s_e^-)^2}\right)}} &= \delta\frac{\left|\frac{w_e^+}{s_e^+}-\frac{w_e^-}{s_e^-}\right|}{\sqrt{(w_e^+ + w_e^-)\left(\frac{w_e^+}{(s_e^+)^2}+\frac{w_e^-}{(s_e^-)^2}\right)}}\\ &= \delta\left(\frac{\left(\frac{w_e^+}{s_e^+}-\frac{w_e^-}{s_e^-}\right)^2}{(w_e^+ + w_e^-)\left(\frac{w_e^+}{(s_e^+)^2}+\frac{w_e^-}{(s_e^-)^2}\right)}\right)^{1/2}\\ &\le \delta\left(\frac{1}{w_e^+ + w_e^-}\left(\frac{(w_e^+/s_e^+)^2}{w_e^+/(s_e^+)^2}+\frac{(w_e^-/s_e^-)^2}{w_e^-/(s_e^-)^2}\right)\right)^{1/2}\\ &= \delta\,. \end{aligned} \tag{9.19}$$

Combining (9.18) and (9.19), we get the upper bound

$$\hat{\gamma} \le (2\cdot R_\star)^{1/2}\cdot\delta\cdot 32\log\|\boldsymbol{w}\|_1 = \delta^2\left\|\boldsymbol{w}\right\|_1\cdot 32\sqrt{6}\cdot\log\|\boldsymbol{w}\|_1 = \gamma\,.$$

□

**Proof of Lemma 3.4.15**

*Proof.* We restate (3.45) in terms of $\gamma \geq \hat{\gamma}$:

$$\left(\frac{w_e^+}{(s_e^+)^2} + \frac{w_e^-}{(s_e^-)^2}\right) \cdot \left|\widetilde{f}_e\right| \leq |h_e| + \gamma\,.$$

More specifically we will use the following, which the above implies for the setting of $\boldsymbol{h} = \delta\left(\frac{\boldsymbol{w}^+}{\boldsymbol{s}^+} - \frac{\boldsymbol{w}^-}{\boldsymbol{s}^-}\right)$:

$$\left(\frac{w_e^+}{(s_e^+)^2} + \frac{w_e^-}{(s_e^-)^2}\right) \cdot \left|\widetilde{f}_e\right| \leq \frac{w_e^+\delta}{s_e^+} + \frac{w_e^-\delta}{s_e^-} + \gamma$$

or equivalently since $\rho_e^+$ and $\rho_e^-$ have opposite signs:

$$\left|\frac{w_e^+\rho_e^+}{s_e^+}\right| + \left|\frac{w_e^-\rho_e^-}{s_e^-}\right| \leq \frac{w_e^+\delta}{s_e^+} + \frac{w_e^-\delta}{s_e^-} + \gamma\,. \tag{9.20}$$

Assume without loss of generality that $s_e^+ \leq s_e^-$, and so $|\rho_e^+| \geq |\rho_e^-|$. Now, for the sake of contradiction we suppose that $|\rho_e^+| \geq C_\infty$ and $\left|\frac{w_e^+\rho_e^+}{s_e^+}\right| + \left|\frac{w_e^-\rho_e^-}{s_e^-}\right| > 6\gamma$. We consider the two cases of Definition 3.4.13:

**(1)** $\max\{w_e^+, w_e^-\} \leq \delta\,\|\boldsymbol{w}\|_1$**:** Since $0 < s_e^+ \leq s_e^- < 1$ and $s_e^+ + s_e^- = 1$, we have that $s_e^- \geq \frac{1}{2}$. Therefore $\frac{w_e^-\delta}{s_e^-} \leq 2\cdot\delta^2\,\|\boldsymbol{w}\|_1 \leq 2\cdot\gamma$. We conclude that

$$\left|\frac{w_e^+\rho_e^+}{s_e^+}\right| + \left|\frac{w_e^-\rho_e^-}{s_e^-}\right| \leq \frac{w_e^+\delta}{s_e^+} + \frac{w_e^-\delta}{s_e^-} + \gamma \leq \frac{1}{2}\left|\frac{w_e^+\rho_e^+}{s_e^+}\right| + 3\gamma\,,$$

since $|\rho_e^+| \geq C_\infty = \frac{1}{2\delta\sqrt{2\|\boldsymbol{w}\|_1}} \geq 2\delta$ by our assumption on $\delta$. Thus

$$\left|\frac{w_e^+\rho_e^+}{s_e^+}\right| + \left|\frac{w_e^-\rho_e^-}{s_e^-}\right| \leq 6\gamma\,,$$

a contradiction.

**(2)** $\min\{w_e^+, w_e^-\} \geq 96\cdot\delta^4\,\|\boldsymbol{w}\|_1^2$**:** We will first prove that $\left|\widetilde{f}_e\right| < 2\delta$. Suppose to the contrary. We have that $|\rho_e^-| = \frac{|\widetilde{f}_e|}{s_e^-} \geq \left|\widetilde{f}_e\right| \geq 2\delta$, so

$$\left|\frac{w_e^+\rho_e^+}{s_e^+}\right| + \left|\frac{w_e^-\rho_e^-}{s_e^-}\right| \leq \frac{w_e^+\delta}{s_e^+} + \frac{w_e^-\delta}{s_e^-} + \gamma \leq \frac{1}{2}\left|\frac{w_e^+\rho_e^+}{s_e^+}\right| + \frac{1}{2}\left|\frac{w_e^-\rho_e^-}{s_e^-}\right| + \gamma$$

and thus

$$\left|\frac{w_e^+\rho_e^+}{s_e^+}\right| + \left|\frac{w_e^-\rho_e^-}{s_e^-}\right| \leq 2\gamma\,,$$

a contradiction, therefore $\left|\widetilde{f}_e\right| < 2\delta$. This immediately implies that $2\delta > \left|\widetilde{f}_e\right| = |\rho_e^+|\, s_e^+ \geq C_\infty s_e^+$, and so $s_e^+ \leq 2\delta/C_\infty$. Therefore we get

$$\left|\frac{w_e^+\rho_e^+}{s_e^+}\right| \geq \frac{(96\cdot\delta^4\left\|\boldsymbol{w}\right\|_1^2)C_\infty}{2\delta/C_\infty} = 48\cdot\delta^3\left\|\boldsymbol{w}\right\|_1^2\cdot C_\infty^2\,.$$

Since $C_\infty^2 = \frac{1}{8\delta^2\|\boldsymbol{w}\|_1}$, we conclude that

$$\left|\frac{w_e^+\rho_e^+}{s_e^+}\right| \geq 6\cdot\delta\left\|\boldsymbol{w}\right\|_1 \geq 3\frac{w_e^-\delta}{s_e^-}\,,$$

where we used the fact that $\|w\|_\infty \leq \|w\|_1$ and the fact that $s_e^- \geq \frac{1}{2}$. Therefore

$$\left|\frac{w_e^+\rho_e^+}{s_e^+}\right| + \left|\frac{w_e^-\rho_e^-}{s_e^-}\right| \leq \frac{w_e^+\delta}{s_e^+} + \frac{w_e^-\delta}{s_e^-} + \gamma \leq \frac{1}{2}\left|\frac{w_e^+\rho_e^+}{s_e^+}\right| + \frac{1}{3}\left|\frac{w_e^+\rho_e^+}{s_e^+}\right| + \gamma\,.$$

Therefore we conclude that

$$\left|\frac{w_e^+\rho_e^+}{s_e^+}\right| + \left|\frac{w_e^-\rho_e^-}{s_e^-}\right| \leq 6\gamma\,,$$

again a contradiction.

□

**Proof of Lemma 3.4.18**

*Proof.* Let us analyze the new residual after performing the update described in (3.47-3.48). Just like in the standard correction step which we analyzed in Section 3.3.4, here we can write:

$$\begin{aligned}
\nabla F_\mu^{\boldsymbol{w}}(\boldsymbol{x}') &= \frac{\boldsymbol{C}^\top\boldsymbol{c}}{\mu} + \boldsymbol{C}^\top\left(\frac{\boldsymbol{w}^+}{(\boldsymbol{s}^+)'} - \frac{\boldsymbol{w}^-}{(\boldsymbol{s}^-)'}\right)\\
&= \nabla F_\mu^{\boldsymbol{w}}(\boldsymbol{x}) + \boldsymbol{C}^\top\left(\frac{\boldsymbol{w}^+}{(\boldsymbol{s}^+)'} - \frac{\boldsymbol{w}^-}{(\boldsymbol{s}^-)'}\right) - \boldsymbol{C}^\top\left(\frac{\boldsymbol{w}^+}{\boldsymbol{s}^+} - \frac{\boldsymbol{w}^-}{\boldsymbol{s}^-}\right)\\
&= \boldsymbol{C}^\top\Delta\boldsymbol{h} - \boldsymbol{C}^\top\left(\frac{\boldsymbol{w}^+\boldsymbol{\rho}^+}{\boldsymbol{s}^+} - \frac{\boldsymbol{w}^-\boldsymbol{\rho}^-}{\boldsymbol{s}^-}\right) + \boldsymbol{C}^\top\left(\frac{\boldsymbol{w}^+}{(\boldsymbol{s}^+)'} - \frac{\boldsymbol{w}^-}{(\boldsymbol{s}^-)'}\right) - \boldsymbol{C}^\top\left(\frac{\boldsymbol{w}^+}{\boldsymbol{s}^+} - \frac{\boldsymbol{w}^-}{\boldsymbol{s}^-}\right)\\
&= \boldsymbol{C}^\top\Delta\boldsymbol{h} - \boldsymbol{C}^\top\left(\frac{\boldsymbol{w}^+(\mathbf{1}+\boldsymbol{\rho}^+)}{\boldsymbol{s}^+} - \frac{\boldsymbol{w}^-(\mathbf{1}+\boldsymbol{\rho}^-)}{\boldsymbol{s}^-}\right) + \boldsymbol{C}^\top\left(\frac{\boldsymbol{w}^+}{(\boldsymbol{s}^+)'} - \frac{\boldsymbol{w}^-}{(\boldsymbol{s}^-)'}\right)\\
&= \boldsymbol{C}^\top\Delta\boldsymbol{h} - \boldsymbol{C}^\top\left(\frac{\boldsymbol{w}^+(\mathbf{1}+\boldsymbol{\rho}^+)(\mathbf{1}-\boldsymbol{\rho}^+)}{(\boldsymbol{s}^+)'} - \frac{\boldsymbol{w}^-(\mathbf{1}+\boldsymbol{\rho}^-)(\mathbf{1}-\boldsymbol{\rho}^-)}{(\boldsymbol{s}^-)'}\right) + \boldsymbol{C}^\top\left(\frac{\boldsymbol{w}^+}{(\boldsymbol{s}^+)'} - \frac{\boldsymbol{w}^-}{(\boldsymbol{s}^-)'}\right)\\
&= \boldsymbol{C}^\top\Delta\boldsymbol{h} + \boldsymbol{C}^\top\left(\frac{\boldsymbol{w}^+(\boldsymbol{\rho}^+)^2}{(\boldsymbol{s}^+)'} - \frac{\boldsymbol{w}^-(\boldsymbol{\rho}^-)^2}{(\boldsymbol{s}^-)'}\right)\,.
\end{aligned}$$

Therefore

$$\nabla F_\mu^{\boldsymbol{w}}(\boldsymbol{x}') - \boldsymbol{C}^\top\Delta\boldsymbol{h} = \boldsymbol{C}^\top\left(\frac{\boldsymbol{w}^+(\boldsymbol{\rho}^+)^2}{(\boldsymbol{s}^+)'} - \frac{\boldsymbol{w}^-(\boldsymbol{\rho}^-)^2}{(\boldsymbol{s}^-)'}\right)\,.$$

Next we show that modifying the weights from $\boldsymbol{w}$ to $\boldsymbol{w}'$ sets a subset of these entries to 0, which will enable us to correct this new perturbed residual. Using the weight update described in (3.53-3.54) we obtain

$$
\begin{aligned}
&- \boldsymbol{C}^\top (\boldsymbol{g} + \Delta \boldsymbol{h}) \\
&= \nabla F_\mu^{\boldsymbol{w}'}(\boldsymbol{x}') - \boldsymbol{C}^\top \Delta \boldsymbol{h} \\
&= \boldsymbol{C}^\top \frac{\boldsymbol{c}}{\mu} - \boldsymbol{C}^\top \Delta \boldsymbol{h} + \boldsymbol{C}^\top \left( \frac{(\boldsymbol{w}^+)'}{(\boldsymbol{s}^+)'} - \frac{(\boldsymbol{w}^-)'}{(\boldsymbol{s}^-)'} \right) \\
&= \nabla F_\mu^{\boldsymbol{w}}(\boldsymbol{x}') - \boldsymbol{C}^\top \Delta \boldsymbol{h} + \boldsymbol{C}^\top \left( \frac{(\boldsymbol{w}^+)' - \boldsymbol{w}^+}{(\boldsymbol{s}^+)'} - \frac{(\boldsymbol{w}^-)' - \boldsymbol{w}^-}{(\boldsymbol{s}^-)'} \right) \\
&= \nabla F_\mu^{\boldsymbol{w}}(\boldsymbol{x}') - \boldsymbol{C}^\top \Delta \boldsymbol{h} + \boldsymbol{C}^\top \left( \frac{\boldsymbol{w}^- (\boldsymbol{\rho}^-)^2}{(\boldsymbol{s}^-)'} \cdot \mathbf{1}_{|\boldsymbol{\rho}^-| \geq C_\infty} - \frac{\boldsymbol{w}^+ (\boldsymbol{\rho}^+)^2}{(\boldsymbol{s}^+)'} \cdot \mathbf{1}_{|\boldsymbol{\rho}^+| \geq C_\infty} \right) \\
&= \boldsymbol{C}^\top \left( \left( \frac{\boldsymbol{w}^+ (\boldsymbol{\rho}^+)^2}{(\boldsymbol{s}^+)'} \cdot \mathbf{1}_{|\boldsymbol{\rho}^+| < C_\infty} \right) - \left( \frac{\boldsymbol{w}^- (\boldsymbol{\rho}^-)^2}{(\boldsymbol{s}^-)'} \cdot \mathbf{1}_{|\boldsymbol{\rho}^-| < C_\infty} \right) \right) .
\end{aligned}
$$

Finally, using Lemma 3.3.4 we certify an upper bound on

$$
\begin{aligned}
\mathcal{E}_{\boldsymbol{w}', \boldsymbol{s}'}(\boldsymbol{g} + \Delta \boldsymbol{h}) &\leq \mathcal{E}^{\max}(\boldsymbol{g} + \Delta \boldsymbol{h}, \boldsymbol{w}', \boldsymbol{s}') \\
&= \frac{1}{2} \sum_{e \in E} \frac{\left( \left( \frac{\boldsymbol{w}^+ (\boldsymbol{\rho}^+)^2}{(\boldsymbol{s}^+)'} \cdot \mathbf{1}_{|\boldsymbol{\rho}^+| < C_\infty} \right) - \left( \frac{\boldsymbol{w}^- (\boldsymbol{\rho}^-)^2}{(\boldsymbol{s}^-)'} \cdot \mathbf{1}_{|\boldsymbol{\rho}^-| < C_\infty} \right) \right)_e^2}{\frac{w_e^{'+}}{(s_e^+)'^2} + \frac{w_e^{'-}}{(s_e^-)'^2}} \\
&\leq \frac{1}{2} \sum_{\substack{e \in E \\ |\rho_e^+| < C_\infty}} \frac{\left( \frac{w_e^+ (\rho_e^+)^2}{(s_e^+)'} \right)^2}{\frac{w_e^{'+}}{(s_e^+)'^2}} + \frac{1}{2} \sum_{\substack{e \in E \\ |\rho_e^-| < C_\infty}} \frac{\left( \frac{w_e^- (\rho_e^-)^2}{(s_e^-)'} \right)^2}{\frac{w_e^{'-}}{(s_e^-)'^2}} \\
&\leq \frac{1}{2} \sum_{\substack{e \in E \\ |\rho_e^+| < C_\infty}} w_e^+ (\rho_e^+)^4 + \frac{1}{2} \sum_{\substack{e \in E \\ |\rho_e^-| < C_\infty}} w_e^- (\rho_e^-)^4 \\
&\leq \frac{1}{2} C_\infty^2 \cdot \sum_{e \in E} \left( w_e^+ (\rho_e^+)^2 + w_e^- (\rho_e^-)^2 \right) \\
&= \frac{1}{2} C_\infty^2 \cdot \sum_{e \in E} (\widetilde{f}_e)^2 \left( \frac{w_e^+}{(s_e^+)^2} + \frac{w_e^-}{(s_e^-)^2} \right) \\
&\leq \frac{1}{2} C_\infty^2 \cdot 8 \cdot \mathcal{E}^{\max}(\boldsymbol{h}, \boldsymbol{w}, \boldsymbol{s}) \\
&\leq \frac{2\delta^2 \|\boldsymbol{w}\|_1}{8\delta^2 \|\boldsymbol{w}\|_1} \\
&= \frac{1}{4} ,
\end{aligned}
$$

where we used the fact that $\boldsymbol{w}' \geq \boldsymbol{w}$, (3.24), and Lemma 3.4.5. □

**Proof of Lemma 3.6.3**

*Proof.* Let $\widetilde{\boldsymbol{f}}_\star = \widetilde{\boldsymbol{f}} + \widetilde{\boldsymbol{f}}'$ where $\widetilde{\boldsymbol{f}}'$ is the restriction of $\widetilde{\boldsymbol{f}}_\star$ to the edges incident to $v_\star$. To prove (3.65), note that from Lemma 3.6.2 there exists a vector $\boldsymbol{\alpha} = (\boldsymbol{\alpha}^+; \boldsymbol{\alpha}^-)$, $(1+\theta)^{-2} \cdot \mathbf{1} \leq \boldsymbol{\alpha} \leq (1-\theta)^{-2} \cdot \mathbf{1}$,

such that for any circulation $\boldsymbol{g} = \boldsymbol{C}_\star \boldsymbol{z}_\star$ in $G_\star$,

$$\left\langle \boldsymbol{g}, \begin{bmatrix} \boldsymbol{h} - \widetilde{\boldsymbol{f}} \left( \frac{\boldsymbol{\alpha}^+ \boldsymbol{w}^+}{(\boldsymbol{s}^+)^2} + \frac{\boldsymbol{\alpha}^- \boldsymbol{w}^-}{(\boldsymbol{s}^-)^2} \right) - R_p \cdot (\widetilde{\boldsymbol{f}})^{p-1} \\ -R_\star \cdot \widetilde{\boldsymbol{f}}' - R_p \cdot (\widetilde{\boldsymbol{f}}')^{p-1} \end{bmatrix} \right\rangle = 0\,. \tag{9.21}$$

Restricting ourselves to circulations supported only in the non-preconditioned graph $G$, one has that for any circulation in $\boldsymbol{g}' = \boldsymbol{C}\boldsymbol{z}$ in $G$:

$$\left\langle \boldsymbol{z}, \boldsymbol{C}^\top \left( \boldsymbol{h} + \Delta \boldsymbol{h} - \widetilde{\boldsymbol{f}} \left( \frac{\boldsymbol{\alpha}^+ \boldsymbol{w}^+}{(\boldsymbol{s}^+)^2} + \frac{\boldsymbol{\alpha}^- \boldsymbol{w}^-}{(\boldsymbol{s}^-)^2} \right) \right) \right\rangle = 0$$

Since this holds for any test vector $\boldsymbol{z}$, it must be that the second term in the inner product is $\mathbf{0}$. Rearranging, it yields the identity from (3.65). Next, we notice that (9.21) is the optimality condition of the following objective:

$$\begin{aligned} \max_{\widetilde{\boldsymbol{f}}_\star = \boldsymbol{C}_\star \widetilde{\boldsymbol{x}}} \left\langle \boldsymbol{h}, \widetilde{\boldsymbol{f}} \right\rangle - \frac{1}{2} \sum_{e \in E} (\widetilde{f}_e)^2 \cdot \left( \frac{\alpha_e^+ w_e^+}{(s_e^+)^2} + \frac{\alpha_e^- w_e^-}{(s_e^-)^2} \right) \\ - \frac{R_\star}{2} \sum_{e \in E'} (\widetilde{f}'_e)^2 - \frac{R_p}{p} \sum_{e \in E \cup E'} (\widetilde{f}_\star)_e^p\,. \end{aligned} \tag{9.22}$$

Let us proceed to bound the norm of the demand routed by $\widetilde{\boldsymbol{f}}$. Consider the value of the objective in (9.22) after truncating it to only the first two terms, which we can write as:

$$\left\langle \boldsymbol{h}, \widetilde{\boldsymbol{f}} \right\rangle - \frac{1}{2} \sum_{e \in E} (\widetilde{f}_e)^2 \cdot \left( \frac{\alpha_e^+ w_e^+}{(s_e^+)^2} + \frac{\alpha_e^- w_e^-}{(s_e^-)^2} \right) \tag{9.23}$$

$$\leq \sum_{e \in E} h_e \cdot \widetilde{f}_e - \frac{1}{3} \sum_{e \in E} (\widetilde{f}_e)^2 \cdot \left( \frac{w_e^+}{(s_e^+)^2} + \frac{w_e^-}{(s_e^-)^2} \right) \tag{9.24}$$

$$\leq \frac{3}{4} \sum_{e \in E} h_e^2 \cdot \left( \frac{w_e^+}{(s_e^+)^2} + \frac{w_e^-}{(s_e^-)^2} \right)^{-1} \tag{9.25}$$

$$= \frac{3}{2} \mathcal{E}^{\max}(\boldsymbol{h}, \boldsymbol{w}, \boldsymbol{s})\,, \tag{9.26}$$

where we used $\boldsymbol{\alpha} \geq \frac{1}{(1+\theta)^2} \cdot \mathbf{1} > \frac{2}{3} \cdot \mathbf{1}$ and the fact that $\langle \boldsymbol{a}, \boldsymbol{b} \rangle \leq \frac{1}{2}\|\boldsymbol{a}\|^2 + \frac{1}{2}\|\boldsymbol{b}\|^2$.

Note that the value of the regularized objective (9.22) is at least 0 since we can always substitute $\widetilde{\boldsymbol{x}} = \mathbf{0}$ and obtain exactly 0. By re-arranging,

$$\frac{R_\star}{2} \sum_{e \in E'} (\widetilde{f}'_e)^2 \tag{9.27}$$

$$\leq \left\langle \boldsymbol{h}, \widetilde{\boldsymbol{f}} \right\rangle - \frac{1}{2} \sum_{e \in E} (\widetilde{f}_e)^2 \cdot \left( \frac{\alpha_e^+ w_e^+}{(s_e^+)^2} + \frac{\alpha_e^- w_e^-}{(s_e^-)^2} \right) - \frac{R_p}{p} \sum_{e \in E \cup E'} (\widetilde{f}_\star)_e^p \tag{9.28}$$

$$\leq \left\langle \boldsymbol{h}, \widetilde{\boldsymbol{f}} \right\rangle - \frac{1}{2} \sum_{e \in E} (\widetilde{f}_e)^2 \cdot \left( \frac{\alpha_e^+ w_e^+}{(s_e^+)^2} + \frac{\alpha_e^- w_e^-}{(s_e^-)^2} \right) \tag{9.29}$$

$$\leq \frac{3}{2} \mathcal{E}^{\max}(\boldsymbol{h}, \boldsymbol{w}, \boldsymbol{s})\,, \tag{9.30}$$

where we also used the fact that the last term of (9.28) is non-positive and (9.26). Therefore (9.30) enables us to upper bound

$$\sum_{e\in E'}(\widetilde{f}'_e)^2 \leq \frac{3}{R_\star}\mathcal{E}^{\max}(\boldsymbol{h},\boldsymbol{w},\boldsymbol{s})\,, \tag{9.31}$$

which implies that

$$\sum_{e\in E'}\left|\widetilde{f}'_e\right| \leq |E'|^{1/2}\cdot\left(\sum_{e\in E'}(\widetilde{f}'_e)^2\right)^{1/2} \tag{9.32}$$

$$\leq \left(3\|\boldsymbol{w}\|_1\cdot\frac{3}{R_\star}\mathcal{E}^{\max}(\boldsymbol{h},\boldsymbol{w},\boldsymbol{s})\right)^{1/2} \tag{9.33}$$

$$= 3\left(\frac{\|\boldsymbol{w}\|_1\cdot\mathcal{E}^{\max}(\boldsymbol{h},\boldsymbol{w},\boldsymbol{s})}{R_\star}\right)^{1/2}, \tag{9.34}$$

a quantity that upper bounds the demand perturbation. Using a similar argument we can upper bound $\left\|\widetilde{\boldsymbol{f}}_\star\right\|_p$. We have

$$\frac{R_p}{p}\sum_{e\in E\cup E'}(\widetilde{f}_\star)_e^p \tag{9.35}$$

$$\leq \left\langle \boldsymbol{h},\widetilde{\boldsymbol{f}}\right\rangle - \frac{1}{2}\sum_{e\in E}(\widetilde{f}_e)^2\cdot\left(\frac{\alpha_e^+w_e^+}{(s_e^+)^2}+\frac{\alpha_e^-w_e^-}{(s_e^-)^2}\right) - \frac{R_\star}{2}\sum_{e\in E'}(\widetilde{f}'_e)^2 \tag{9.36}$$

$$\leq \left\langle \boldsymbol{h},\widetilde{\boldsymbol{f}}\right\rangle - \frac{1}{2}\sum_{e\in E}(\widetilde{f}_e)^2\cdot\left(\frac{\alpha_e^+w_e^+}{(s_e^+)^2}+\frac{\alpha_e^-w_e^-}{(s_e^-)^2}\right) \tag{9.37}$$

$$\leq \frac{3}{2}\mathcal{E}^{\max}(\boldsymbol{h},\boldsymbol{w},\boldsymbol{s})\,, \tag{9.38}$$

thus concluding that

$$\left\|\widetilde{\boldsymbol{f}}_\star\right\|_p = \left(\sum_{e\in E\cup E'}(\widetilde{f}_\star)_e^p\right)^{1/p} \leq \left(\frac{p\cdot\frac{3}{2}\mathcal{E}^{\max}(\boldsymbol{h},\boldsymbol{w},\boldsymbol{s})}{R_p}\right)^{1/p}. \tag{9.39}$$

□

## 9.2 Appendix for Chapter 4

### 9.2.1 Maintaining the Schur Complement

Following the scheme from [73] we maintain a dynamic Schur complement of the graph onto a subset of terminals $C$. The approach follows rather directly from [73] and leverages the recent work of [18] to dynamically maintain an edge sparsifier of the Schur complement of the graph onto $C$. Compared to [73] we do not require a parameter that depends on the adaptivity of the adversary. In addition, when adding a vertex to $C$ we also return a $(1+\varepsilon)$-approximation of the effective resistance $R_{eff}(v,C)$, which gets returned by the function call.

**Lemma 9.2.1** (DynamicSC (Theorem 4, [73])). *There is a* DynamicSC *data structure supporting*

*the following operations with the given runtimes against oblivious adversaries, for constants* $0 < \beta, \varepsilon < 1$*:*

- INITIALIZE$(G, C^{(init)}, \boldsymbol{r}, \varepsilon, \beta)$*: Initializes a graph* $G$ *with resistances* $\boldsymbol{r}$ *and a set of safe terminals* $C^{(safe)}$*. Sets the terminal set* $C = C^{(safe)} \cup C^{(init)}$*. Runtime:* $\widetilde{O}\left(m\beta^{-4}\varepsilon^{-4}\right)$*.*
- ADDTERMINAL$(v \in V(G))$*: Returns* $\widetilde{R}_{eff}(C, v) \approx_2 R_{eff}(C, v)$ *and adds* $v$ *as a terminal. Runtime: Amortized* $\widetilde{O}\left(\beta^{-2}\varepsilon^{-2}\right)$*.*
- TEMPORARYADDTERMINALS$(\Delta C \subseteq V(G))$*: Adds all vertices in the set* $\Delta C$ *as (temporary) terminals. Runtime: Worst case* $\widetilde{O}\left(K^2\beta^{-4}\varepsilon^{-4}\right)$*, where* $K$ *is the total number of terminals added by all of the* TEMPORARYADDTERMINALS *operations that have not been rolled back using* ROLLBACK*. All* TEMPORARYADDTERMINALS *operations should be rolled back before the next call to* ADDTERMINALS*.*
- UPDATE$(e, r)$*: Under the guarantee that both endpoints of* $e$ *are terminals, updates* $r_e = r$*. Runtime: Worst case* $\widetilde{O}(1)$*.*
- $\widetilde{SC}()$*: Returns a spectral sparsifier* $\widetilde{SC} \approx_{1+\varepsilon} SC(G, C)$ *(with respect to resistances* $\boldsymbol{r}$*) with* $\widetilde{O}\left(|C|\varepsilon^{-2}\right)$ *edges. Runtime: Worst case* $\widetilde{O}\left(\left(\beta m + (K\beta^{-2}\varepsilon^{-2})^2\right)\varepsilon^{-2}\right)$ *where* $K$ *is the total number of terminals added by all of the* TEMPORARYADDTERMINALS *operations that have not been rolled back.*
- ROLLBACK()*: Rolls back the last* UPDATE*,* ADDTERMINALS*, or* TEMPORARYADDTERMINALS *if it exists. The runtime is the same as the original operation.*

*Finally, all calls return valid outputs with high probability. The size of* $C$ *should always be* $O(\beta m)$*.*

This data structure is analyzed in detail in [73]. Additionally, let us show that an approximation to $R_{eff}(v, C)$ can be efficiently computed along with the ADDTERMINAL operation. To get an estimate we simply inspect the neighbors of $v$ in the sparsified Schur complement of $C \cup \{v\}$ and compute the inverse of the sum of their inverses. This is indeed a $1 + O(\varepsilon)$-approximation, as effective resistances are preserved within a $1 + O(\varepsilon)$ factor in the sparsifier.

To show that this operation takes little amortized time, we note that by the proof appearing in [73, Lemma 6.2], vertex $v$ appears in amortized $\widetilde{O}(1)$ expanders maintained dynamically. As the dynamic sparsifier keeps $\widetilde{O}\left(\varepsilon^{-2}\right)$ neighbors of $v$ from each expander, the number of neighbors to inspect with each call is $\widetilde{O}\left(\varepsilon^{-2}\right)$, which also bounds the time necessary to approximate the resistance.

### 9.2.2 Auxiliary Lemmas

**Lemma 4.2.6.** *Let* $\boldsymbol{d}$ *be a demand vector, let* $\boldsymbol{r}$ *be resistances, and let* $C \subseteq V$ *be a subset of vertices. Then*

$$\mathcal{E}_{\boldsymbol{r}}\left(\boldsymbol{\pi}^C(\boldsymbol{d})\right) \leq \mathcal{E}_{\boldsymbol{r}}\left(\boldsymbol{d}\right).$$

*Proof.* Letting $F = V \setminus C$, and $\boldsymbol{L}$ be the Laplacian of the underlying graph, we can write

$$\boldsymbol{\pi}^C(\boldsymbol{d}) = \boldsymbol{d}_C - \boldsymbol{L}_{CF}\boldsymbol{L}_{FF}^{-1}\boldsymbol{d}_F.$$

By factoring $\boldsymbol{L}^+$ as

$$\boldsymbol{L}^+ = \begin{bmatrix} I & 0 \\ -\boldsymbol{L}_{FF}^{-1}\boldsymbol{L}_{FC} & I \end{bmatrix} \begin{bmatrix} SC(\boldsymbol{L}, C)^+ & 0 \\ 0 & \boldsymbol{L}_{FF}^{-1} \end{bmatrix} \begin{bmatrix} I & -\boldsymbol{L}_{CF}\boldsymbol{L}_{FF}^{-1} \\ 0 & I \end{bmatrix}$$

we can write

$$\mathcal{E}_{\boldsymbol{r}}(\boldsymbol{d}) = \boldsymbol{d}^\top \boldsymbol{L}^+ \boldsymbol{d} = \begin{bmatrix} \boldsymbol{\pi}^C(\boldsymbol{d}) \\ \boldsymbol{d}_F \end{bmatrix}^\top \begin{bmatrix} SC(\boldsymbol{L}, C)^+ & 0 \\ 0 & \boldsymbol{L}_{FF}^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\pi}^C(\boldsymbol{d}) \\ \boldsymbol{d}_F \end{bmatrix} = \|\boldsymbol{\pi}^C(\boldsymbol{d})\|^2_{SC(\boldsymbol{L},C)^+} + \|\boldsymbol{d}_F\|^2_{\boldsymbol{L}_{FF}^{-1}}.$$

Furthermore, we can use the same factorization to write

$$\mathcal{E}_{\boldsymbol{r}}(\boldsymbol{\pi}^C(\boldsymbol{d})) = \begin{bmatrix} \boldsymbol{\pi}^C(\boldsymbol{d}) \\ 0 \end{bmatrix}^\top \begin{bmatrix} SC(\boldsymbol{L}, C)^+ & 0 \\ 0 & \boldsymbol{L}_{FF}^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\pi}^C(\boldsymbol{d}) \\ 0 \end{bmatrix} = \|\boldsymbol{\pi}^C(\boldsymbol{d})\|^2_{SC(\boldsymbol{L},C)^+},$$

which proves the claim. □

**Lemma 9.2.2.** *For any $\mu \in (1/\mathrm{poly}(m), \mathrm{poly}(m))$, we have $\|\boldsymbol{r}(\mu)\|_\infty \leq m^{\widetilde{O}(\log m)}$.*

*Proof.* By Appendix A in [10], for some $\mu_0 = \Theta(\|\boldsymbol{c}\|_2)$, the solution $\boldsymbol{f} = \boldsymbol{u}/2$ has

$$\left\| \boldsymbol{C}^\top \left( \frac{\boldsymbol{c}}{\mu_0} + \frac{\mathbf{1}}{\boldsymbol{s}^+} - \frac{\mathbf{1}}{\boldsymbol{s}^-} \right) \right\|_{(\boldsymbol{C}^\top \boldsymbol{R} \boldsymbol{C})^+} \leq 1/10\,.$$

This implies that $\min_e \{s_e(\mu_0)^+, s_e(\mu_0)^-\} \geq \min_e u_e/4 \geq 1/4$, and so $\|\boldsymbol{r}(\mu_0)\|_\infty \leq O(1)$. Additinally, $\|\boldsymbol{c}\|_\infty \in [1, \mathrm{poly}(m)]$, so $\mu_0 = \Theta(\mathrm{poly}(m))$.

Now, for any integer $i \geq 0$ we let $\mu_{i+1} = \mu_i \cdot (1 - 1/\sqrt{m})^{\sqrt{m}/10}$. By Lemma 9.2.7 we have that $\boldsymbol{r}(\mu_{i+1}) \approx_{m^2} \boldsymbol{r}(\mu_i)$, and so

$$\boldsymbol{r}\left(\frac{1}{\mathrm{poly}(m)}\right) = \boldsymbol{r}(\mu_{\widetilde{O}(\log m)}) \leq \left(\frac{9}{100} m^2\right)^{\widetilde{O}(\log m)} \boldsymbol{r}(\mu_0) \leq m^{\widetilde{O}(\log m)} \boldsymbol{r}(\mu_0) \leq m^{\widetilde{O}(\log m)}.$$

□

**Lemma 9.2.3.** *Given a graph $G(V, E)$ with resistances $\boldsymbol{r}$ and any parameter $\varepsilon > 0$, there exists an algorithm that runs in time $\widetilde{O}\left(m/\varepsilon^2\right)$ and produces a matrix $\boldsymbol{Q} \in \mathbb{R}^{\widetilde{O}\left(1/\varepsilon^2\right) \times n}$ such that with high probability for any $u, v \in V$,*

$$R_{eff}(u, v) \approx_{1+\varepsilon} \|\boldsymbol{Q}\mathbf{1}_u - \boldsymbol{Q}\mathbf{1}_v\|_2^2$$

### 9.2.3 Deferred Proofs from Section 4.3

#### Central path stability bounds

**Lemma 9.2.4** (Central path energy stability)**.** *Consider a minimum cost flow instance on a graph $G(V, E)$. For any $\mu > 0$ and $\mu' = \mu/(1 + 1/\sqrt{m})^k$ for some $k \in (0, \sqrt{m}/10)$, we have*

$$\sum_{e \in E} \left( \frac{1}{s_e(\mu)^+ \cdot s_e(\mu')^+} + \frac{1}{s_e(\mu)^- \cdot s_e(\mu')^-} \right) \left(f_e(\mu') - f_e(\mu)\right)^2 \leq 2k^2$$

*Proof of Lemma 9.2.4.* We let $\delta = 1/\sqrt{m}$, $\boldsymbol{f} = \boldsymbol{f}(\mu)$, $\boldsymbol{s} = \boldsymbol{s}(\mu)$, $\boldsymbol{r} = \boldsymbol{r}(\mu)$, $\boldsymbol{f}' = \boldsymbol{f}(\mu')$, $\boldsymbol{s}' = \boldsymbol{s}(\mu')$,

and $\boldsymbol{r}' = \boldsymbol{r}(\mu')$. We also set $\widetilde{\boldsymbol{f}} = \boldsymbol{f}' - \boldsymbol{f}$. By definition of centrality we have

$$\boldsymbol{C}^\top \left( \frac{1}{\boldsymbol{s}^-} - \frac{1}{\boldsymbol{s}^+} \right) = \boldsymbol{C}^\top \frac{\boldsymbol{c}}{\mu}$$
$$\boldsymbol{C}^\top \left( \frac{1}{\boldsymbol{s}^- + \widetilde{\boldsymbol{f}}} - \frac{1}{\boldsymbol{s}^+ - \widetilde{\boldsymbol{f}}} \right) = \boldsymbol{C}^\top \frac{\boldsymbol{c}}{\mu'},$$

which, after subtracting, give

$$\boldsymbol{C}^\top \left( \frac{1}{\boldsymbol{s}^- + \widetilde{\boldsymbol{f}}} - \frac{1}{\boldsymbol{s}^-} - \frac{1}{\boldsymbol{s}^+ - \widetilde{\boldsymbol{f}}} + \frac{1}{\boldsymbol{s}^+} \right) = \boldsymbol{C}^\top \left( \frac{\boldsymbol{c}}{\mu'} - \frac{\boldsymbol{c}}{\mu} \right)$$
$$\Leftrightarrow \boldsymbol{C}^\top \left( \left( \frac{1}{\boldsymbol{s}^-(\boldsymbol{s}^- + \widetilde{\boldsymbol{f}})} + \frac{1}{\boldsymbol{s}^+(\boldsymbol{s}^+ - \widetilde{\boldsymbol{f}})} \right) \widetilde{\boldsymbol{f}} \right) = - \left( (1+\delta)^k - 1 \right) \boldsymbol{C}^\top \frac{\boldsymbol{c}}{\mu}.$$

As $\widetilde{\boldsymbol{f}} = \boldsymbol{C}\boldsymbol{x}$ for some $\boldsymbol{x}$, after taking the inner product of both sides with $\boldsymbol{x}$ we get

$$\left\langle \widetilde{\boldsymbol{f}}, \left( \frac{1}{\boldsymbol{s}^-(\boldsymbol{s}^- + \widetilde{\boldsymbol{f}})} + \frac{1}{\boldsymbol{s}^+(\boldsymbol{s}^+ - \widetilde{\boldsymbol{f}})} \right) \widetilde{\boldsymbol{f}} \right\rangle = - \left( (1+\delta)^k - 1 \right) \left\langle \frac{\boldsymbol{c}}{\mu}, \widetilde{\boldsymbol{f}} \right\rangle . \tag{9.40}$$

We will now prove that $-\left\langle \frac{\boldsymbol{c}}{\mu}, \widetilde{\boldsymbol{f}} \right\rangle \leq k\sqrt{m}$. First of all, by differentiating the centrality condition

$$\boldsymbol{C}^\top \left( \frac{\boldsymbol{c}}{\nu} + \frac{\mathbf{1}}{\boldsymbol{s}(\nu)^+} - \frac{\mathbf{1}}{\boldsymbol{s}(\nu)^-} \right) = \mathbf{0}$$

with respect to $\nu$ we get

$$\boldsymbol{C}^\top \left( -\frac{\boldsymbol{c}}{\nu^2} + \left( \frac{\mathbf{1}}{(\boldsymbol{s}(\nu)^+)^2} + \frac{\mathbf{1}}{(\boldsymbol{s}(\nu)^-)^2} \right) \frac{d\boldsymbol{f}(\nu)}{d\nu} \right) = \mathbf{0},$$

or equivalently

$$\boldsymbol{C}^\top \left( \boldsymbol{r}(\nu) \frac{d\boldsymbol{f}(\nu)}{d\nu} \right) = -\frac{1}{\nu} \boldsymbol{C}^\top \left( \frac{\mathbf{1}}{\boldsymbol{s}(\nu)^+} - \frac{\mathbf{1}}{\boldsymbol{s}(\nu)^-} \right).$$

If we set $g(\boldsymbol{s}) = \frac{\frac{\mathbf{1}}{\boldsymbol{s}^+} - \frac{\mathbf{1}}{\boldsymbol{s}^-}}{\boldsymbol{r}}$, this can also be equivalently written as

$$\frac{d\boldsymbol{f}(\nu)}{d\nu} = -\frac{1}{\nu} \left( g(\boldsymbol{s}(\nu)) - (\boldsymbol{R}(\nu))^{-1} \boldsymbol{B} (\boldsymbol{B}^\top (\boldsymbol{R}(\nu)^{-1}) \boldsymbol{B})^+ \boldsymbol{B}^\top g(\boldsymbol{s}(\nu)) \right).$$

We have

$$\begin{aligned}
-\left\langle \frac{\boldsymbol{c}}{\mu}, \widetilde{\boldsymbol{f}} \right\rangle &= -\int_{\nu=\mu}^{\mu'} \left\langle \frac{\boldsymbol{c}}{\mu}, d\boldsymbol{f}(\nu) \right\rangle \\
&= \frac{1}{\mu} \int_{\nu=\mu}^{\mu'} \left\langle \frac{\nu}{\boldsymbol{s}(\nu)^-} - \frac{\nu}{\boldsymbol{s}(\nu)^+}, \frac{1}{\nu} \left( g(\boldsymbol{s}(\nu)) - (\boldsymbol{R}(\nu))^{-1} \boldsymbol{B} (\boldsymbol{B}^\top (\boldsymbol{R}(\nu))^{-1} \boldsymbol{B})^+ \boldsymbol{B}^\top g(\boldsymbol{s}(\nu)) \right) \right\rangle d\nu \\
&= -\frac{1}{\mu} \int_{\nu=\mu}^{\mu'} \left\langle \sqrt{\boldsymbol{r}(\nu)} g(\boldsymbol{s}(\nu)), \boldsymbol{\Pi}_{\ker(\boldsymbol{B}^\top (\boldsymbol{R}(\nu))^{-1/2})} \sqrt{\boldsymbol{r}(\nu)} g(\boldsymbol{s}(\nu)) \right\rangle d\nu \\
&= \frac{1}{\mu} \int_{\nu=\mu'}^{\mu} \left\| \boldsymbol{\Pi}_{\ker(\boldsymbol{B}^\top (\boldsymbol{R}(\nu))^{-1/2})} \sqrt{\boldsymbol{r}(\nu)} g(\boldsymbol{s}(\nu)) \right\|_2^2 d\nu \\
&\leq \frac{1}{\mu} \int_{\nu=\mu'}^{\mu} \left\| \sqrt{\boldsymbol{r}(\nu)} g(\boldsymbol{s}(\nu)) \right\|_2^2 d\nu \\
&\leq \frac{1}{\mu} \int_{\nu=\mu'}^{\mu} m d\nu \\
&= m \frac{\mu - \mu'}{\mu} \\
&= m(1 - (1+\delta)^{-k}) \\
&\leq \delta k m \\
&= k\sqrt{m}\,,
\end{aligned}$$

where $\boldsymbol{\Pi}_{\ker(\boldsymbol{B}^\top (\boldsymbol{R}(\nu))^{-1/2})} = \boldsymbol{I} - (\boldsymbol{R}(\nu))^{-1/2} \boldsymbol{B} (\boldsymbol{B}^\top (\boldsymbol{R}(\nu))^{-1} \boldsymbol{B})^+ \boldsymbol{B}^\top (\boldsymbol{R}(\nu))^{-1/2}$ is the orthogonal projection onto the kernel of $\boldsymbol{B}^\top (\boldsymbol{R}(\nu))^{-1/2}$.

Plugging this into (9.40) and using the fact that $(1+\delta)^k \leq 1 + 1.1\delta k = 1 + 1.1k/\sqrt{m}$, we get

$$\sum_{e \in E} \left( \frac{1}{s_e(\mu)^+ \cdot s_e(\mu')^+} + \frac{1}{s_e(\mu)^- \cdot s_e(\mu')^-} \right) \left( f_e(\mu') - f_e(\mu) \right)^2 \leq 2k^2\,.$$

□

We give an auxiliary lemma which converts between different kinds of slack approximations.

**Lemma 9.2.5.** *We consider flows $\boldsymbol{f}, \boldsymbol{f}'$ with slacks $\boldsymbol{s}, \boldsymbol{s}'$ and resistances $\boldsymbol{r}, \boldsymbol{r}'$. Then,*

$$\max\left\{ \left| \frac{s_e'^+ - s_e^+}{s_e^+} \right|, \left| \frac{s_e'^- - s_e^-}{s_e^-} \right| \right\} \leq \sqrt{r_e}\,|f_e' - f_e| \leq \sqrt{2} \max\left\{ \left| \frac{s_e'^+ - s_e^+}{s_e^+} \right|, \left| \frac{s_e'^- - s_e^-}{s_e^-} \right| \right\}$$

*and if $r_e \not\approx_{1+\gamma} r_e'$ for some $\gamma \in (0,1)$, then $\sqrt{r_e}\,|f_e' - f_e| \geq \gamma/6$.*

*Proof.* For the first one, note that

$$r_e = \frac{1}{(s_e^+)^2} + \frac{1}{(s_e^-)^2} \in \left[ \max\left\{ \frac{1}{(s_e^+)^2}, \frac{1}{(s_e^-)^2} \right\}, 2\max\left\{ \frac{1}{(s_e^+)^2}, \frac{1}{(s_e^-)^2} \right\} \right].$$

Together with the fact that $|f_e' - f_e| = |s_e'^+ - s_e^+| = |s_e'^- - s_e^-|$, it implies the first statement.

For the second one, without loss of generality let $s_e^+ \leq s_e^-$, so by the previous statement we have $\sqrt{r_e}\,|f_e' - f_e| \geq \frac{|s_e'^+ - s_e^+|}{s_e^+}$. If this is $< \gamma/6$ then $(1 - \gamma/6) s_e^+ \leq s_e'^+ \leq (1 + \gamma/6) s_e^+$, so

$s_e'^{+} \approx_{1+\gamma/3} s_e^{+}$. However, we also have that $\frac{|s_e'^{-} - s_e^{-}|}{s_e^{-}} \leq \frac{|s_e'^{+} - s_e^{+}|}{s_e^{+}} \leq \gamma/6$, so $s_e'^{-} \approx_{1+\gamma/3} s_e^{-}$. Therefore, $r_e' = \frac{1}{(s_e'^{+})^2} + \frac{1}{(s_e'^{-})^2} \approx_{1+\gamma} \frac{1}{(s_e^{+})^2} + \frac{1}{(s_e^{-})^2} = r_e$, a contradiction. □

The following lemma is a fine-grained explanation of how resistances can change.

**Lemma 9.2.6.** *Consider a minimum cost flow instance on a graph $G(V,E)$ and parameters $\mu > 0$ and $\mu' \geq \mu/(1+1/\sqrt{m})^k$, where $k \in (0, \sqrt{m}/10)$. For any $e \in E$ and $\gamma \in (0,1)$ we let* $\text{change}(e,\gamma)$ *be the largest integer $t(e) \geq 0$ such that there are real numbers $\mu = \mu_1(e) > \mu_2(e) > \cdots > \mu_{t(e)+1}(e) = \mu'$ with $\sqrt{r_e(\mu_i)}\,|f_e(\mu_{i+1}) - f_e(\mu_i)| \geq \gamma$ for all $i \in [t(e)]$.*

*Then,* $\sum_{e \in E} (\text{change}(e,\gamma))^2 \leq O(k^2/\gamma^2)$.

*Proof of Lemma 9.2.6.* First, we assume that without loss of generality, $r_e(\mu_{i+1}(e)) \approx_{(1+6\gamma)^2} r_e(\mu_i(e))$ for all $e \in E$ and $i \in [t(e)]$. If this is not true, then by continuity there exists a $\nu \in (\mu_{i+1}, \mu_i)$ such that $r_e(\mu_{i+1}(e)) \not\approx_{1+6\gamma} r_e(\nu)$ and $r_e(\nu) \not\approx_{1+6\gamma} r_e(\mu_i(e))$. By Lemma 9.2.5, this implies that $\sqrt{r_e(\mu_{i+1}(e))}\,|f_e(\nu) - f_e(\mu_{i+1}(e))| \geq \gamma$ and $\sqrt{r_e(\nu)}\,|f_e(\mu_i(e)) - f_e(\nu)| \geq \gamma$. Therefore we can break the interval $(\mu_{i+1}, \mu_i)$ into $(\mu_{i+1}, \nu)$ and $(\nu, \mu_i)$ and make the statement stronger.

Similarly, we also assume that $r_e(\nu) \approx_{(1+6\gamma)^3} r_e(\mu_i(e))$ for all $e \in E$, $i \in [t(e)]$, and $\nu \in (\mu_{i+1}, \mu_i)$. If this is not the case, then by using the fact that $r_e(\mu_{i+1}(e)) \approx_{(1+6\gamma)^2} r_e(\mu_i(e))$, we also get that $r_e(\mu_{i+1}) \not\approx_{1+6\gamma} r_e(\nu)$, and so we can again break the interval as before and obtain a stronger statement.

Now, we look at the following integral:

$$\mathcal{E} := \int_{\nu=\mu}^{\mu'} \sum_{e \in E} r_e(\nu) \left(\frac{df_e(\nu)}{d\nu}\right)^2 |d\nu| \,,$$

where $df_e(\nu)$ is the differential of the flow $f_e(\nu)$ with respect to the centrality parameter. Similarly to Lemma 9.2.4, we use the following equation that describes how the flow changes:

$$\frac{d\boldsymbol{f}(\nu)}{d\nu} = -\frac{1}{\nu}\left(g(\boldsymbol{s}(\nu)) - (\boldsymbol{R}(\nu))^{-1}\boldsymbol{B}(\boldsymbol{B}^\top(\boldsymbol{R}(\nu)^{-1})\boldsymbol{B})^{+}\boldsymbol{B}^\top g(\boldsymbol{s}(\nu))\right).$$

This implies that

$$\begin{aligned}
\left\|\sqrt{\boldsymbol{r}(\nu)}\frac{d\boldsymbol{f}(\nu)}{d\nu}\right\|_2^2 &= \frac{1}{\nu^2}\left\|\sqrt{\boldsymbol{r}(\nu)}g(\boldsymbol{s}(\nu)) - (\boldsymbol{R}(\nu))^{-1/2}\boldsymbol{B}(\boldsymbol{B}^\top(\boldsymbol{R}(\nu)^{-1})\boldsymbol{B})^{+}\boldsymbol{B}^\top g(\boldsymbol{s}(\nu))\right\|_2^2 \\
&\leq \frac{1}{\nu^2}\left\|\left(I - (\boldsymbol{R}(\nu))^{-1/2}\boldsymbol{B}(\boldsymbol{B}^\top(\boldsymbol{R}(\nu)^{-1})\boldsymbol{B})^{+}\boldsymbol{B}^\top(\boldsymbol{R}(\nu))^{-1/2}\right)\sqrt{\boldsymbol{r}}g(\boldsymbol{s}(\nu))\right\|_2^2 \\
&\leq \frac{1}{\nu^2}\left\|\sqrt{\boldsymbol{r}}g(\boldsymbol{s}(\nu))\right\|_2^2 \\
&\leq \frac{m}{\nu^2}\,,
\end{aligned}$$

and so

$$\mathcal{E} \leq \int_{\nu=\mu}^{\mu'} \frac{m}{\nu^2}\,|d\nu| = m\left(\frac{1}{\mu'} - \frac{1}{\mu}\right) \leq m\frac{1.1\delta k}{\mu} = 1.1k\sqrt{m}/\mu\,. \tag{9.41}$$

On the other hand, for any $e \in E$ and $i \in [t(e)]$ we have

$$\begin{aligned}\int_{\nu=\mu_i(e)}^{\mu_{i+1}(e)} r_e(\nu)\left(\frac{df_e(\nu)}{d\nu}\right)^2 |d\nu| &\geq \frac{r_e(\mu_i(e))}{(1+6\gamma)^3}\int_{\nu=\mu_i(e)}^{\mu_{i+1}(e)}\left(\frac{df_e(\nu)}{d\nu}\right)^2|d\nu| \\ &\geq \frac{r_e(\mu_i(e))}{(1+6\gamma)^3}\frac{\left(\int_{\nu=\mu_i(e)}^{\mu_{i+1}(e)}\left|\frac{df_e(\nu)}{d\nu}\right||d\nu|\right)^2}{\int_{\nu=\mu_i(e)}^{\mu_{i+1}(e)}|d\nu|} \\ &= \frac{r_e(\mu_i(e))}{(1+6\gamma)^3(\mu_i(e)-\mu_{i+1}(e))}\left(f(\mu_i(e))-f(\mu_{i+1}(e))\right)^2 \\ &\geq \frac{\gamma^2}{36(1+6\gamma)^3(\mu_i(e)-\mu_{i+1}(e))}\,,\end{aligned}$$

where we used the Cauchy-Schwarz inequality. Now, note that

$$\begin{aligned}\int_{\nu=\mu_1(e)}^{\mu_{t(e)+1}(e)} r_e(\nu)\left(\frac{df_e(\nu)}{d\nu}\right)^2|d\nu| &\geq \sum_{i=1}^{t(e)}\frac{\gamma^2}{(1+6\gamma)^3(\mu_i(e)-\mu_{i+1}(e))} \\ &\geq \frac{\gamma^2(t(e))^2}{(1+6\gamma)^3(\mu-\mu')} \\ &\geq \frac{\gamma^2(t(e))^2\sqrt{m}}{(1+6\gamma)^3 k\mu}\,,\end{aligned}$$

where remember that $t(e) = \text{change}(e,\gamma)$ and we again used Cauchy-Schwarz. Summing this up for all $e \in E$ and combining with (9.41), we get that $\sum_{e\in E}(\text{change}(e,\gamma))^2 \leq O(k^2/\gamma^2)$. □

**Lemma 9.2.7** (Central path $\ell_\infty$ slack stability)**.** *Consider a minimum cost flow instance on a graph $G(V,E)$. For any $\mu > 0$ and $\mu' = \mu/(1+1/\sqrt{m})^k$ for some $k \in (0, \sqrt{m}/10)$, we have*

$$\boldsymbol{s}(\mu') \approx_{3k^2} \boldsymbol{s}(\mu)\,.$$

*Proof of Lemma 9.2.7.* By Lemma 9.2.4, for any $e \in E$ we have that

$$\left(\frac{1}{s_e(\mu)^+ \cdot s_e(\mu')^+} + \frac{1}{s_e(\mu)^- \cdot s_e(\mu')^-}\right)\left(f_e(\mu') - f_e(\mu)\right)^2 \leq 2k^2\,. \tag{9.42}$$

If $s_e(\mu')^+ = (1+c)\cdot s_e(\mu)^+$ for some $c \geq 0$, then

$$(f_e(\mu') - f_e(\mu))^2 = c^2(s_e(\mu)^+)^2$$

and

$$s_e(\mu)^+ \cdot s_e(\mu')^+ = (1+c)(s_e(\mu)^+)^2\,,$$

so by (9.42) we have that $c \leq 3k^2$.

Similarly, if $s_e(\mu')^+ = (1+c)^{-1}\cdot s_e(\mu)^+$ for some $c \geq 0$, then

$$(f_e(\mu') - f_e(\mu))^2 = c^2(s_e(\mu')^+)^2$$

and

$$s_e(\mu)^+ \cdot s_e(\mu')^+ = (1+c)(s_e(\mu')^+)^2\,,$$

so by (9.42) we have that $c \le 3k^2$.

We have proved that $s_e(\mu')^+ \approx_{3k^2} s_e(\mu)^+$ and by symmetry we also have $s_e(\mu')^- \approx_{3k^2} s_e(\mu)^-$. □

**Proof of Lemma 4.3.5**

Our goal is to keep track of how close $\boldsymbol{f}^*$ remains to centrality (in $\ell_2$ norm) and how close $\boldsymbol{f}$ remains to $\boldsymbol{f}^*$ in $\ell_\infty$ norm. From these two we can conclude that at all times $\boldsymbol{f}$ is close in $\ell_\infty$ to the central flow. We first prove the following lemma, which bounds how the distance of $\boldsymbol{f}^*$ to centrality (measured in energy of the residual) degrades when taking a progress step.

**Lemma 9.2.8.** *Let $\boldsymbol{f}^*$ be a flow with slacks $\boldsymbol{s}^*$ and resistances $\boldsymbol{r}^*$, and $\boldsymbol{f}$ be a flow with slacks $\boldsymbol{s}$ and resistances $\boldsymbol{r}$, where $\boldsymbol{s} \approx_{1+\varepsilon_{\text{solve}}} \boldsymbol{s}^*$ for some $\varepsilon_{\text{solve}} \in (0, 0.1)$. We define $\boldsymbol{f}'^* = \boldsymbol{f}^* + \varepsilon_{\text{step}}\widetilde{\boldsymbol{f}}^*$ for some $\varepsilon_{\text{step}} \in (0, 0.1)$ (and the new slacks $\boldsymbol{s}'^*$), where*

$$\widetilde{\boldsymbol{f}}^* = \delta g(\boldsymbol{s}) - \delta \boldsymbol{R}^{-1}\boldsymbol{B}(\boldsymbol{B}^\top \boldsymbol{R}^{-1}\boldsymbol{B})^+ \boldsymbol{B}^\top g(\boldsymbol{s})\,, \tag{9.43}$$

*$\delta = \frac{1}{\sqrt{m}}$, and $g(\boldsymbol{s}) := \frac{\frac{1}{\boldsymbol{s}^+} - \frac{1}{\boldsymbol{s}^-}}{\boldsymbol{r}}$. If we let $\boldsymbol{h} = \frac{\boldsymbol{c}}{\mu} + \frac{1}{\boldsymbol{s}^{*+}} - \frac{1}{\boldsymbol{s}^{*-}}$ and $\boldsymbol{h}' = \frac{\boldsymbol{c}(1+\varepsilon_{\text{step}}\delta)}{\mu} + \frac{1}{\boldsymbol{s}'^{*+}} - \frac{1}{\boldsymbol{s}'^{*-}}$ be the residuals of $\boldsymbol{f}^*$ and $\boldsymbol{f}'^*$ for some $\mu > 0$, then*

$$\left\|\boldsymbol{C}^\top \boldsymbol{h}'\right\|_{\overline{\boldsymbol{H}}^+} \le (1+\varepsilon_{\text{step}}\delta)\left\|\boldsymbol{C}^\top \boldsymbol{h}\right\|_{\overline{\boldsymbol{H}}^+} + 5\left\|\frac{\boldsymbol{r}^*}{\bar{\boldsymbol{r}}}\right\|_\infty^{1/2} \varepsilon_{\text{solve}} \cdot \varepsilon_{\text{step}} + 2\left\|\frac{\boldsymbol{r}'^*}{\bar{\boldsymbol{r}}}\right\|_\infty^{1/2} \varepsilon_{\text{step}}^2\,,$$

*where $\bar{\boldsymbol{r}}$ are some arbitrary resistances and $\overline{\boldsymbol{H}} = \boldsymbol{C}^\top \overline{\boldsymbol{R}}\boldsymbol{C}$.*

*Proof.* Let $\boldsymbol{\rho}^+ = \varepsilon_{\text{step}}\widetilde{\boldsymbol{f}}^*/\boldsymbol{s}^{*+}$ and $\boldsymbol{\rho}^- = -\varepsilon_{\text{step}}\widetilde{\boldsymbol{f}}^*/\boldsymbol{s}^{*-}$. First of all, it is easy to see that

$$\begin{aligned}
\|\boldsymbol{\rho}\|_2 &\le \left\|\frac{\boldsymbol{s}}{\boldsymbol{s}^*}\right\|_\infty \left\|\frac{\boldsymbol{s}^*}{\boldsymbol{s}}\boldsymbol{\rho}\right\|_2 \\
&\le \varepsilon_{\text{step}}(1+\varepsilon_{\text{solve}})\left\|\widetilde{\boldsymbol{f}}^*\right\|_{\boldsymbol{r},2} \\
&= \varepsilon_{\text{step}}\delta(1+\varepsilon_{\text{solve}})\left\|\sqrt{\boldsymbol{r}}g(\boldsymbol{s}) - \boldsymbol{R}^{-1/2}\boldsymbol{B}(\boldsymbol{B}^\top \boldsymbol{R}^{-1}\boldsymbol{B})^+\boldsymbol{B}^\top g(\boldsymbol{s})\right\|_2 \\
&= \varepsilon_{\text{step}}\delta(1+\varepsilon_{\text{solve}})\left\|\left(\boldsymbol{I} - \boldsymbol{R}^{-1/2}\boldsymbol{B}(\boldsymbol{B}^\top \boldsymbol{R}^{-1}\boldsymbol{B})^+\boldsymbol{B}^\top \boldsymbol{R}^{-1/2}\right)\sqrt{\boldsymbol{r}}g(\boldsymbol{s})\right\|_2 \\
&\le \varepsilon_{\text{step}}\delta(1+\varepsilon_{\text{solve}})\left\|\sqrt{\boldsymbol{r}}g(\boldsymbol{s})\right\|_2 \\
&= \varepsilon_{\text{step}}\delta(1+\varepsilon_{\text{solve}})\left\|\frac{\frac{1}{\boldsymbol{s}^+} - \frac{1}{\boldsymbol{s}^-}}{\sqrt{\frac{1}{(\boldsymbol{s}^+)^2} + \frac{1}{(\boldsymbol{s}^-)^2}}}\right\|_2 \\
&\le \varepsilon_{\text{step}}\delta(1+\varepsilon_{\text{solve}})\sqrt{m} \\
&= \varepsilon_{\text{step}}(1+\varepsilon_{\text{solve}})\,.
\end{aligned}$$

We bound the energy to route the residual of $\boldsymbol{f}'^*$ as

$$\begin{aligned}
&\left\|\boldsymbol{C}^\top \boldsymbol{h}'\right\|_{\overline{\boldsymbol{H}}^+} \\
&= \left\|\boldsymbol{C}^\top \boldsymbol{h} + \boldsymbol{C}^\top \left(\frac{\varepsilon_{\text{step}}\delta \boldsymbol{c}}{\mu} + \frac{1}{\boldsymbol{s}'^{*+}} - \frac{1}{\boldsymbol{s}^{*+}} - \frac{1}{\boldsymbol{s}'^{*-}} + \frac{1}{\boldsymbol{s}^{*-}}\right)\right\|_{\overline{\boldsymbol{H}}^+} \\
&= \left\|\boldsymbol{C}^\top \boldsymbol{h} + \boldsymbol{C}^\top \left(\frac{\varepsilon_{\text{step}}\delta \boldsymbol{c}}{\mu} + \frac{\boldsymbol{\rho}^+}{\boldsymbol{s}'^{*+}} - \frac{\boldsymbol{\rho}^-}{\boldsymbol{s}'^{*-}}\right)\right\|_{\overline{\boldsymbol{H}}^+} \\
&= \left\|\boldsymbol{C}^\top \boldsymbol{h} + \boldsymbol{C}^\top \left(\frac{\varepsilon_{\text{step}}\delta \boldsymbol{c}}{\mu} + \frac{\boldsymbol{\rho}^+}{\boldsymbol{s}^{*+}} - \frac{\boldsymbol{\rho}^-}{\boldsymbol{s}^{*-}}\right) + \boldsymbol{C}^\top \left(\frac{(\boldsymbol{\rho}^+)^2}{\boldsymbol{s}'^{*+}} - \frac{(\boldsymbol{\rho}^-)^2}{\boldsymbol{s}'^{*-}}\right)\right\|_{\overline{\boldsymbol{H}}^+}.
\end{aligned}$$

Now, using (9.43) we get that $\boldsymbol{r}\widetilde{\boldsymbol{f}}^* = \delta \boldsymbol{r} g(\boldsymbol{s}) - \delta \boldsymbol{B}(\boldsymbol{B}^\top \boldsymbol{R}^{-1}\boldsymbol{B})^+ \boldsymbol{B}^\top g(\boldsymbol{s})$ and so $\boldsymbol{C}^\top\left(\boldsymbol{r}\widetilde{\boldsymbol{f}}^*\right) = \delta \boldsymbol{C}^\top(\boldsymbol{r} g(\boldsymbol{s}))$, which follows by the fact that for any $i$, $\mathbf{1}_i^\top \boldsymbol{C}^\top \boldsymbol{B} = \left(\boldsymbol{B}^\top \boldsymbol{C}\mathbf{1}_i\right)^\top = \mathbf{0}$, since $\boldsymbol{C}\mathbf{1}_i$ is a circulation by definition of $\boldsymbol{C}$. As $\boldsymbol{r}\widetilde{\boldsymbol{f}}^* = \left(\frac{1}{(\boldsymbol{s}^+)^2} + \frac{1}{(\boldsymbol{s}^-)^2}\right)\widetilde{\boldsymbol{f}}^* = \varepsilon_{\text{step}}^{-1}\frac{\boldsymbol{s}^{*+}}{(\boldsymbol{s}^+)^2}\boldsymbol{\rho}^+ - \varepsilon_{\text{step}}^{-1}\frac{\boldsymbol{s}^{*-}}{(\boldsymbol{s}^-)^2}\boldsymbol{\rho}^-$, we have $\varepsilon_{\text{step}}\delta \boldsymbol{C}^\top(\boldsymbol{r} g(\boldsymbol{s})) = \boldsymbol{C}^\top\left(\frac{\boldsymbol{s}^{*+}}{(\boldsymbol{s}^+)^2}\boldsymbol{\rho}^+ - \frac{\boldsymbol{s}^{*-}}{(\boldsymbol{s}^-)^2}\boldsymbol{\rho}^-\right)$ and so

$$\begin{aligned}
&\left\|\boldsymbol{C}^\top \boldsymbol{h} + \boldsymbol{C}^\top \left(\frac{\varepsilon_{\text{step}}\delta \boldsymbol{c}}{\mu} + \frac{\boldsymbol{\rho}^+}{\boldsymbol{s}^{*+}} - \frac{\boldsymbol{\rho}^-}{\boldsymbol{s}^{*-}}\right) + \boldsymbol{C}^\top \left(\frac{(\boldsymbol{\rho}^+)^2}{\boldsymbol{s}'^{*+}} - \frac{(\boldsymbol{\rho}^-)^2}{\boldsymbol{s}'^{*-}}\right)\right\|_{\overline{\boldsymbol{H}}^+} \\
&= \left\|\boldsymbol{C}^\top \boldsymbol{h} + \boldsymbol{C}^\top \left(\frac{\varepsilon_{\text{step}}\delta \boldsymbol{c}}{\mu} + \varepsilon_{\text{step}}\delta \boldsymbol{r} g(\boldsymbol{s}) - \frac{\boldsymbol{s}^{*+}}{(\boldsymbol{s}^+)^2}\boldsymbol{\rho}^+ + \frac{\boldsymbol{s}^{*-}}{(\boldsymbol{s}^-)^2}\boldsymbol{\rho}^- + \frac{\boldsymbol{\rho}^+}{\boldsymbol{s}^{*+}} - \frac{\boldsymbol{\rho}^-}{\boldsymbol{s}^{*-}}\right) + \boldsymbol{C}^\top \left(\frac{(\boldsymbol{\rho}^+)^2}{\boldsymbol{s}'^{*+}} - \frac{(\boldsymbol{\rho}^-)^2}{\boldsymbol{s}'^{*-}}\right)\right\|_{\overline{\boldsymbol{H}}^+} \\
&= \left\|\boldsymbol{C}^\top \boldsymbol{h} + \varepsilon_{\text{step}}\delta \boldsymbol{C}^\top\left(\frac{\boldsymbol{c}}{\mu} + \boldsymbol{r} g(\boldsymbol{s})\right) + \boldsymbol{C}^\top\left(\left(\mathbf{1} - \left(\frac{\boldsymbol{s}^{*+}}{\boldsymbol{s}^+}\right)^2\right)\frac{\boldsymbol{\rho}^+}{\boldsymbol{s}^{*+}} - \left(\mathbf{1} - \left(\frac{\boldsymbol{s}^{*-}}{\boldsymbol{s}^-}\right)^2\right)\frac{\boldsymbol{\rho}^-}{\boldsymbol{s}^{*-}}\right)\right. \\
&\left. + \boldsymbol{C}^\top\left(\frac{(\boldsymbol{\rho}^+)^2}{\boldsymbol{s}'^{*+}} - \frac{(\boldsymbol{\rho}^-)^2}{\boldsymbol{s}'^{*-}}\right)\right\|_{\overline{\boldsymbol{H}}^+} \\
&\leq (1 + \varepsilon_{\text{step}}\delta)\left\|\boldsymbol{C}^\top \boldsymbol{h}\right\|_{\overline{\boldsymbol{H}}^+} + 5\left\|\frac{\boldsymbol{r}^*}{\bar{\boldsymbol{r}}}\right\|_\infty^{1/2} \varepsilon_{\text{solve}} \cdot \varepsilon_{\text{step}} + 2\left\|\frac{\boldsymbol{r}'^*}{\bar{\boldsymbol{r}}}\right\|_\infty^{1/2} \varepsilon_{\text{step}}^2
\end{aligned}$$

where we have used the triangle inequality, the fact that

$$
\begin{aligned}
&\varepsilon_{\text{step}}\delta \left\| \boldsymbol{C}^\top \left( \frac{\boldsymbol{c}}{\mu} + \boldsymbol{r} g(\boldsymbol{s}) \right) \right\|_{\overline{\boldsymbol{H}}^+} \\
&= \varepsilon_{\text{step}}\delta \left\| \boldsymbol{C}^\top \left( \frac{\boldsymbol{c}}{\mu} + \frac{1}{\boldsymbol{s}^+} - \frac{1}{\boldsymbol{s}^-} \right) \right\|_{\overline{\boldsymbol{H}}^+} \\
&\le \varepsilon_{\text{step}}\delta \left\| \boldsymbol{C}^\top \left( \frac{\boldsymbol{c}}{\mu} + \frac{1}{\boldsymbol{s}^{*+}} - \frac{1}{\boldsymbol{s}^{*-}} \right) \right\|_{\overline{\boldsymbol{H}}^+} + \varepsilon_{\text{step}}\delta \left\| \boldsymbol{C}^\top \left( \frac{1}{\boldsymbol{s}^+} - \frac{1}{\boldsymbol{s}^{*+}} - \frac{1}{\boldsymbol{s}^-} + \frac{1}{\boldsymbol{s}^{*-}} \right) \right\|_{\overline{\boldsymbol{H}}^+} \\
&\le \varepsilon_{\text{step}}\delta \left\| \boldsymbol{C}^\top \left( \frac{\boldsymbol{c}}{\mu} + \frac{1}{\boldsymbol{s}^{*+}} - \frac{1}{\boldsymbol{s}^{*-}} \right) \right\|_{\overline{\boldsymbol{H}}^+} + \varepsilon_{\text{step}}\delta \left\| \frac{\boldsymbol{r}^*}{\bar{\boldsymbol{r}}} \right\|_\infty^{1/2} \left\| \boldsymbol{C}^\top \left( \frac{1}{\boldsymbol{s}^+} - \frac{1}{\boldsymbol{s}^{*+}} - \frac{1}{\boldsymbol{s}^-} + \frac{1}{\boldsymbol{s}^{*-}} \right) \right\|_{(\boldsymbol{C}^\top \boldsymbol{R}^* \boldsymbol{C})^+} \\
&\le \varepsilon_{\text{step}}\delta \left\| \boldsymbol{C}^\top \boldsymbol{h} \right\|_{\overline{\boldsymbol{H}}^+} + \varepsilon_{\text{step}}\delta \left\| \frac{\boldsymbol{r}^*}{\bar{\boldsymbol{r}}} \right\|_\infty^{1/2} \left\| \frac{\frac{1}{\boldsymbol{s}^+} - \frac{1}{\boldsymbol{s}^{*+}} - \frac{1}{\boldsymbol{s}^-} + \frac{1}{\boldsymbol{s}^{*-}}}{\left( \frac{1}{(\boldsymbol{s}^{*+})^2} + \frac{1}{(\boldsymbol{s}^{*-})^2} \right)^{1/2}} \right\|_2 \\
&\le \varepsilon_{\text{step}}\delta \left\| \boldsymbol{C}^\top \boldsymbol{h} \right\|_{\overline{\boldsymbol{H}}^+} + \varepsilon_{\text{step}}\delta \left\| \frac{\boldsymbol{r}^*}{\bar{\boldsymbol{r}}} \right\|_\infty^{1/2} \left( \left\| \frac{\frac{1}{\boldsymbol{s}^+} - \frac{1}{\boldsymbol{s}^{*+}}}{\left( \frac{1}{(\boldsymbol{s}^{*+})^2} \right)^{1/2}} \right\|_2 + \left\| \frac{\frac{1}{\boldsymbol{s}^-} - \frac{1}{\boldsymbol{s}^{*-}}}{\left( \frac{1}{(\boldsymbol{s}^{*-})^2} \right)^{1/2}} \right\|_2 \right) \\
&= \varepsilon_{\text{step}}\delta \left\| \boldsymbol{C}^\top \boldsymbol{h} \right\|_{\overline{\boldsymbol{H}}^+} + \varepsilon_{\text{step}}\delta \left\| \frac{\boldsymbol{r}^*}{\bar{\boldsymbol{r}}} \right\|_\infty^{1/2} \left( \left\| \frac{\boldsymbol{s}^{*+}}{\boldsymbol{s}^+} - \mathbf{1} \right\|_2 + \left\| \frac{\boldsymbol{s}^{*-}}{\boldsymbol{s}^-} - \mathbf{1} \right\|_2 \right) \\
&\le \varepsilon_{\text{step}}\delta \left\| \boldsymbol{C}^\top \boldsymbol{h} \right\|_{\overline{\boldsymbol{H}}^+} + \varepsilon_{\text{step}}\delta \left\| \frac{\boldsymbol{r}^*}{\bar{\boldsymbol{r}}} \right\|_\infty^{1/2} 2\varepsilon_{\text{solve}}\sqrt{m} \ (\text{as } \boldsymbol{s} \approx_{1+\varepsilon_{\text{solve}}} \boldsymbol{s}^*) \\
&= \varepsilon_{\text{step}}\delta \left\| \boldsymbol{C}^\top \boldsymbol{h} \right\|_{\overline{\boldsymbol{H}}^+} + 2 \left\| \frac{\boldsymbol{r}^*}{\bar{\boldsymbol{r}}} \right\|_\infty^{1/2} \varepsilon_{\text{step}}\varepsilon_{\text{solve}}
\end{aligned}
$$

and similarly, using $\left\| 1 - \left( \frac{\boldsymbol{s}^*}{\boldsymbol{s}} \right)^2 \right\|_\infty \le \varepsilon_{\text{solve}}(2 + \varepsilon_{\text{solve}})$, the fact that

$$
\begin{aligned}
&\left\| \boldsymbol{C}^\top \left( \left( \mathbf{1} - \left( \frac{\boldsymbol{s}^{*+}}{\boldsymbol{s}^+} \right)^2 \right) \frac{\boldsymbol{\rho}^+}{\boldsymbol{s}^{*+}} - \left( \mathbf{1} - \left( \frac{\boldsymbol{s}^{*-}}{\boldsymbol{s}^-} \right)^2 \right) \frac{\boldsymbol{\rho}^-}{\boldsymbol{s}^{*-}} \right) + \boldsymbol{C}^\top \left( \frac{(\boldsymbol{\rho}^+)^2}{\boldsymbol{s}'^{*+}} + \frac{(\boldsymbol{\rho}^-)^2}{\boldsymbol{s}'^{*-}} \right) \right\|_{\overline{\boldsymbol{H}}^+} \\
&\le \left\| \frac{\boldsymbol{r}^*}{\bar{\boldsymbol{r}}} \right\|_\infty^{1/2} \varepsilon_{\text{solve}}(2 + \varepsilon_{\text{solve}}) \left\| \boldsymbol{\rho} \right\|_2 + \left\| \frac{\boldsymbol{r}'^*}{\bar{\boldsymbol{r}}} \right\|_\infty^{1/2} \left\| \boldsymbol{\rho} \right\|_4^2 \\
&\le \left\| \frac{\boldsymbol{r}^*}{\bar{\boldsymbol{r}}} \right\|_\infty^{1/2} \varepsilon_{\text{step}}\varepsilon_{\text{solve}}(2 + \varepsilon_{\text{solve}})(1 + \varepsilon_{\text{solve}}) + \left\| \frac{\boldsymbol{r}'^*}{\bar{\boldsymbol{r}}} \right\|_\infty^{1/2} \varepsilon_{\text{step}}^2 (1 + \varepsilon_{\text{solve}})^2 \\
&\le 3 \left\| \frac{\boldsymbol{r}^*}{\bar{\boldsymbol{r}}} \right\|_\infty^{1/2} \varepsilon_{\text{step}}\varepsilon_{\text{solve}} + 2 \left\| \frac{\boldsymbol{r}'^*}{\bar{\boldsymbol{r}}} \right\|_\infty^{1/2} \varepsilon_{\text{step}}^2 .
\end{aligned}
$$

□

We will also use the following lemma, which is standard [10].

**Lemma 9.2.9** (Small residual implies $\ell_\infty$ closeness)**.** *Given a flow $\boldsymbol{f} = \boldsymbol{f}^0 + \boldsymbol{C}\boldsymbol{x}$ with slacks $\boldsymbol{s}$ and resistances $\boldsymbol{r}$, if $\left\| \boldsymbol{C}^\top \left( \frac{\boldsymbol{c}}{\mu} + \frac{1}{\boldsymbol{s}^+} - \frac{1}{\boldsymbol{s}^-} \right) \right\|_{(\boldsymbol{C}^\top \boldsymbol{R} \boldsymbol{C})^+} \le 1/1000$ then $\boldsymbol{f}$ is $(\mu, 1.01)$-central.*

Applying Lemma 9.2.8 for $T = \frac{k}{\varepsilon_{\text{step}}}$ iterations, we get the lemma below, which measures the closeness of $\boldsymbol{f}^*$ to the central path in $\ell_2$ after $T$ iterations.

**Lemma 9.2.10** (Centrality of $\boldsymbol{f}^*$)**.** *Let $\boldsymbol{f}^{*1}, \dots, \boldsymbol{f}^{*T+1}$ be flows with slacks $\boldsymbol{s}^{*1}, \dots, \boldsymbol{s}^{*T+1}$ and resistances $\boldsymbol{r}^{*1}, \dots, \boldsymbol{r}^{*T+1}$, and $\boldsymbol{f}^1, \dots, \boldsymbol{f}^{T+1}$ be flows with slacks $\boldsymbol{s}^1, \dots, \boldsymbol{s}^{T+1}$ and resistances $\boldsymbol{r}^1, \dots, \boldsymbol{r}^{T+1}$, such that $\boldsymbol{s}^t \approx_{1+\varepsilon_{\text{solve}}} \boldsymbol{s}^{*t}$ for all $t \in [T]$, where $T = \frac{k}{\varepsilon_{\text{step}}}$ for some $k \leq \sqrt{m}/10$, $\varepsilon_{\text{step}} \in (0, 0.1)$ and $\varepsilon_{\text{solve}} \in (0, 0.1)$. Additionally, we have that*

- *$\boldsymbol{f}^{*1}$ is $\mu$-central*
- *For all $t \in [T]$, $\boldsymbol{f}^{*t+1} = \boldsymbol{f}^{*t} + \varepsilon_{\text{step}} \cdot \widetilde{\boldsymbol{f}}^t$, where*

$$\left\| \sqrt{\boldsymbol{r}^t} \left( \widetilde{\boldsymbol{f}}^{*t} - \widetilde{\boldsymbol{f}}^t \right) \right\|_\infty \leq \varepsilon \,,$$

$$\widetilde{\boldsymbol{f}}^{*t} = \delta g(\boldsymbol{s}^t) - \delta (\boldsymbol{R}^t)^{-1} \boldsymbol{B} \left( \boldsymbol{B}^\top (\boldsymbol{R}^t)^{-1} \boldsymbol{B} \right)^+ \boldsymbol{B}^\top g(\boldsymbol{s}^t)$$

*and $\delta = \frac{1}{\sqrt{m}}$.*

*Then, $\boldsymbol{f}^{*T+1}$ is $(\mu/(1+\varepsilon_{\text{step}}\delta)^T, 1.01)$-central, as long as we set $\varepsilon_{\text{step}} \leq 10^{-5}k^{-3}$ and $\varepsilon_{\text{solve}} \leq 10^{-5}k^{-3}$.*

*Proof.* For all $t \in [T+1]$, we denote the residual of $\boldsymbol{f}^{*t}$ as $\boldsymbol{h}^t = \frac{\boldsymbol{c}(1+\varepsilon_{\text{step}}\delta)^{t-1}}{\mu} + \frac{1}{\boldsymbol{s}^{+,*t}} - \frac{1}{\boldsymbol{s}^{-,*t}}$. Note that $\boldsymbol{C}^\top \boldsymbol{h}^1 = \boldsymbol{0}$ as $\boldsymbol{f}^{*1}$ is $\mu$-central.

We assume that the statement of the lemma is not true, and let $\widehat{T}$ be the smallest $t \in [T+1]$ such that $\boldsymbol{f}^{*t}$ is not $(\mu/(1+\varepsilon_{\text{step}}\delta)^{t-1}, 1.01)$-central. Obviously $\widehat{T} > 1$. This means that $\boldsymbol{f}^{*t}$ is $(\mu/(1+\varepsilon_{\text{step}}\delta)^{t-1}, 1.01)$-central for all $t \in [\widehat{T}-1]$, i.e. $\boldsymbol{s}^{*t} \approx_{1.01} \boldsymbol{s}\left(\mu/(1+\varepsilon_{\text{step}}\delta)^{t-1}\right)$.

Also, note that by Lemma 9.2.7 about slack stability, and since $(1+\varepsilon_{\text{step}}\delta)^{|\widehat{T}-t|} \leq (1+\delta)^{1.1k}$, we have $\boldsymbol{s}\left(\mu/(1+\varepsilon_{\text{step}}\delta)^{t-1}\right) \approx_{3.7k^2} \boldsymbol{s}\left(\mu/(1+\varepsilon_{\text{step}}\delta)^{\widehat{T}-1}\right)$ for all $t \in [T+1]$. Additionally, note that, as shown in proof of Lemma 9.2.8, we have

$$\left\| \widetilde{\boldsymbol{f}}^{*\widehat{T}-1} \right\|_{\boldsymbol{r}^{\widehat{T}-1},\infty} \leq \left\| \widetilde{\boldsymbol{f}}^{*\widehat{T}-1} \right\|_{\boldsymbol{r}^{\widehat{T}-1},2} \leq 1 \,,$$

so

$$\begin{aligned}
\left\| \frac{\boldsymbol{s}^{*\widehat{T}}}{\boldsymbol{s}^{*\widehat{T}-1}} - \boldsymbol{1} \right\|_\infty &= \varepsilon_{\text{step}} \left\| \frac{\widetilde{\boldsymbol{f}}^{\widehat{T}-1}}{\boldsymbol{s}^{*\widehat{T}-1}} \right\|_\infty \\
&\leq \varepsilon_{\text{step}}(1+\varepsilon_{\text{solve}}) \left\| \sqrt{\boldsymbol{r}^{\widehat{T}-1}} \widetilde{\boldsymbol{f}}^{\widehat{T}-1} \right\|_\infty \\
&\leq \varepsilon_{\text{step}}(1+\varepsilon_{\text{solve}}) \left( \left\| \sqrt{\boldsymbol{r}^{\widehat{T}-1}} \widetilde{\boldsymbol{f}}^{*\widehat{T}-1} \right\|_\infty + \varepsilon \right) \\
&\leq \varepsilon_{\text{step}}(1+\varepsilon_{\text{solve}})\,(1+\varepsilon) \\
&\leq 1.3\varepsilon_{\text{step}} \,.
\end{aligned}$$

From this we conclude that $\boldsymbol{s}^{*\widehat{T}} \approx_{1+2.6\varepsilon_{\text{step}}} \boldsymbol{s}^{*\widehat{T}-1}$, and from the previous discussion we get that

$$\boldsymbol{s}^{*\widehat{T}} \approx_{1+2.6\varepsilon_{\text{step}}} \boldsymbol{s}^{*\widehat{T}-1} \approx_{1.01} \boldsymbol{s}(\mu/(1+\varepsilon_{\text{step}}\delta)^{\widehat{T}-1}) \approx_{3.7k^2} \boldsymbol{s}(\mu/(1+\varepsilon_{\text{step}}\delta)^{t-1}) \approx_{1.01} \boldsymbol{s}^{*t} \,,$$

so $\boldsymbol{s}^{*\widehat{T}} \approx_{4k^2} \boldsymbol{s}^{*t}$ for all $t \in [\widehat{T}-1]$.

On the other hand, if we apply Lemma 9.2.8 $\widehat{T}-1$ times with $\bar{\boldsymbol{r}}=\boldsymbol{r}^{*\widehat{T}}$, we get

$$
\begin{aligned}
&\left\|\boldsymbol{C}^\top \boldsymbol{h}^{\widehat{T}}\right\|_{\left(\boldsymbol{C}^\top \boldsymbol{R}^{\widehat{T}} \boldsymbol{C}\right)^+} \\
&= \left\|\boldsymbol{C}^\top \boldsymbol{h}^{\widehat{T}}\right\|_{\overline{\boldsymbol{H}}^+} \\
&\le (1+\varepsilon_{\text{step}}\delta)\left\|\boldsymbol{C}^\top \boldsymbol{h}^{\widehat{T}-1}\right\|_{\overline{\boldsymbol{H}}^+} + 5\left\|\frac{\boldsymbol{r}^{*\widehat{T}-1}}{\bar{\boldsymbol{r}}}\right\|_\infty^{1/2} \varepsilon_{\text{step}}\cdot\varepsilon_{\text{solve}} + 2\left\|\frac{\boldsymbol{r}^{*\widehat{T}}}{\bar{\boldsymbol{r}}}\right\|_\infty^{1/2}\varepsilon_{\text{step}}^2 \\
&\dots \\
&\le 5\sum_{t=1}^{\widehat{T}-1}(1+\varepsilon_{\text{step}}\delta)^{\widehat{T}-t-1}\left\|\frac{\boldsymbol{r}^{*t}}{\boldsymbol{r}^{*\widehat{T}}}\right\|_\infty^{1/2}\varepsilon_{\text{step}}\cdot\varepsilon_{\text{solve}} + 2\sum_{t=1}^{\widehat{T}-1}(1+\varepsilon_{\text{step}}\delta)^{\widehat{T}-t-1}\left\|\frac{\boldsymbol{r}^{*t+1}}{\boldsymbol{r}^{*\widehat{T}}}\right\|_\infty^{1/2}\varepsilon_{\text{step}}^2 \\
&\le 6T\max_{t\in[\widehat{T}-1]}\left\|\frac{\boldsymbol{r}^{*t}}{\boldsymbol{r}^{*\widehat{T}}}\right\|_\infty^{1/2}\varepsilon_{\text{step}}\cdot\varepsilon_{\text{solve}} + 2.4T\max_{t\in[\widehat{T}-1]}\left\|\frac{\boldsymbol{r}^{*t+1}}{\boldsymbol{r}^{*\widehat{T}}}\right\|_\infty^{1/2}\varepsilon_{\text{step}}^2 \\
&\le 24Tk^2\varepsilon_{\text{step}}\cdot\varepsilon_{\text{solve}} + 10Tk^2\varepsilon_{\text{step}}^2 \\
&= 24k^3\varepsilon_{\text{solve}} + 10k^3\varepsilon_{\text{step}} \\
&\le 1/1000\,,
\end{aligned}
$$

where we used the fact that $(1+\varepsilon_{\text{step}}\delta)^{\widehat{T}} \le e^{\varepsilon_{\text{step}}\delta T} = e^{\delta k} \le 1.2$ and our setting of $\varepsilon_{\text{solve}} \le 10^{-5}k^{-3}$ and $\varepsilon_{\text{step}} \le 10^{-5}k^{-3}$. By Lemma 9.2.9 this implies that $\boldsymbol{f}^{*\widehat{T}}$ is $(\mu/(1+\varepsilon_{\text{step}}\delta)^{\widehat{T}-1}, 1.01)$-central, a contradiction.

□

We are now ready to prove the following lemma, which is the goal of this section:

**Lemma 4.3.5.** *Let $\boldsymbol{f}^1,\dots,\boldsymbol{f}^{T+1}$ be flows with slacks $\boldsymbol{s}^t$ and resistances $\boldsymbol{r}^t$ for $t\in[T+1]$, where $T=\frac{k}{\varepsilon_{\text{step}}}$ for some $k\le\sqrt{m}/10$ and $\varepsilon_{\text{step}}=10^{-5}k^{-3}$, such that*

- *$\boldsymbol{f}^1$ is $(\mu, 1+\varepsilon_{\text{solve}}/8)$-central for $\varepsilon_{\text{solve}}=10^{-5}k^{-3}$*
- *For all $t\in[T]$, $\boldsymbol{f}^{t+1}=\begin{cases}\boldsymbol{f}(\mu)+\varepsilon_{\text{step}}\sum_{i=1}^{t}\widetilde{\boldsymbol{f}}^i & \text{if } \exists i\in[t]:\widetilde{\boldsymbol{f}}^i\neq\boldsymbol{0}\\ \boldsymbol{f}^1 & \text{otherwise}\end{cases}$, where*

$$\widetilde{\boldsymbol{f}}^{*t}=\delta g(\boldsymbol{s}^t)-\delta(\boldsymbol{R}^t)^{-1}\boldsymbol{B}\left(\boldsymbol{B}^\top(\boldsymbol{R}^t)^{-1}\boldsymbol{B}\right)^+\boldsymbol{B}^\top g(\boldsymbol{s}^t)$$

*for $\delta=\frac{1}{\sqrt{m}}$ and*

$$\left\|\sqrt{\boldsymbol{r}^t}\left(\widetilde{\boldsymbol{f}}^{*t}-\widetilde{\boldsymbol{f}}^t\right)\right\|_\infty\le\varepsilon$$

*for $\varepsilon=10^{-6}k^{-6}$.*

*Then, setting $\varepsilon_{\text{step}}=\varepsilon_{\text{solve}}=10^{-5}k^{-3}$ and $\varepsilon=10^{-6}k^{-6}$ we get that $\boldsymbol{s}^{T+1}\approx_{1.1}\boldsymbol{s}\left(\mu/(1+\varepsilon_{\text{step}}\delta)^{k\varepsilon_{\text{step}}^{-1}}\right)$.*

*Proof.* We set $\boldsymbol{f}^{*1}=\boldsymbol{f}(\mu)$ and for each $t\in[T]$,

$$\boldsymbol{f}^{*t+1}=\boldsymbol{f}^{*t}+\varepsilon_{\text{step}}\widetilde{\boldsymbol{f}}^{*t}\,,$$

and the corresponding slacks $\boldsymbol{s}^{*t}$ and resistances $\boldsymbol{r}^{*t}$. Let $\widehat{T}$ be the first $t \in [T+1]$ such that $\boldsymbol{s}^{\widehat{T}} \not\approx_{1+\varepsilon_{\text{solve}}} \boldsymbol{s}^{*\widehat{T}}$. Obviously $t > 1$ as $\boldsymbol{s}^1 \approx_{1+\varepsilon_{\text{solve}}/8} \boldsymbol{s}(\mu) = \boldsymbol{s}^{*1}$.

Now, for all $t \in [T]$ we have

$$\left\| \sqrt{\boldsymbol{r}^t} \left( \widetilde{\boldsymbol{f}}^{*t} - \widetilde{\boldsymbol{f}}^t \right) \right\|_\infty \leq \varepsilon \,.$$

Fix some $e \in E$. If $\widetilde{f}_e^t = \mathbf{0}$ for all $t \in [\widehat{T}-1]$, then we have $\sqrt{r_e^{\widehat{T}}} \left| \widetilde{f}_e^{*t} \right| = \sqrt{r_e^t} \left| \widetilde{f}_e^{*t} \right| \leq \varepsilon$ for all such $t$. This means that

$$\begin{aligned}
\sqrt{r_e^{\widehat{T}}} \left| f_e^{*\widehat{T}} - f_e^{\widehat{T}} \right| &\leq \sqrt{r_e^{\widehat{T}}} \left| f_e^{*\widehat{T}} - f_e^{*1} \right| + \sqrt{r_e^{\widehat{T}}} \left| f_e^{*1} - f_e^{\widehat{T}} \right| \\
&= \sqrt{r_e^{\widehat{T}}} \left| f_e^{*\widehat{T}} - f_e^{*1} \right| + \sqrt{r_e^{\widehat{T}}} \left| f_e^{*1} - f_e^1 \right| \\
&\leq \varepsilon_{\text{step}} \sqrt{r_e^{\widehat{T}}} \sum_{t=1}^{\widehat{T}-1} \left| \widetilde{f}_e^{*t} \right| + \sqrt{r_e^{\widehat{T}}} \left| f_e^{*1} - f_e^1 \right| \\
&\leq \widehat{T} \varepsilon_{\text{step}} \varepsilon + \sqrt{r_e^{\widehat{T}}} \left| f_e^{*1} - f_e^1 \right| \\
&\leq k\varepsilon + \sqrt{2}\varepsilon_{\text{solve}}/8 \\
&\leq \varepsilon_{\text{solve}}/2 \,,
\end{aligned}$$

as long as $\varepsilon \leq \varepsilon_{\text{solve}}/(2k) = O(1/k^4)$. In the second to last inequality we used Lemma 9.2.5.

Otherwise, there exists $t \in [\widehat{T}-1]$ such that $\widetilde{f}_e^t \neq \mathbf{0}$, and by definition $f_e^{\widehat{T}} = f_e(\mu) + \varepsilon_{\text{step}} \sum_{t=1}^{\widehat{T}-1} \widetilde{f}_e^t$, so

$$\begin{aligned}
\sqrt{r_e^{*\widehat{T}}} \left| f_e^{*\widehat{T}} - f_e^{\widehat{T}} \right| &\leq \sqrt{r_e^{*\widehat{T}}} \left| f_e^{*1} - f_e(\mu) \right| + \varepsilon_{\text{step}} \sum_{t=1}^{\widehat{T}-1} \sqrt{r_e^{*\widehat{T}}} \left| \widetilde{f}_e^{*t} - \widetilde{f}_e^t \right| \\
&\leq 3k^2 \varepsilon_{\text{step}} \sum_{t=1}^{\widehat{T}-1} \sqrt{r_e^{*t}} \left| \widetilde{f}_e^{*t} - \widetilde{f}_e^t \right| \\
&\leq 3k^2 \varepsilon_{\text{step}} (1+\varepsilon_{\text{solve}}) \sum_{t=1}^{\widehat{T}-1} \sqrt{r_e^t} \left| \widetilde{f}_e^{*t} - \widetilde{f}_e^t \right| \\
&\leq 3k^2 \varepsilon_{\text{step}} (1+\varepsilon_{\text{solve}}) T\varepsilon \\
&\leq 4k^3 \varepsilon \,,
\end{aligned}$$

where we have used Lemma 9.2.7 and the fact that $\boldsymbol{s}^t \approx_{1+\varepsilon_{\text{solve}}} \boldsymbol{s}^{*t}$ for all $t \in [\widehat{T}-1]$ which also implies that $\sqrt{\boldsymbol{r}^t} \approx_{1+\varepsilon_{\text{solve}}} \sqrt{\boldsymbol{r}^{*t}}$. Setting $\varepsilon = \frac{\varepsilon_{\text{solve}}}{8k^3} = O\left(\frac{1}{k^6}\right)$, this becomes $\leq \varepsilon_{\text{solve}}/2$.

Therefore we have proved that $\left\| \sqrt{\boldsymbol{r}^{*\widehat{T}}} \left( \boldsymbol{f}^{*\widehat{T}} - \boldsymbol{f}^{\widehat{T}} \right) \right\|_\infty \leq \varepsilon_{\text{solve}}/2$, and so $\boldsymbol{s}^{*\widehat{T}} \approx_{1+\varepsilon_{\text{solve}}} \boldsymbol{s}^{\widehat{T}}$, a contradiction. Therefore we conclude that $\boldsymbol{s}^t \approx_{1+\varepsilon_{\text{solve}}} \boldsymbol{s}^{*t}$ for all $t \in [T+1]$.

Now, as long as $\varepsilon_{\text{step}}, \varepsilon_{\text{solve}} \leq 10^{-5}k^{-3}$, we can apply Lemma 9.2.10, which guarantees that $\boldsymbol{s}^{*T+1} \approx_{1.01} \boldsymbol{s}\left(\mu/(1+\varepsilon_{\text{step}}\delta)^T\right)$, and so $\boldsymbol{s}^{T+1} \approx_{1.1} \boldsymbol{s}\left(\mu/(1+\varepsilon_{\text{step}}\delta)^T\right)$. Therefore we set $\varepsilon_{\text{step}} = \varepsilon_{\text{solve}} = 10^{-5}k^{-3}$ and $\varepsilon = 10^{-6}k^{-6} \leq \frac{\varepsilon_{\text{solve}}}{8k^3}$. □

**Proof of Lemma 4.3.11**

*Proof.* We will apply Lemma 4.3.5 with $\boldsymbol{f}^1$ being the flow corresponding to the resistances $\mathcal{L}.\boldsymbol{r} = \mathcal{C}.\boldsymbol{r}$, and $T = k\varepsilon_{\text{step}}^{-1}$. Note that it is important to maintain the invariant $\mathcal{L}.\boldsymbol{r} = \mathcal{C}.\boldsymbol{r}$ throughout the algorithm so that both data structures correspond to the same electrical flow problem. For each $t \in [T]$, for the $t$-th iteration, Lemma 4.3.5 requires an estimate $\widetilde{\boldsymbol{f}}^t$ such that $\left\|\sqrt{\boldsymbol{r}^t}\left(\widetilde{\boldsymbol{f}}^{*t} - \widetilde{\boldsymbol{f}}^t\right)\right\|_\infty \leq \varepsilon$, where

$$\widetilde{\boldsymbol{f}}^{*t} = \delta g(\boldsymbol{s}^t) - \delta(\boldsymbol{R}^t)^{-1}\boldsymbol{B}\left(\boldsymbol{B}^\top(\boldsymbol{R}^t)^{-1}\boldsymbol{B}\right)^+ \boldsymbol{B}^\top g(\boldsymbol{s}^t)$$

and $\delta = 1/\sqrt{m}$.

We claim that such an estimate can be computed for all $t$ by using $\mathcal{L}$ and $\mathcal{C}$. We apply the following process for each $t \in [T]$:

- Let $Z$ be the edge set returned by $\mathcal{L}$.SOLVE().
- Call $\mathcal{C}$.CHECK($e$) for each $e \in Z$ to obtain flow values $\widetilde{f}_e^t$.
- Compute $\boldsymbol{f}^t$ and its slacks $\boldsymbol{s}^{t+1}$ and resistances $\boldsymbol{r}^{t+1}$ as in Lemma 4.3.5, i.e.

$$f_e^{t+1} = \begin{cases} f_e(\mu) + \varepsilon_{\text{step}} \sum_{i=1}^{t} \widetilde{f}_e^i & \text{if } \exists i \in [t] : \widetilde{f}_e^i \neq 0 \\ f_e^1 & \text{otherwise} \end{cases}.$$

  This can be computed in $O(|Z|)$ by adding either $\varepsilon_{\text{step}}\widetilde{f}_e^i$ or $f_e(\mu) - f_e^1 + \varepsilon_{\text{step}}\widetilde{f}_e^i$ to $f_e^t$ for each $e \in Z$.

- Call $\mathcal{L}$.UPDATE($e, \boldsymbol{f}^{t+1}$) and $\mathcal{C}$.UPDATE($e, \boldsymbol{f}^{t+1}$) for all $e$ in the support of $\widetilde{\boldsymbol{f}}^t$. Note that $\mathcal{L}$.UPDATE works as long as

$$r_e^{\max}/\alpha \leq r_e^{t+1} \leq \alpha \cdot r_e^{\min},$$

  where $r_e^{\max}$, $r_e^{\min}$ are the maximum and minimum values of $\mathcal{L}.r_e$ since the last call to $\mathcal{L}$.BATCHUPDATE. After this, we have $\mathcal{L}.\boldsymbol{r} = \mathcal{C}.\boldsymbol{r} = \boldsymbol{r}^{t+1}$.

In the above process, when $\mathcal{L}$.SOLVE() is called we have $\mathcal{L}.\boldsymbol{r} = \boldsymbol{r}^t$ (for $t = 1$ this is true because $\mathcal{L}.\boldsymbol{r}$ are the resistances corresponding to $\boldsymbol{f}^1$). By the $(\alpha, \beta, \varepsilon/2)$-LOCATOR guarantees in Definition 4.3.7, with high probability $Z$ contains all the edges $e$ such that $\sqrt{r_e^t}\left|\widetilde{f}_e^{*t}\right| \geq \varepsilon/2$. Now, for each $e \in Z$, $\mathcal{C}$.CHECK($e$) returns a flow value $\widetilde{f}_e^t$ such that:

- $\sqrt{r_e^t}\left|\widetilde{f}_e^t - \widetilde{f}_e^{*t}\right| \leq \varepsilon$
- if $\sqrt{r_e^t}\left|\widetilde{f}_e^{*t}\right| < \varepsilon/2$, then $\widetilde{f}_e^t = 0$.

Therefore, the condition that

$$\sqrt{\boldsymbol{r}^t}\left\|\widetilde{\boldsymbol{f}}^t - \widetilde{\boldsymbol{f}}^{*t}\right\|_\infty \leq \varepsilon$$

is satisfied. Additionally $\widetilde{\boldsymbol{f}}^t$ is independent of the randomness of $\mathcal{L}$, because (the distribution of) $\widetilde{\boldsymbol{f}}^t$ would be the same if $\mathcal{C}$.CHECK was run for **all** edges $e$.

It remains to show that the LOCATOR requirement

$$\boldsymbol{r}^{\max}/\alpha \leq \boldsymbol{r}^{t+1} \leq \alpha \cdot \boldsymbol{r}^{\min}$$

is satisfied. Consider the minimum value of $t$ for which this is not satisfied. By Lemma 4.3.5, we have that

$$\boldsymbol{s}^{\tau+1} \approx_{1.1} \boldsymbol{s}\left(\mu/(1+\varepsilon_{\text{step}}\delta)^{\tau}\right) \tag{9.44}$$

for any $\tau \in [t]$.

Now let $\widehat{\boldsymbol{r}}$ be the resistances of $\mathcal{L}$ at any point since the last call to $\mathcal{L}$.BatchUpdate. By the lemma statement and (9.44), we know that $\widehat{\boldsymbol{r}} \approx_{1.1^2} \boldsymbol{r}(\hat{\mu})$ for some $\hat{\mu} \in [\mu/(1+\varepsilon_{\text{step}}\delta)^t, \mu^0]$. However, we also know that $\mu^0 \leq \mu \cdot (1+\varepsilon_{\text{step}}\delta)^{(0.5\alpha^{1/4}-k)\varepsilon_{\text{step}}^{-1}}$ and so

$$\frac{\hat{\mu}}{\mu/(1+\varepsilon_{\text{step}}\delta)^t} \leq (1+\varepsilon_{\text{step}}\delta)^{0.5\alpha^{1/4}\varepsilon_{\text{step}}^{-1}} \leq (1+\delta)^{0.5\alpha^{1/4}},$$

so by Lemma 9.2.7 we have

$$\boldsymbol{s}\left(\mu/(1+\varepsilon_{\text{step}}\delta)^t\right) \approx_{0.75\alpha^{1/2}} \boldsymbol{s}\left(\hat{\mu}\right).$$

As $\boldsymbol{s}^{t+1} \approx_{1.1} \boldsymbol{s}\left(\mu/(1+\varepsilon_{\text{step}}\delta)^t\right)$, we have that $\boldsymbol{s}^{t+1} \approx_{0.825\alpha^{1/2}} \boldsymbol{s}(\hat{\mu})$, and so

$$\boldsymbol{r}^{t+1} \approx_{0.825\alpha} \boldsymbol{r}(\hat{\mu}) \approx_{1.1^2} \widehat{\boldsymbol{r}}.$$

This means that $\boldsymbol{r}^{t+1} \approx_{\alpha} \widehat{\boldsymbol{r}}$ and is a contradiction.

We conclude that the requirements of $\mathcal{L}$ are met for all $t$, and as a result Lemma 4.3.5 shows that $\boldsymbol{s}^{T+1} \approx_{1.1} \boldsymbol{s}\left(\mu/(1+\varepsilon_{\text{step}}\delta)^{k\varepsilon_{\text{step}}^{-1}}\right)$. By Lemma 4.3.6, we can now obtain $\boldsymbol{f}\left(\mu/(1+\varepsilon_{\text{step}}\delta)^{k\varepsilon_{\text{step}}^{-1}}\right)$. Finally, we return $\mathcal{L}.\boldsymbol{r}$ and $\mathcal{C}$ to their original states.

**Success probability.** We note that all the outputs of $\mathcal{C}$ are independent of the randomness of $\mathcal{L}$, and $\mathcal{L}$ is only updated based on these outputs. As each operation of $\mathcal{L}$ succeeds with high probability, the whole process succeeds with high probability as well.

**Runtime.** The recentering operation in Lemma 4.3.6 takes $\widetilde{O}(m)$. Additionally, we call $\mathcal{L}$.Solve $k\varepsilon_{\text{step}}^{-1} = O(k^4)$ times and, as $|Z| = O(1/\varepsilon^2)$, the total number of times $\mathcal{L}$.Update, $\mathcal{C}$.Update, and $\mathcal{C}$.Check are called is $O(k\varepsilon_{\text{step}}^{-1}\varepsilon^{-2}) = O(k^{16})$.

□

**Proof of Lemma 4.3.10**

*Proof.* Let $\delta = 1/\sqrt{m}$. Over a number of $T = \widetilde{O}\left(m^{1/2}/k\right)$ iterations, we will repeatedly apply MultiStep (Lemma 4.3.11). We will also replace the oracle from Definition 4.3.9 by the Checker data structure in Section 9.2.6.

**Initialization.** We first initialize the Locator with error $\varepsilon/2$, by calling $\mathcal{L}$.Initialize($\boldsymbol{f}$). Let $\boldsymbol{s}^t$ be the slacks $\mathcal{L}.\boldsymbol{s}$ before the $t$-th iteration and $\boldsymbol{r}^t$ the corresponding resistances, and $\boldsymbol{s}^{0t}$ be the slacks $\mathcal{L}.\boldsymbol{r}^0$ before the $t$-th iteration and $\boldsymbol{r}^{0t}$ the corresponding resistances, for $t \in [T]$. Also, we let $\mu_t = \mu/\left(1+\varepsilon_{\text{step}}\delta\right)^{(t-1)k\varepsilon_{\text{step}}^{-1}}$. We will maintain the invariant that $\boldsymbol{s}^t \approx_{1+\varepsilon_{\text{solve}}/8} \boldsymbol{s}\left(\mu_t\right)$, which is a requirement in order to apply Lemma 4.3.11.

As in [73], we will also need to maintain $O(k^4)$ Checkers $\mathcal{C}^i$ for $i \in [O(k^4)]$, so we call $\mathcal{C}^i$.Initialize($\boldsymbol{f}, \varepsilon, \beta_{\text{Checker}}$) for each one of these. Note that in general $\beta_{\text{Checker}} \neq \beta$, as the vertex sparsifiers $\mathcal{L}$ and $\mathcal{C}^i$ will not be on the same vertex set. As in Lemma 4.3.11, we will maintain the invariant that $\mathcal{L}.\boldsymbol{r} = \mathcal{C}^i.\boldsymbol{r}$ for all $i$.

**Resistance updates.** Assuming that all the requirements of Lemma 4.3.11 (MULTISTEP) are satisfied at the $t$-th iteration, that lemma computes a flow $\bar{\boldsymbol{f}} = \boldsymbol{f}(\mu_{t+1})$ with slacks $\bar{\boldsymbol{s}}$. In order to guarantee that $\boldsymbol{s}^{t+1} \approx_{1+\varepsilon_{\text{solve}}/8} \boldsymbol{s}(\mu_{t+1})$, we let $Z$ be the set of edges such that either

$$s_e^{+,t} \not\approx_{1+\varepsilon_{\text{solve}}/8} \bar{s}_e^{+} \text{ or } s_e^{-,t} \not\approx_{1+\varepsilon_{\text{solve}}/8} \bar{s}_e^{-}$$

and then call $\mathcal{L}$.UPDATE$(e, \bar{\boldsymbol{f}})$ for all $e \in Z$. This guarantees that $s_e^{+,t+1} = \bar{s}_e^{+}$ and $s_e^{-,t+1} = \bar{s}_e^{-}$ for all $e \in Z$ and so $\boldsymbol{s}^{t+1} \approx_{1+\varepsilon_{\text{solve}}/8} \bar{\boldsymbol{s}} = \boldsymbol{s}(\mu_{t+1})$. We also apply the same updates to the $\mathcal{C}^i$'s using $\mathcal{C}^i$.UPDATE, in order to ensure that they have the same resistances with $\mathcal{L}$.

**Batched resistance updates.** The number of times $\mathcal{L}$.UPDATE is called can be quite large because of multiple edges on which error slowly accumulates. This is because in general $\Omega(m)$ resistances will be updated throughout the algorithm. As LOCATOR.UPDATE is only slightly sublinear, this would lead to an $\Omega(m^{3/2})$-time algorithm. For this reason, as in [73], we occasionally (every $\widehat{T}$ iterations for some $\widehat{T} \geq 1$ to be defined later) perform batched updates by calling $\mathcal{L}$.BATCHUPDATE$(Z, \bar{\boldsymbol{f}})$, where $Z$ is the set of edges such that either

$$s_e^{+,t} \not\approx_{1+\varepsilon_{\text{solve}}/16} \bar{s}_e^{+} \text{ or } s_e^{-,t} \not\approx_{1+\varepsilon_{\text{solve}}/16} \bar{s}_e^{-} .$$

This again guarantees that $s_e^{+,t+1} = \bar{s}_e^{+}$ and $s_e^{-,t+1} = \bar{s}_e^{-}$ for all $e \in Z$ and so $\boldsymbol{s}^{t+1} \approx_{1+\varepsilon_{\text{solve}}/16} \bar{\boldsymbol{s}} = \boldsymbol{s}(\mu_{t+1})$. Note that after updating $\mathcal{L}.\boldsymbol{s}$ and $\mathcal{L}.\boldsymbol{r}$, this operation also sets $\mathcal{L}.\boldsymbol{r}^0 = \mathcal{L}.\boldsymbol{r}$. We perform the same resistance updates to the $\mathcal{C}^i$'s in the regular (i.e. not batched) way, using $\mathcal{C}^i$.UPDATE.

**LOCATOR requirements.** What is left is to ensure that the requirements of Lemma 4.3.11 are satisfied at the $t$-th iteration, as well as that the requirements of $\mathcal{L}$.UPDATE and $\mathcal{L}$.BATCHUPDATE from Definition 4.3.7 are satisfied.

The requirements are as follows:

1. Lemma 4.3.11: $\boldsymbol{s}^{0t} \approx_{1+\varepsilon_{\text{solve}}/8} \boldsymbol{s}\left(\mu^0\right)$ for some $\mu^0 \leq \mu_t \cdot (1+\varepsilon_{\text{step}}\delta)^{(0.5\alpha^{1/4}-k)\varepsilon_{\text{step}}^{-1}}$.

   Note that $\mathcal{L}.\boldsymbol{s}^0$ is updated every time $\mathcal{L}$.BATCHUPDATE is called, and after the call we have $\mathcal{L}.\boldsymbol{s}^0 = \mathcal{L}.\boldsymbol{s} \approx_{1+\varepsilon_{\text{solve}}/16} \boldsymbol{s}(\mu^0)$ for some $\mu^0 > 0$. To ensure that it is called often enough, we call $\mathcal{L}$.BATCHUPDATE$(\emptyset)$ every $(0.5\alpha^{1/4}/k - 1)\varepsilon_{\text{step}}^{-1}$ iterations. Because of this, we have $\mu^0 \leq \mu_t \cdot (1+\varepsilon_{\text{step}}\delta)^{(0.5\alpha^{1/4}/k-1)\varepsilon_{\text{step}}^{-1}\cdot k} = \mu_t \cdot (1+\varepsilon_{\text{step}}\delta)^{(0.5\alpha^{1/4}-k)\varepsilon_{\text{step}}^{-1}}$. Additionally, for any resistances $\widehat{\boldsymbol{r}}$ that $\mathcal{L}$ had at any point since the last call to $\mathcal{L}$.BATCHUPDATE, it is immediate that

   $$\widehat{\boldsymbol{r}} \approx_{(1+\varepsilon_{\text{solve}}/8)^2} \boldsymbol{r}(\hat{\mu})$$

   for some $\hat{\mu} \in [\mu_t, \mu^0]$, as this is exactly the invariant that our calls to $\mathcal{L}$.UPDATE maintain. Therefore, $\widehat{\boldsymbol{r}} \approx_{1.1^2} \boldsymbol{r}(\hat{\mu})$.

2. $\mathcal{L}$.UPDATE: $r_e^{\max}/\alpha \leq r_e^{t+1} \leq \alpha \cdot r_e^{\min}$, where $r_e^{\min}, r_e^{\max}$ are the minimum and maximum values that $\mathcal{L}.r_e$ has had since the last call to $\mathcal{L}$.BATCHUPDATE.

   Let $\widehat{\boldsymbol{r}}$ be any value of $\mathcal{L}.\boldsymbol{r}$ since the last call to $\mathcal{L}$.BATCHUPDATE. Because of the invariant maintained by resistance updates (including inside MULTISTEP), we have that $\widehat{\boldsymbol{r}}$ are $(\hat{\mu}, 1.1)$-central resistances for some $\hat{\mu}$ such that

   $$\mu_{t+1} \leq \hat{\mu} \leq \mu_{t+1} \cdot (1+\varepsilon_{\text{step}}\delta)^{(0.5\alpha^{1/4}-k)\varepsilon_{\text{step}}^{-1}} \leq \mu_{t+1} \cdot (1+\delta)^{0.5\alpha^{1/4}} .$$

As in the previous item, we have that and $\boldsymbol{s}^{0t}$ are $(\mu^0, 1+\varepsilon_{\text{solve}}/8)$-central. By Lemma 9.2.7 this implies

$$\boldsymbol{s}(\mu_{t+1}) \approx_{0.75\alpha^{1/2}} \boldsymbol{s}(\hat{\mu})\,,$$

and since $\widehat{\boldsymbol{r}} \approx_{1.1^2} \boldsymbol{r}(\hat{\mu})$, we conclude that $\boldsymbol{r}(\mu_{t+1}) \approx_\alpha \widehat{\boldsymbol{r}}$.

3. $\mathcal{L}$.BatchUpdate: Between any two successive calls to $\mathcal{L}$.Initialize, the number of edges updated (number of calls to $\mathcal{L}$.Update plus the sum of $|Z|$ for all calls to $\mathcal{L}$.BatchUpdate) is $O(\beta m)$.

   We make sure that this is satisfied by calling $\mathcal{L}$.Initialize($\bar{\boldsymbol{f}}$) every $\varepsilon_{\text{solve}}\sqrt{\beta m}/k$ iterations, where $\bar{\boldsymbol{f}} = \boldsymbol{f}(\mu_t)$, at the beginning of the $t$-th iteration.

   Consider any two successive initializations at iterations $t^{init}$ and $t^{end}$ respectively. Let $\ell$ be the number of edges $e$ that have potentially been updated, i.e. such that either

$$s_e(\mu_{t^{init}})^+ \not\approx_{1+\varepsilon_{\text{solve}}/16} s_e(\mu_i)^+ \text{ or } s_e(\mu_{t^{init}})^- \not\approx_{1+\varepsilon_{\text{solve}}/16} s_e(\mu_i)^-$$

   for some $i \in \left[t^{init}, t^{end}\right]$. First, note that this implies that

$$\sqrt{r_e(\mu_{t^{init}})}\,|f_e(\mu_{t^{init}}) - f_e(\mu_i)| > \frac{\varepsilon_{\text{solve}}/16}{1+\varepsilon_{\text{solve}}/16} > \varepsilon_{\text{solve}}/17\,.$$

   Now, by the fact that $t^{end} - t^{init} \le \varepsilon_{\text{solve}}\sqrt{\beta m}/k$, we have that

$$\mu_{t^{init}} \le \mu_i \cdot (1+\varepsilon_{\text{step}}\delta)^{k\varepsilon_{\text{step}}^{-1}\cdot\varepsilon_{\text{solve}}\sqrt{\beta m}/k} \le \mu_i \cdot (1+\delta)^{\varepsilon_{\text{solve}}\sqrt{\beta m}}\,.$$

   By applying Lemma 9.2.6 with $k = \varepsilon_{\text{solve}}\sqrt{\beta m}$ and $\gamma = \varepsilon_{\text{solve}}/17$, we get that $\ell \le O(\beta m)$. Therefore the statement follows.

We will also need to show how to implement the PerfectChecker used in MultiStep using the $\mathcal{C}^i$'s, as well as how to satisfy all Checker requirements.

**Checker requirements.**

1. Implementing $\varepsilon$-PerfectChecker inside MultiStep.

   We follow almost the same procedure as in [73], other than the fact that we also need to provide some additional information to $\mathcal{C}^i$.Solve. Each call to PerfectChecker.Update translates to calls to $\mathcal{C}^i$.TemporaryUpdate for all $i$. In addition, the $i$-th batch of calls to PerfectChecker.Check inside MultiStep (i.e. that corresponding to a single set of edges returned by $\mathcal{L}$.Solve) is only run on $\mathcal{C}^i$ using $\mathcal{C}^i$.Check. As each call to $\mathcal{C}^i$.Check is independent of previous calls to it, we can get correct outputs with high probability even when we run it multiple times (one for each edge returned by $\mathcal{L}$.Solve).

   In order to guarantee that we have a vector $\boldsymbol{\pi}^i_{old}$ as required by $\mathcal{C}^i$.Check, once every $k^4$ calls to MultiStep (i.e. if $t$ is a multiple of $k^4$) we compute

$$\boldsymbol{\pi}^i_{old} = \boldsymbol{\pi}^{C^{i,t}}\left(\boldsymbol{B}^\top g(\boldsymbol{s}(\mu_t))\right)$$

   for all $i \in [O(k^4)]$, where $C^{i,t}$ is the vertex set of the Schur complement data structure stored internally by the $\mathcal{C}^i$ right before the $t$-th call to MultiStep. This can be computed in $\widetilde{O}(m)$ for each $i$ as in DemandProjector.Initialize in Lemma 4.4.14. Now, the total number

of $\mathcal{C}^i$.TemporaryUpdates that have not been rolled back is $O(k^{16})$, and the total number of $\mathcal{C}^i$.Updates over $k^4$ calls to MultiStep by Lemma 9.2.6 is $O(k^{10}/\varepsilon_{\text{solve}}^2) = O(k^{16})$. This means that the total number of terminal insertions to $C^{i,t}$ as well as resistance changes is $O(k^{16})$. By Lemma 4.4.12, if $C^i$ is the current state of the vertex set of the Schur complement of $\mathcal{C}^i$ and $\boldsymbol{s}$ are the current slacks,

$$\mathcal{E}_{\boldsymbol{r}}\left(\boldsymbol{\pi}^i_{old} - \boldsymbol{\pi}^{C^i}(g(\boldsymbol{s}))\right) \leq \widetilde{O}\left(\alpha'\beta^{-4}_{\textsc{Checker}}\right)\cdot k^{32}\,,$$

where $\alpha'$ is the largest possible multiplicative change of some $\mathcal{C}^i.r_e$ since the computation of $\boldsymbol{\pi}^i_{old}$. Furthermore, note that $\boldsymbol{\pi}_{old}$ is supported on $C^i$. This is because $C^{i,t} \subseteq C^i$ and $C^{i,t}$ does not contain any temporary terminals.

Now, as we have already proved in Lemma 4.3.11, at any point inside the $t$-th call to MultiStep, $\mathcal{C}^i.\boldsymbol{r}$ are $(\hat{\mu}, 1.1)$-central resistances for some $\hat{\mu} \in [\mu_{t+1}, \mu_t]$.

Fix $\hat{\mu} \in [\mu_{t+1}, \mu_t], \hat{\mu}' \in [\mu_{t'+1}, \mu_{t'}]$, as well as the corresponding resistances of $\mathcal{C}^i$, $\widehat{\boldsymbol{r}}$, $\widehat{\boldsymbol{r}}'$, where $t' \geq t$. Now, note that since we are computing $\boldsymbol{\pi}^i_{old}$ every $k^4$ calls to MultiStep, we have that

$$\frac{\hat{\mu}}{\hat{\mu}'} \leq \frac{\mu_t}{\mu_{t'+1}} \leq (1+\varepsilon_{\text{step}}\delta)^{k\varepsilon^{-1}_{\text{step}}\cdot(t'-t+1)} \leq (1+\delta)^{O(k^5)}\,,$$

so Lemma 9.2.7 implies that

$$\boldsymbol{s}(\hat{\mu}) \approx_{O(k^{10})} \boldsymbol{s}(\hat{\mu}')\,.$$

As $\widehat{\boldsymbol{r}} \approx_{1.1^2} \boldsymbol{r}(\hat{\mu})$ and $\widehat{\boldsymbol{r}}' \approx_{1.1^2} \boldsymbol{r}(\hat{\mu}')$, we get that $\widehat{\boldsymbol{r}} \approx_{O(k^{20})} \widehat{\boldsymbol{r}}'$. Therefore, $\alpha' \leq O(k^{20})$. Setting $\beta_{\textsc{Checker}} \geq \widetilde{\Omega}\left(\alpha'^{1/4}k^8\varepsilon^{-1/2}m^{-1/4}\right) = \widetilde{\Omega}\left(k^{16}/m^{1/4}\right)$, we get that

$$\widetilde{O}\left(\alpha'\beta^{-4}_{\textsc{Checker}}\right)\cdot k^{32} \leq \varepsilon^2 m/4\,,$$

as required by $\mathcal{C}^i$.Check.

Finally, at the end of MultiStep we bring all $\mathcal{C}^i$ to their original state before calling MultiStep, by calling $\mathcal{C}^i$.Rollback. We also update all the resistances of $\mathcal{L}$ to their original state by calling $\mathcal{L}$.Update.

2. Between any two successive calls to $\mathcal{C}^i$.Initialize, the total number of edges updated at any point (via $\mathcal{C}^i$.Update or $\mathcal{C}^i$.TemporaryUpdate that have not been rolled back) is $O(\beta_{\textsc{Checker}}m)$.

   For Update, we can apply a similar analysis as in the Locator case to show that if we call $\mathcal{C}^i$.Initialize every $\varepsilon_{\text{solve}}\sqrt{\beta_{\textsc{Checker}}m}/k$ iterations, the total number of updates never exceeds $O(\beta_{\textsc{Checker}}m)$. For TemporaryUpdate, note that at any time there are at most $O(k^{16})$ of these that have not been rolled back (this is inside MultiStep). Therefore, as long as $k^{16} \leq O(\beta_{\textsc{Checker}}m) \Leftrightarrow \beta_{\textsc{Checker}} \geq \Omega(k^{16}/m)$, the requirement is met.

**Output Guarantee.** After the application of Lemma 4.3.11 at the last iteration, we will have $\bar{\boldsymbol{f}} = \boldsymbol{f}(\mu_{T+1})$, where $\mu_{T+1} = \mu/(1+\varepsilon_{\text{step}}\delta)^{\widetilde{O}(m^{1/2}\varepsilon^{-1}_{\text{step}})} = \mu/\text{poly}(m) \leq m^{-10}$.

**Success probability.** Note that all operations of Locator and Checker work with high probability. Regarding the interaction of the randomness of these data structures and the fact that they work against oblivious adversaries, we defer to [73], where there is a detailed discussion of why this works.

In short, note that outside of MultiStep, all updates are deterministic (as they only depend on the central path), and in MultiStep the updates to Locator and Checker only depend on outputs of a Checker. As each time we are getting the output from a different Checker, the inputs to $\mathcal{C}^i$.Check are independent of the randomness of $\mathcal{C}^i$, and thus succeed with high probability. Finally, note that the output of Locator is only passed onto $\mathcal{C}^i$.Check, whose output is then independent of the inputs received by Locator. Therefore, Locator does not "leak" any randomness.

Our only deviation from [73] in Checker has to do with the extra input of Checker.Check ($\boldsymbol{\pi}^i_{old}$). However, note that this is computed outside of MultiStep, and as such the only randomness it depends on is the $\beta_{\text{Checker}}$-congestion reduction subset $C^i$ generated when calling $\mathcal{C}^i$.Initialize. As such, it only depends on the internal randomness of $\mathcal{C}^i$. As we mentioned, the output of $\mathcal{C}^i$.Check is never fed back to $\mathcal{C}^i$, and thus the operation works with high probability.

**Runtime (except Checker).** Each call to MultiStep (Lemma 4.3.11) takes time $\widetilde{O}(m)$ plus $O(k^{16})$ calls to $\mathcal{L}$.Update and $O(k^4)$ calls to $\mathcal{L}$.Solve. As the total number of iterations is $\widetilde{O}\left(m^{1/2}/k\right)$, the total time because of calls to MultiStep is $\widetilde{O}\left(m^{3/2}/k\right)$, plus $\widetilde{O}\left(m^{1/2}k^{15}\right)$ calls to $\mathcal{L}$.Update and $\widetilde{O}\left(m^{1/2}k^3\right)$ calls to $\mathcal{L}$.Solve.

Now, the total number of calls to $\mathcal{L}$.Initialize is $\widetilde{O}\left(\frac{m^{1/2}/k}{\varepsilon_{\text{solve}}\sqrt{\beta m}/k}\right) = \widetilde{O}\left(k^3\beta^{-1/2}\right)$.

The total number of calls to $\mathcal{L}$.BatchUpdate($\emptyset$) is $\widetilde{O}\left(\frac{m^{1/2}/k}{0.5\alpha^{1/4}/k}\right) = \widetilde{O}\left(m^{1/2}\alpha^{-1/4}\right)$ and the total number of calls to $\mathcal{L}$.BatchUpdate($Z, \boldsymbol{f}$) is $\widetilde{O}\left(\frac{m^{1/2}}{k\widehat{T}}\right)$. Regarding the size of $Z$, let us focus on the calls to $\mathcal{L}$.BatchUpdate($Z, \boldsymbol{f}$) between two successive calls to $\mathcal{L}$.Initialize. We already showed that the sum of $|Z|$ over all calls during this interval is $O(\beta m)$. Therefore the total sum of $|Z|$ over all iterations of the algorithm is $\widetilde{O}\left(mk^3\beta^{1/2}\right)$.

In order to bound the number of calls to $\mathcal{L}$.Update, we concentrate on those between two successive calls to $\mathcal{L}$.BatchUpdate($Z, \boldsymbol{f}$) in iterations $t^{old}$ and $t^{new} > t^{old}$. After the call to $\mathcal{L}$.BatchUpdate($Z, \boldsymbol{f}$) in iteration $t^{old}$ we have $\mathcal{L}.\boldsymbol{s} \approx_{1+\varepsilon_{\text{solve}}/16} \boldsymbol{s}(\mu_{t^{old}})$. Fix $\mu \in \left[\mu(t^{new}), \mu(t^{old})\right]$ and let $\ell$ be the number of $e \in E$ such that $s_e(\mu) \not\approx_{1+\varepsilon_{\text{solve}}/8} s_e^{t^{old}}$. As $s_e^{t^{old}} \approx_{1+\varepsilon_{\text{solve}}/16} s_e(\mu_{t^{old}})$ by the guarantees of $\mathcal{L}$.BatchUpdate, this implies that $s_e(\mu) \not\approx_{1+\varepsilon_{\text{solve}}/16} s_e(\mu_{t^{old}})$, and so

$$\sqrt{r_e(\mu^{t^{old}})}\,|f_e(\mu_{t^{old}}) - f_e(\mu)| \geq \frac{\varepsilon_{\text{solve}}/16}{1+\varepsilon_{\text{solve}}/16} > \varepsilon_{\text{solve}}/17\,.$$

As

$$\mu_{t^{old}} \leq \mu \cdot (1+\varepsilon_{\text{step}}\delta)^{k\varepsilon_{\text{step}}^{-1}\cdot\widehat{T}} \leq \mu \cdot (1+\delta)^{k\cdot\widehat{T}}\,,$$

by applying Lemma 9.2.6 with $k = (1+\delta)^{k\cdot\widehat{T}}$ and $\gamma = \varepsilon_{\text{solve}}/17$, we get that $\ell \leq O(k^2\widehat{T}^2\varepsilon_{\text{solve}}^{-2})$. As there are $\widetilde{O}\left(\frac{m^{1/2}}{k\widehat{T}}\right)$ calls to $\mathcal{L}$.BatchUpdate($Z, \boldsymbol{f}$), the total number of calls to $\mathcal{L}$.Update is

$$\widetilde{O}\left(m^{1/2}\widehat{T}\varepsilon_{\text{solve}}^{-2}\right)\widetilde{O}\left(m^{1/2}k^6\widehat{T}\right)\,.$$

We conclude that we have runtime $\widetilde{O}\left(m^{3/2}/k\right)$, plus

- $\widetilde{O}\left(k^3\beta^{-1/2}\right)$ calls to $\mathcal{L}$.Initialize,
- $\widetilde{O}\left(m^{1/2}k^3\right)$ calls to $\mathcal{L}$.Solve,
- $\widetilde{O}\left(m^{1/2}\left(k^6\widehat{T} + k^{15}\right)\right)$ calls to $\mathcal{L}$.Update,

- $\widetilde{O}\left(m^{1/2}\alpha^{-1/4}\right)$ calls to $\mathcal{L}$.BatchUpdate($\emptyset$), and
- $\widetilde{O}\left(m^{1/2}k^{-1}\widehat{T}^{-1}\right)$ calls to $\mathcal{L}$.BatchUpdate($Z, \boldsymbol{f}$).

**Runtime of Checker.** We look at each operation separately. We begin with the runtime of Checker.Check. We have

$$\widetilde{O}\left(\underbrace{m^{1/2}/k}_{\#\text{ calls to MultiStep}} \cdot \underbrace{k^{16}}_{\#\text{ calls in each MultiStep}} \cdot \underbrace{(\beta_{\text{Checker}}m + (k^{16}\beta_{\text{Checker}}^{-2}\varepsilon^{-2})^2))\varepsilon^{-2}}_{\text{runtime per call}}\right).$$

To make the first term $\widetilde{O}\left(m^{3/2}/k\right)$, we set $\beta_{\text{Checker}} = k^{-28}$. Note that this satisfies our previous requirements that $\beta_{\text{Checker}} \geq \widetilde{\Omega}\left(k^{16}/m\right)$ and $\beta_{\text{Checker}} \geq \widetilde{\Omega}\left(k^{16}/m^{1/4}\right)$ as long as $k \leq m^{1/176}$. Therefore the total runtime because of this operation is $\widetilde{O}\left(m^{3/2}/k + m^{1/2}k^{195}\right)$.

For Checker.Initialize, we have

$$\widetilde{O}\left(\underbrace{k^4}_{\#\text{Checkers}} \cdot \underbrace{k^3\beta_{\text{Checker}}^{-1/2}}_{\#\text{times initialized}} \cdot \underbrace{m\beta_{\text{Checker}}^{-4}\varepsilon^{-4}}_{\text{runtime per init}}\right) = \widetilde{O}\left(mk^{157}\right).$$

For Checker.Update, similarly with the analysis of Locator but noting that there are no batched updates, we have

$$\widetilde{O}\left(\underbrace{k^4}_{\#\text{Checkers}} \cdot \underbrace{m\varepsilon_{\text{solve}}^{-2}}_{\#\text{calls per Checker}} \cdot \underbrace{\beta_{\text{Checker}}^{-2}\varepsilon^{-2}}_{\text{runtime per call}}\right) = \widetilde{O}\left(mk^{78}\right).$$

For Checker.TemporaryUpdate, we have

$$\widetilde{O}\left(\underbrace{k^4}_{\#\text{Checkers}} \cdot \underbrace{m^{1/2}/k}_{\#\text{ calls to MultiStep}} \cdot \underbrace{k^{16}}_{\#\text{ calls per MultiStep}} \cdot \underbrace{(k^{16}\beta_{\text{Checker}}^{-2}\varepsilon^{-2})^2}_{\text{runtime per call}}\right) = \widetilde{O}\left(m^{1/2}k^{187}\right).$$

Finally, note that, by definition, computing the vectors $\boldsymbol{\pi}_{old}^i$ takes $\widetilde{O}\left(m^{3/2}/k\right)$, as we do it once per $k^4$ calls to MultiStep and it takes $\widetilde{O}\left(mk^4\right)$.

As long as $k \leq m^{1/316}$, the total runtime because of Checker is $\widetilde{O}\left(m^{3/2}/k\right)$. □

### 9.2.4 Deferred Proofs from Section 4.4

#### Proof of Lemma 4.4.4

We first provide a helper lemma for upper bounding escape probabilities in terms of the underlying graph's resistances.

**Lemma 9.2.11** (Bounding escape probabilities)**.** *Let a graph with resistances $\boldsymbol{r}$, and consider a random walk which at each step moves from the current vertex $u$ to an adjacent vertex $v$ sampled with probability proportional to $1/r_{uv}$. Let $p_u^{\{u,t\}}(s)$ represent the probability that a walk starting at $s$*

*hits $u$ before $t$. Then*

$$p_u^{\{u,t\}}(s) = \frac{R_{eff}(s,t)}{R_{eff}(u,t)} \cdot p_s^{\{s,t\}}(u) \leq \frac{R_{eff}(s,t)}{R_{eff}(u,t)} \leq \frac{r_{st}}{R_{eff}(u,t)}\,.$$

*Proof.* Using standard arguments we can prove that if $\boldsymbol{L}$ is the Laplacian associated with the underlying graph, then

$$p_u^{\{u,t\}}(s) = \frac{(\mathbf{1}_s - \mathbf{1}_t)^\top \boldsymbol{L}^+ (\mathbf{1}_u - \mathbf{1}_t)}{R_{eff}(u,t)}\,.$$

This immediately yields the claim as we can further write it as

$$p_u^{\{u,t\}}(s) = \frac{R_{eff}(s,t)}{R_{eff}(u,t)} \cdot \frac{(\mathbf{1}_u - \mathbf{1}_t)^\top \boldsymbol{L}^+ (\mathbf{1}_s - \mathbf{1}_t)}{R_{eff}(s,t)} = \frac{R_{eff}(s,t)}{R_{eff}(u,t)} \cdot p_s^{\{s,t\}}(u) \leq \frac{R_{eff}(s,t)}{R_{eff}(u,t)}\,,$$

where we crucially used the symmetry of $\boldsymbol{L}$. The final inequality is due to the fact that $R_{eff}(s,t) \leq r_{st}$.

Now let us prove the claimed identity for escape probabilities. Let $\boldsymbol{\psi}$ be the vector defined by $\psi_i = p_u^{\{u,t\}}(i)$ for all $i \in V$, which clearly satisfies $\psi_u = 1$ and $\psi_t = 0$. Furthermore, for all $i \notin \{u,t\}$ we have

$$\psi_i = \sum_{j \sim i} \frac{r_{ij}^{-1}}{\sum_{k \sim i} r_{ik}^{-1}} \psi_j\,,$$

which can be written in short as

$$(\boldsymbol{L}\boldsymbol{\psi})_i = 0 \quad \text{for all } i \notin \{s,t\}\,.$$

Now we solve the corresponding linear system. We interpret $\boldsymbol{\psi}$ as electrical potentials corresponding to routing $1/R_{eff}(u,t)$ units of electrical flow from $u$ to $t$. Indeed, by Ohm's law, this corresponds to a potential difference $\psi_u - \psi_t = 1$. Furthermore, this shows that

$$\psi_s - \psi_t = (\mathbf{1}_s - \mathbf{1}_t)^\top \boldsymbol{L}^+ (\mathbf{1}_u - \mathbf{1}_t) \cdot \frac{1}{R_{eff}(u,t)}\,,$$

which concludes the proof. □

Now we are ready to prove the main statement.

*Proof of Lemma 4.4.4.* Note that the demand can be decomposed as $\boldsymbol{d} - \boldsymbol{\pi}^C(\boldsymbol{d}) = \boldsymbol{d}^1 - \boldsymbol{d}^2$, where $\boldsymbol{d}^1 = \frac{\mathbf{1}_s}{\sqrt{r_{st}}} - \boldsymbol{\pi}^C(\frac{\mathbf{1}_s}{\sqrt{r_{st}}})$ and $\boldsymbol{d}^2 = \frac{\mathbf{1}_t}{\sqrt{r_{st}}} - \boldsymbol{\pi}^C(\frac{\mathbf{1}_t}{\sqrt{r_{st}}})$. Now let $p^1$ be the probability distribution of $s - C$ random walks obtained via a random walk from $s$ with transition probabilities proportional to inverse resistances. Similarly, let $p^2$ be the probability distribution of $t - C$ random walks obtained by running the same process starting from $t$.

Now, it is well known that an electrical flow is the sum of these random walks, i.e.

$$\boldsymbol{R}^{-1}\boldsymbol{B}\boldsymbol{L}^+\boldsymbol{d}^1 = \frac{1}{\sqrt{r_{st}}} \cdot \mathbb{E}_{P \sim p^1}\left[net(P)\right]$$

and similarly for $\boldsymbol{d}^2$

$$\boldsymbol{R}^{-1}\boldsymbol{B}\boldsymbol{L}^+\boldsymbol{d}^2 = \frac{1}{\sqrt{r_{st}}} \cdot \mathbb{E}_{P \sim p^2}\left[net(P)\right]\,,$$

where $net(P) \in \mathbb{R}^m$ is a flow vector whose $e$-th entry is the net number of times the edge $e = (u,v)$

is used by $P$. Therefore we can write:

$$\left|\frac{\phi_u - \phi_v}{\sqrt{r_{uv}}}\right| = \sqrt{r_{uv}}\left|\boldsymbol{R}^{-1}\boldsymbol{B}\boldsymbol{L}^{+}\boldsymbol{d}\right|_{uv}$$
$$= \sqrt{\frac{r_{uv}}{r_{st}}}\left|\mathbb{E}_{P^1\sim p^1}\left[net_e(P^1)\right] - \mathbb{E}_{P^2\sim p^2}\left[net_e(P^2)\right]\right| .$$

Let us also subdivide $e$ by inserting an additional vertex $w$ in the middle (i.e. $r_{uw} = r_{wv} = r_{uv}/2$). This has no effect in the random walks, but will be slightly more convenient in terms of notation. The first expectation term can be expressed as

$$\mathbb{E}_{P^1\sim p^1}\left[net_e(P^1)\right] = \Pr_{P^1\sim p^1}\left[P^1 \text{ visits } t \text{ before } C\cup\{w\}\right]\cdot\mathbb{E}_{P^1\sim p^1}\left[net_e(P^1) \mid P^1 \text{ visits } t \text{ before } C\cup\{w\}\right]$$
$$+ \Pr_{P^1\sim p^1}\left[P^1 \text{ visits } w \text{ before } C\cup\{t\}\right]\cdot\mathbb{E}_{P^1\sim p^1}\left[net_e(P^1) \mid P^1 \text{ visits } t \text{ before } C\cup\{t\}\right] .$$

Now, note that

$$\mathbb{E}_{P^1\sim p^1}\left[net_e(P^1) \mid P^1 \text{ visits } t \text{ before } C\cup\{w\}\right] = \mathbb{E}_{P^2\sim p^2}\left[net_e(P^2)\right] .$$

Additionally,

$$\Pr_{P^1\sim p^1}\left[P^1 \text{ visits } w \text{ before } C\cup\{t\}\right]$$
$$= \Pr_{P^1\sim p^1}\left[P^1 \text{ visits } w \text{ before } C\right]\cdot\Pr_{P^1\sim p^1}\left[P^1 \text{ visits } w \text{ before } t \mid P^1 \text{ visits } w \text{ before } C\right]$$

The first term of the product is $p_w^{C\cup\{w\}}(s)$. For the second term, we define a new graph $\widehat{G}$ by deleting $C$, and denote the hitting probabilities in $\widehat{G}$ by $\widehat{p}$. Then, the second term is equal to $\widehat{p}_w^{\{t,w\}}(s)$.

We have concluded that

$$\mathbb{E}_{P^1\sim p^1}\left[net_e(P^1)\right] \le \mathbb{E}_{P^2\sim p^2}\left[net_e(P^2)\right] + p_w^{C\cup\{w\}}(s)\cdot\widehat{p}_w^{\{t,w\}}(s) .$$

Combining this with the symmetric argument for $p^2$ shows that

$$\left|\mathbb{E}_{P^1\sim p^1}\left[net_e(P^1)\right] - \mathbb{E}_{P^2\sim p^2}\left[net_e(P^2)\right]\right| \le p_w^{C\cup\{w\}}(s)\cdot\widehat{p}_w^{\{t,w\}}(s) + p_w^{C\cup\{w\}}(t)\cdot\widehat{p}_w^{\{s,w\}}(t) .$$

Using Lemma 9.2.11 and the fact that $\widehat{R}_{eff}(w,t) \ge r_{uv}/4$ ($\widehat{R}_{eff}$ are the effective resistances in $\widehat{G}$), we can bound

$$\widehat{p}_w^{\{t,w\}}(s) \le \min\{1, \frac{r_{st}}{\widehat{R}_{eff}(w,t)}\} \le \min\{1, 4\frac{r_{st}}{r_{uv}}\} ,$$

and the same upper bound holds for $\widehat{p}_w^{\{s,w\}}(t)$. Therefore,

$$\left|\mathbb{E}_{P^1\sim p^1}\left[net_e(P^1)\right] - \mathbb{E}_{P^2\sim p^2}\left[net_e(P^2)\right]\right|$$
$$\le \min\{1, 4\frac{r_{st}}{r_{uv}}\}\left(p_w^{C\cup\{w\}}(s) + p_w^{C\cup\{w\}}(t)\right)$$
$$\le 2\sqrt{\frac{r_{st}}{r_{uv}}}\left(p_w^{C\cup\{w\}}(s) + p_w^{C\cup\{w\}}(t)\right)$$
$$\le 2\sqrt{\frac{r_{st}}{r_{uv}}}\left(p_u^{C\cup\{u\}}(s) + p_v^{C\cup\{v\}}(s) + p_u^{C\cup\{u\}}(t) + p_v^{C\cup\{v\}}(t)\right) .$$

Putting everything together, we have that

$$\left|\frac{\phi_u - \phi_v}{\sqrt{r_{uv}}}\right| \leq 2\left(p_u^{C\cup\{u\}}(s) + p_v^{C\cup\{v\}}(s) + p_u^{C\cup\{u\}}(t) + p_v^{C\cup\{v\}}(t)\right).$$

□

**Proof of Lemma 4.4.5**

*Proof.* Let $\boldsymbol{d} = \boldsymbol{B}^\top \frac{\mathbf{1}_e}{\sqrt{\boldsymbol{r}}}$ and $e = (u, w)$. Note that $\mathcal{E}_r(\boldsymbol{d}) \leq r_e \cdot (\frac{1}{\sqrt{r_e}})^2 = 1$, therefore the case that remains is

$$R_{eff}(C, e) \geq 36 \cdot r_e\,. \tag{9.45}$$

For each $v \in C$ by Lemma 4.4.6 we have that

$$\left|\pi_v^C(\boldsymbol{d})\right| \leq (p_v^C(u) + p_v^C(w)) \cdot \frac{\sqrt{r_e}}{R_{eff}(v, e)}\,.$$

Now, we would like to bound the energy of routing $\boldsymbol{\pi}^C(\boldsymbol{d})$ by the energy to route it via $w$. For each $v \in C$ we let $\boldsymbol{d}^v$ be the following demand:

$$\boldsymbol{d}^v = \pi_v^C(\boldsymbol{d}) \cdot (\mathbf{1}_v - \mathbf{1}_w)\,.$$

Note that $\boldsymbol{\pi}^C(\boldsymbol{d}) = \sum_{v\in C} \boldsymbol{d}^v$. We have,

$$\begin{aligned}
\sqrt{\mathcal{E}_{\boldsymbol{r}}(\pi^C(\boldsymbol{d}))} &= \sqrt{\mathcal{E}_{\boldsymbol{r}}\left(\pi^C\left(\sum_{v\in C}\boldsymbol{d}^v\right)\right)} \\
&\leq \sum_{v\in C}\sqrt{\mathcal{E}_{\boldsymbol{r}}\left(\pi^C\left(\boldsymbol{d}^v\right)\right)} \\
&\leq \sum_{v\in C}(R_{eff}(v, w))^{1/2}\left|\pi_v^C(\boldsymbol{d})\right| \\
&\leq \sum_{v\in C}(R_{eff}(v, w))^{1/2} \cdot (p_v^C(u) + p_v^C(w)) \cdot \frac{\sqrt{r_e}}{R_{eff}(v, e)}\,.
\end{aligned}$$

Now, note that, because $R_{eff}$ is a metric,

$$\begin{aligned}
R_{eff}(v, u) &\geq R_{eff}(v, w) - |R_{eff}(v, w) - R_{eff}(v, u)| \\
&\geq R_{eff}(v, w) - r_e \\
&\geq R_{eff}(v, w) - \frac{1}{36}R_{eff}(C, e) \\
&\geq \frac{35}{36}R_{eff}(v, w)\,,
\end{aligned}$$

where we used the triangle inequality twice, (9.45), and also the fact that $R_{eff}(v,w) \geq R_{eff}(C,e)$ because $v \in C$ and $w \in e$. Now, note also that

$$R_{eff}(v,e) \geq \frac{1}{2}\min\{R_{eff}(v,w), R_{eff}(v,u) \geq \frac{1}{2}\cdot\frac{35}{36}R_{eff}(v,w)\}\,,$$

so

$$\begin{aligned}
&2\sum_v (R_{eff}(v,w))^{1/2}\cdot(p_v^C(u)+p_v^C(w))\cdot\frac{\sqrt{r_e}}{R_{eff}(v,e)}\\
&\leq 2\sqrt{2\cdot\frac{36}{35}}\sum_v (p_v^C(u)+p_v^C(w))\cdot\sqrt{\frac{r_e}{R_{eff}(v,e)}}\\
&\leq 6\cdot\sqrt{\frac{r_e}{R_{eff}(C,e)}}\,,
\end{aligned}$$

where we used the fact that $\sum_{v\in C}(p_v^C(u)+p_v^C(w)) = 2$ and $R_{eff}(v,e) \geq R_{eff}(C,e)$ because $v \in C$. □

**Proof of Lemma 4.4.9**

*Proof.* By definition of the fact that $e$ is not $\varepsilon$-important, we have

$$R_{eff}(C,e) > r_e/\varepsilon^2\,.$$

Using the fact that the demand $\boldsymbol{\pi}^C\left(\boldsymbol{B}^\top\frac{\boldsymbol{p}}{\sqrt{\boldsymbol{r}}}\right)$ is supported on $C$ and Lemma 4.4.5, we get

$$\begin{aligned}
\left|\left\langle \mathbf{1}_e, \boldsymbol{R}^{-1/2}\boldsymbol{B}\boldsymbol{\phi}^*\right\rangle\right| &= \left|\left\langle \boldsymbol{\pi}^C\left(\boldsymbol{B}^\top\frac{\mathbf{1}_e}{\sqrt{\boldsymbol{r}}}\right), \boldsymbol{\phi}_C^*\right\rangle\right|\\
&\leq \sqrt{\mathcal{E}_{\boldsymbol{r}}\left(\boldsymbol{\pi}^C\left(\boldsymbol{B}^\top\frac{\mathbf{1}_e}{\sqrt{\boldsymbol{r}}}\right)\right)}\cdot\sqrt{E_{\boldsymbol{r}}(\boldsymbol{\phi}^*)}\\
&= \sqrt{\mathcal{E}_{\boldsymbol{r}}\left(\boldsymbol{\pi}^C\left(\boldsymbol{B}^\top\frac{\mathbf{1}_e}{\sqrt{\boldsymbol{r}}}\right)\right)}\cdot\delta\sqrt{\mathcal{E}_{\boldsymbol{r}}\left(\boldsymbol{\pi}^C\left(\boldsymbol{B}^\top\frac{\boldsymbol{p}}{\sqrt{\boldsymbol{r}}}\right)\right)}\\
&\leq 6\sqrt{\frac{r_e}{R_{eff}(C,e)}}\cdot\delta\sqrt{m}\\
&\leq 6\varepsilon\,.
\end{aligned}$$

□

**Proof of Lemma 4.4.11**

*Proof.* We note that

$$\begin{aligned}
&\sqrt{\mathcal{E}_{\boldsymbol{r}}\left(\boldsymbol{\pi}^{C\cup\{v\}}(\boldsymbol{B}^\top\frac{\boldsymbol{q}}{\sqrt{\boldsymbol{r}}})-\boldsymbol{\pi}^{C}(\boldsymbol{B}^\top\frac{\boldsymbol{q}}{\sqrt{\boldsymbol{r}}})\right)}\\
&=\left|\pi_v^{C\cup\{v\}}(\boldsymbol{B}^\top\frac{\boldsymbol{q}}{\sqrt{\boldsymbol{r}}})\right|\sqrt{R_{eff}(C,v)}\\
&\leq\sum_{e=(u,w)\in E}\left|\pi_v^{C\cup\{v\}}(\boldsymbol{B}^\top\frac{q_e}{\sqrt{r_e}}\mathbf{1}_e)\right|\sqrt{R_{eff}(C,v)}\\
&\leq\sum_{e=(u,w)\in E}(p_v^{C\cup\{v\}}(u)+p_v^{C\cup\{v\}}(w))\cdot\min\left\{\sqrt{\frac{R_{eff}(C,v)}{r_e}},\frac{\sqrt{r_eR_{eff}(C,v)}}{R_{eff}(e,v)}\right\},
\end{aligned}$$

where we used Lemma 4.4.6. For some sufficiently large $c$ to be defined later, we partition $E$ into $X$ and $Y$, where $X=\{e\in E\mid R_{eff}(C,v)\leq c^2\cdot r_e \text{ or } r_eR_{eff}(C,v)\leq c^2\cdot(R_{eff}(e,v))^2\}$ and $Y=E\backslash X$. We first note that

$$\begin{aligned}
&\sum_{e=(u,w)\in X}(p_v^{C\cup\{v\}}(u)+p_v^{C\cup\{v\}}(w))\cdot\min\left\{\sqrt{\frac{R_{eff}(C,v)}{r_e}},\frac{\sqrt{r_eR_{eff}(C,v)}}{R_{eff}(e,v)}\right\}\\
&\leq c\cdot\sum_{e=(u,w)\in X}(p_v^{C\cup\{v\}}(u)+p_v^{C\cup\{v\}}(w))\\
&\leq c\cdot\widetilde{O}\left(\beta^{-2}\right),
\end{aligned}$$

where the last inequality follows by the congestion reduction property.

Now, let $e=(u,w)\in Y$. We will prove that both $u$ and $w$ are much closer to $v$ than $C$. This, in turn, will imply that their hitting probabilities on $v$ are roughly the same, and so they mostly cancel out in the projection.

First of all, we let $R_{eff}(C,v)=c_1^2\cdot r_e$ and $r_eR_{eff}(C,v)=c_2^2\cdot(R_{eff}(e,v))^2$, for some $c_1,c_2>0$, where by definition $c_1,c_2\geq c$. Now, we assume without loss of generality that $R_{eff}(u,v)\leq R_{eff}(w,v)$, and so

$$R_{eff}(u,v)\leq 2R_{eff}(e,v)=\frac{2}{c_2}\sqrt{r_eR_{eff}(C,v)}=\frac{2}{c_1c_2}R_{eff}(C,v)\leq\frac{2}{c_1c_2}\left(R_{eff}(C,u)+R_{eff}(u,v)\right),$$

so

$$R_{eff}(u,v)\leq\frac{\frac{2}{c_1c_2}}{1-\frac{2}{c_1c_2}}R_{eff}(C,u)=\frac{2}{c_1c_2-2}R_{eff}(C,u)\leq\frac{3}{c_1c_2}R_{eff}(C,u).\tag{9.46}$$

Futhermore, note that

$$R_{eff}(w,v)\leq R_{eff}(u,v)+r_e\leq\left(\frac{2}{c_1c_2}+\frac{1}{c_1^2}\right)R_{eff}(C,v)\leq\left(\frac{2}{c_1c_2}+\frac{1}{c_1^2}\right)\left(R_{eff}(C,w)+R_{eff}(w,v)\right),$$

and so we have

$$R_{eff}(w,v)\leq\frac{\frac{2}{c_1c_2}+\frac{1}{c_1^2}}{1-\frac{2}{c_1c_2}-\frac{1}{c_1^2}}R_{eff}(C,w)\leq 3\left(\frac{1}{c_1c_2}+\frac{1}{c_1^2}\right)R_{eff}(C,w).\tag{9.47}$$

Now, by Lemma 9.2.11 together with (9.46) we have

$$p_v^{C\cup\{v\}}(u) \geq 1 - \frac{3}{c_1 c_2}$$

and with (9.47) we have

$$p_v^{C\cup\{v\}}(w) \geq 1 - 3\left(\frac{1}{c_1 c_2} + \frac{1}{c_1^2}\right),$$

therefore

$$\left|p_v^{C\cup\{v\}}(u) - p_v^{C\cup\{v\}}(w)\right| \leq 6\left(\frac{1}{c_1 c_2} + \frac{1}{c_1^2}\right).$$

So,

$$\begin{aligned}
\left|\pi_v^{C\cup\{v\}}(\boldsymbol{B}^\top \frac{q_e}{\sqrt{r_e}}\mathbf{1}_e)\right| \sqrt{R_{eff}(C,v)} &= \left|p_v^{C\cup\{v\}}(u) - p_v^{C\cup\{v\}}(w)\right| \sqrt{\frac{R_{eff}(C,v)}{r_e}} \\
&\leq 6\left(\frac{1}{c_1 c_2} + \frac{1}{c_1^2}\right)\cdot c_1 \\
&= 6\left(\frac{1}{c_2} + \frac{1}{c_1}\right) \\
&= O\left(\frac{1}{c}\right).
\end{aligned}$$

Now, we will apply Lemma 4.4.10 to prove that with high probability $|Y| \leq \widetilde{O}\left(\beta^{-1}\right)$. The reason we can apply the lemma is that for any $e = (u,w) \in Y$, we have

$$R_{eff}(e,v) = \frac{1}{c_1 c_2} R_{eff}(C,v) \leq \frac{1}{2} R_{eff}(C,v),$$

and so $Y \subseteq N_E(v, R_{eff}(C,v)/2)$. Therefore, we get that

$$\sum_{e=(u,w)\in Y} \left|\pi_v^{C\cup\{v\}}(\boldsymbol{B}^\top \frac{q_e}{\sqrt{r_e}}\mathbf{1}_e)\right| \sqrt{R_{eff}(C,v)} \leq c^{-1}\cdot \widetilde{O}\left(\beta^{-1}\right).$$

Overall, we conclude that

$$\sqrt{\mathcal{E}_{\boldsymbol{r}}\left(\boldsymbol{\pi}^{C\cup\{v\}}(\boldsymbol{B}^\top \frac{\boldsymbol{q}}{\sqrt{\boldsymbol{r}}}) - \boldsymbol{\pi}^{C}(\boldsymbol{B}^\top \frac{\boldsymbol{q}}{\sqrt{\boldsymbol{r}}})\right)} \leq (c + c^{-1})\widetilde{O}\left(\beta^{-2}\right) = \widetilde{O}\left(\beta^{-2}\right),$$

by setting $c$ to be a large enough constant. □

**Proof of Lemma 4.4.12**

*Proof.* We write

$$\begin{aligned}
&\boldsymbol{\pi}^{C^T,\boldsymbol{r}^T}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}^T}{\sqrt{\boldsymbol{r}^T}}\right)-\boldsymbol{\pi}^{C^0,\boldsymbol{r}^0}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}^0}{\sqrt{\boldsymbol{r}^0}}\right)\\
&=\underbrace{\sum_{i \text{ is an } \textsc{AddTerminal}}\pi_{v^i}^{C^{i+1},\boldsymbol{r}^i}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}^i}{\sqrt{\boldsymbol{r}^i}}\right)\cdot\left(\mathbf{1}_{v^i}-\boldsymbol{\pi}^{C^i,\boldsymbol{r}^i}\left(\mathbf{1}_{v^i}\right)\right)}_{\boldsymbol{d}_{Add}}\\
&+\underbrace{\sum_{i \text{ is an } \textsc{Update}}\boldsymbol{\pi}^{C^i,\boldsymbol{r}^i}\left(\boldsymbol{B}^\top\left(\frac{\boldsymbol{q}^{i+1}}{\sqrt{\boldsymbol{r}^{i+1}}}-\frac{\boldsymbol{q}^i}{\sqrt{\boldsymbol{r}^i}}\right)\mathbf{1}_{e^i}\right)}_{\boldsymbol{d}_{Upd}},
\end{aligned}$$

which implies that

$$\sqrt{\mathcal{E}_{\boldsymbol{r}^T}\left(\boldsymbol{\pi}^{C^T,\boldsymbol{r}^T}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}^T}{\sqrt{\boldsymbol{r}^T}}\right)-\boldsymbol{\pi}^{C^0,\boldsymbol{r}^0}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}^0}{\sqrt{\boldsymbol{r}^0}}\right)\right)}\le\sqrt{\mathcal{E}_{\boldsymbol{r}^T}(\boldsymbol{d}_{Add})}+\sqrt{\mathcal{E}_{\boldsymbol{r}^T}(\boldsymbol{d}_{Upd})}\,.$$

We bound each of these terms separately. For the second one, we have that

$$\begin{aligned}
&\sqrt{\mathcal{E}_{\boldsymbol{r}^T}\left(\boldsymbol{d}_{Upd}\right)}\\
&\le\sum_{i \text{ is an } \textsc{Update}}\sqrt{\mathcal{E}_{\boldsymbol{r}^T}\left(\boldsymbol{\pi}^{C^i,\boldsymbol{r}^i}\left(\boldsymbol{B}^\top\left(\frac{\boldsymbol{q}^{i+1}}{\sqrt{\boldsymbol{r}^{i+1}}}-\frac{\boldsymbol{q}^i}{\sqrt{\boldsymbol{r}^i}}\right)\mathbf{1}_{e^i}\right)\right)}\\
&=\sum_{i \text{ is an } \textsc{Update}}\sqrt{\mathcal{E}_{\boldsymbol{r}^T}\left(\boldsymbol{B}^\top\left(\frac{\boldsymbol{q}^{i+1}}{\sqrt{\boldsymbol{r}^{i+1}}}-\frac{\boldsymbol{q}^i}{\sqrt{\boldsymbol{r}^i}}\right)\mathbf{1}_{e^i}\right)}\\
&\le\sum_{i \text{ is an } \textsc{Update}}\left(\sqrt{\mathcal{E}_{\boldsymbol{r}^T}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}^{i+1}}{\sqrt{\boldsymbol{r}^{i+1}}}\mathbf{1}_{e^i}\right)}+\sqrt{\mathcal{E}_{\boldsymbol{r}^T}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}^i}{\sqrt{\boldsymbol{r}^i}}\mathbf{1}_{e^i}\right)}\right)\\
&\le\max_i\left\|\frac{\boldsymbol{r}^T}{\boldsymbol{r}^i}\right\|_\infty^{1/2}\sum_{i \text{ is an } \textsc{Update}}\left(\sqrt{\mathcal{E}_{\boldsymbol{r}^{i+1}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}^{i+1}}{\sqrt{\boldsymbol{r}^{i+1}}}\mathbf{1}_{e^i}\right)}+\sqrt{\mathcal{E}_{\boldsymbol{r}^i}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}^i}{\sqrt{\boldsymbol{r}^i}}\mathbf{1}_{e^i}\right)}\right)\\
&\le 2\max_i\left\|\frac{\boldsymbol{r}^T}{\boldsymbol{r}^i}\right\|_\infty^{1/2}T\,.
\end{aligned}$$

For the first one, we have

$$\begin{aligned}
&\sqrt{\mathcal{E}_{\boldsymbol{r}^T}\left(\boldsymbol{d}_{Add}\right)}\\
&\le\sum_{i \text{ is an } \textsc{AddTerminal}}\left|\pi_{v^i}^{C^{i+1},\boldsymbol{r}^i}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}^i}{\sqrt{\boldsymbol{r}^i}}\right)\right|\cdot\sqrt{\mathcal{E}_{\boldsymbol{r}^T}\left(\mathbf{1}_{v^i}-\boldsymbol{\pi}^{C^i,\boldsymbol{r}^i}\left(\mathbf{1}_{v^i}\right)\right)}\\
&\le\left\|\frac{\boldsymbol{r}^T}{\boldsymbol{r}^i}\right\|_\infty^{1/2}\sum_{i \text{ is an } \textsc{AddTerminal}}\left|\pi_{v^i}^{C^{i+1},\boldsymbol{r}^i}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}^i}{\sqrt{\boldsymbol{r}^i}}\right)\right|\cdot\sqrt{\mathcal{E}_{\boldsymbol{r}^i}\left(\mathbf{1}_{v^i}-\boldsymbol{\pi}^{C^i,\boldsymbol{r}^i}\left(\mathbf{1}_{v^i}\right)\right)}\\
&\le\widetilde{O}\left(\max_i\left\|\frac{\boldsymbol{r}^T}{\boldsymbol{r}^i}\right\|_\infty^{1/2}\beta^{-2}\right)\cdot T\,,
\end{aligned}$$

where in the last inequality we used Lemma 4.4.11. The desired statement now follows immediately. □

**Proof of Lemma 4.3.8**

*Proof.* **INITIALIZE**($\boldsymbol{f}$): We set $\boldsymbol{s}^+ = \boldsymbol{u} - \boldsymbol{f}$, $\boldsymbol{s}^- = \boldsymbol{f}$, $\boldsymbol{r}^0 = \boldsymbol{r} = \frac{1}{(\boldsymbol{s}^+)^2} + \frac{1}{(\boldsymbol{s}^-)^2}$.

We first initialize a $\beta$-congestion reduction subset $C$ based on Lemma 4.4.2, which takes time $\widetilde{O}\left(m\beta^{-2}\right)$, and a data structure DYNAMICSC for maintaining the sparsified Schur Complement onto $C$, as described in Appendix 9.2.1, which takes time $\widetilde{O}\left(m\beta^{-4}\varepsilon^{-4}\right)$. We also set $C^0 = C$.

Then, we generate an $\widetilde{O}\left(\varepsilon^{-2}\right) \times m$ sketching matrix $\boldsymbol{Q}$ as in (Lemma 5.1, [73] v2), which takes time $\widetilde{O}\left(m\varepsilon^{-2}\right)$, and let its rows be $\boldsymbol{q}^i$ for $i \in [\widetilde{O}\left(\varepsilon^{-2}\right)]$.

In order to compute the set of important edges, we use Lemma 9.2.3 after contracting $C$, which shows that we can compute all resistances of the form $R_{eff}(C, u)$ for $u \in V \backslash C$ up to a factor of 2 in $\widetilde{O}(m)$. From these, we can get 4-approximate estimates of $R_{eff}(C, e)$ for $e \in E \backslash E(C)$, using the fact that

$$\min\{R_{eff}(C, u), R_{eff}(C, w)\} \approx_2 R_{eff}(C, e)\,.$$

Then, in $O(m)$ time, we can easily compute a set of edges $S$ such that

$$\{e \mid e \text{ is } \frac{\varepsilon\beta}{\alpha}\text{-important } \} \subseteq S \subseteq \{e \mid e \text{ is } \frac{\varepsilon\beta}{4\alpha}\text{-important}\}$$

We also need to sample the random walks that will be used inside the demand projection data structures. We use (Lemma 5.15, [73] v2) to sample $h = \widetilde{O}\left(\widehat{\varepsilon}^{-4}\beta^{-6} + \widehat{\varepsilon}^{-2}\beta^{-2}\gamma^{-2}\right)$ random walks for each $u \in V \backslash C$ and $e \in E \backslash E(C)$ with $u \in e$, where we set $\gamma = \frac{\varepsilon}{4\alpha}$ so that $S$ is a subset of $\gamma$-important edges. Note that, by Definition 4.3.7, a $\gamma$-important edge will always remain $\gamma$-important until the LOCATOR is re-initialized, as any edge's resistive distance to $C$ can only decrease, and its own resistance is constant. Therefore $S$ can be assumed to always be a subset of $\gamma$-important edges.

The runtime to sample the set $\mathcal{P}$ of these random walks is

$$\widetilde{O}\left(mh\beta^{-2}\right) = \widetilde{O}\left(m(\widehat{\varepsilon}^{-4}\beta^{-8} + \widehat{\varepsilon}^{-2}\beta^{-4}\gamma^{-2})\right) = \widetilde{O}\left(m(\widehat{\varepsilon}^{-4}\beta^{-8} + \widehat{\varepsilon}^{-2}\varepsilon^{-2}\alpha^2\beta^{-4})\right)\,.$$

In order to be able to detect congested edges, we will initialize $\widetilde{O}\left(\varepsilon^{-2}\right)$ demand projection data structures, with the guarantees from Lemma 4.4.14. We will maintain an approximation to $\boldsymbol{\pi}^C(\boldsymbol{B}^\top \frac{\boldsymbol{q}^i_S}{\sqrt{\boldsymbol{r}}})$ for all $i \in \widetilde{O}\left(\varepsilon^{-2}\right)$, where $\boldsymbol{q}^i$ are the rows of the sketching matrix that we have generated, as well as $\boldsymbol{\pi}^{old} := \boldsymbol{\pi}^{C^0, \boldsymbol{r}^0}\left(\boldsymbol{B}^\top \frac{\boldsymbol{p}^0}{\sqrt{\boldsymbol{r}^0}}\right)$, where $\boldsymbol{p}^0 = \sqrt{\boldsymbol{r}^0} g(\boldsymbol{s}^0) = \frac{\frac{1}{\boldsymbol{s}^{+,0}} - \frac{1}{\boldsymbol{s}^{-,0}}}{\sqrt{\boldsymbol{r}^0}}$.

Specifically, we call

$$\text{DP}^i.\text{INITIALIZE}(C, \boldsymbol{r}, \boldsymbol{q}^i, S, \mathcal{P})$$

for all $i \in [\widetilde{O}\left(\varepsilon^{-2}\right)]$, and also exactly compute $\boldsymbol{\pi}^{old}$, which can be done by calling

$$\text{DEMANDPROJECTOR.INITIALIZE}(C, \boldsymbol{r}, \boldsymbol{p}, [m], \mathcal{P})\,.$$

The total runtime for this operation is dominated by the random walk generation, and is

$$\widetilde{O}\left(m(\widehat{\varepsilon}^{-4}\beta^{-8} + \widehat{\varepsilon}^{-2}\varepsilon^{-2}\alpha^2\beta^{-4})\right)\,.$$

**UPDATE**($e, \boldsymbol{f}$): We set $s_e^+ = u_e - f_e$, $s_e^- = f_e$, and $r_e = \frac{1}{(s_e^+)^2} + \frac{1}{(s_e^-)^2}$. Then, we also set $p_e = \frac{\frac{1}{s_e^+} - \frac{1}{s_e^-}}{\sqrt{r_e}}$.

We distinguish two cases:

- $e \in E(C)$:

  In this case, we can simply call

  $$\text{DynamicSC.Update}(e, r_e)$$

  and

  $$\text{DP}^i.\text{Update}(e, \boldsymbol{r}, \boldsymbol{q})$$

  for all $i \in [\widetilde{O}\left(\varepsilon^{-2}\right)]$. Note that we can do this as $\text{DP}^i$ was initialized with resistances $\boldsymbol{r}^0$ and $\boldsymbol{r}^0 \approx_\alpha \boldsymbol{r}$.

- $e \in E \backslash E(C)$:

  We let $e = (u, w)$. We want to insert $u$ and $w$ into $C$, but for doing that $\text{DP}^i$'s require constant factor estimates of the resistances $R_{eff}(C, u)$ and $R_{eff}(C \cup \{u\}, w)$. In order to get these estimates, we will use DynamicSC.

  We first call

  $$\text{DynamicSC.AddTerminal}(u)\,,$$

  which takes time $\widetilde{O}\left(\beta^{-2}\varepsilon^{-2}\right)$ and returns $\widetilde{R}_{eff}(C, u) \approx_2 R_{eff}(C, u)$. Given this estimate, we can call

  $$\text{DP}^i.\text{AddTerminal}(u, \widetilde{R}_{eff}(C, u))\,,$$

  for all $i \in [\widetilde{O}\left(\varepsilon^{-2}\right)]$, each of which takes time

  $$\widetilde{O}\left(\widehat{\varepsilon}^{-4}\beta^{-8} + \widehat{\varepsilon}^{-2}\beta^{-6}\gamma^{-2}\right) = \widetilde{O}\left(\widehat{\varepsilon}^{-4}\beta^{-8} + \widehat{\varepsilon}^{-2}\varepsilon^{-2}\alpha^2\beta^{-6}\right)\,.$$

  Now, we can set $C = C \cup \{u\}$ and repeat the same process for $w$.

  Finally, to update the resistance, note that we now have $e \in E(C)$, so we apply the procedure from the first case.

Finally, if the total number of calls to $\text{DP}^i$.AddTerminal for some fixed $i$ since the last call to $\mathcal{L}$.BatchUpdate($\emptyset$) exceeds $\frac{\varepsilon}{\widehat{\varepsilon}\alpha^{1/2}}$ (note that the number of calls is actually the same for all $i$), we call $\mathcal{L}$.BatchUpdate($\emptyset$) in order to re-initialize the demand projections.

We conclude that the total runtime is

$$\widetilde{O}\left(m\frac{\widehat{\varepsilon}\alpha^{1/2}}{\varepsilon^3} + \widehat{\varepsilon}^{-4}\varepsilon^{-2}\beta^{-8} + \widehat{\varepsilon}^{-2}\varepsilon^{-4}\alpha^2\beta^{-6}\right)\,,$$

where the first term comes from amortizing the calls to $\mathcal{L}$.BatchUpdate($\emptyset$), each of which, as we will see, takes $\widetilde{O}\left(m\varepsilon^{-2}\right)$.

**BatchUpdate**$(Z, \boldsymbol{f})$: First, for each $e \in Z$, we set $s_e^+ = u_e - f_e$, $s_e^- = f_e$, $r_e^0 = r_e = \frac{1}{(s_e^+)^2} + \frac{1}{(s_e^-)^2}$, and $p_e^0 = p_e = \frac{\frac{1}{s_e^+} - \frac{1}{s_e^-}}{\sqrt{r_e}}$.

For each $e = (u, w) \in Z$, we call

$$\text{DynamicSC.AddTerminal}(u)$$

and

$$\text{DynamicSC.AddTerminal}(w)$$

(if $u$ and $w$ are not already in $C$), and then we call

$$\textsc{DynamicSC.Update}(e, r_e)\,.$$

Then, we set $C^0 = C = C \cup (\cup_{(u,w)\in Z}\{u, w\})$. Additionally, we re-compute $\boldsymbol{\pi}^{old}$ based on the new values of $C^0, \boldsymbol{r}^0, \boldsymbol{p}^0$. All of this takes time $\widetilde{O}\left(m + |Z|\beta^{-2}\varepsilon^{-2}\right)$.

Now, to pass these updates to the DemandProjectors, we first have to re-compute the set of important edges $S$ (with the newly updated resistances) as any set such that

$$\{e \mid e \text{ is } \frac{\varepsilon}{\alpha}\text{-important } \} \subseteq S \subseteq \{e \mid e \text{ is } \frac{\varepsilon}{4\alpha}\text{-important}\}\,.$$

As we have already argued, this takes $\widetilde{O}\left(m\right)$.

Now, finally, we re-initialize all the DemandProjectors by calling

$$\text{DP}^i.\textsc{Initialize}(C, \boldsymbol{r}, \boldsymbol{q}, S, \mathcal{P})\,.$$

for all $i \in [\widetilde{O}\left(\varepsilon^{-2}\right)]$, where each call takes $\widetilde{O}\left(m\right)$.

We conclude with a total runtime of

$$\widetilde{O}\left(m\varepsilon^{-2} + |Z|\beta^{-2}\varepsilon^{-2}\right)\,.$$

**Solve**(): This operation performs the main task of the locator, which is to detect congested edges. We will do that by using the approximate demand projections that we have been maintaining.

We remind that the congestion vector we are trying to approximate to $O(\varepsilon)$ additive accuracy is

$$\boldsymbol{\rho}^* = \delta\sqrt{\boldsymbol{r}}g(\boldsymbol{s}) - \delta\boldsymbol{R}^{-1/2}\boldsymbol{B}\boldsymbol{L}^{+}\boldsymbol{B}^\top g(\boldsymbol{s})\,.$$

We will first reduce the problem of finding the entries of $\boldsymbol{\rho}^*$ with magnitude $\geq \Omega(\varepsilon)$, to the problem of computing an $O(\varepsilon)$-additive approximation to

$$v_i^* = \delta \cdot \left\langle \boldsymbol{\pi}^C\left(\boldsymbol{B}^\top \frac{\boldsymbol{q}_S^i}{\sqrt{\boldsymbol{r}}}\right), \widetilde{SC}^{+}\boldsymbol{\pi}^{C^0,\boldsymbol{r}^0}\left(\boldsymbol{B}^\top \frac{\boldsymbol{p}^0}{\sqrt{\boldsymbol{r}^0}}\right)\right\rangle$$

for all $i \in [\widetilde{O}\left(\varepsilon^{-2}\right)]$, where $\widetilde{SC}$ is the approximate Schur complement maintained in DynamicSC. Then, we will see how to approximate $v_i^*$ to additive accuracy $O(\varepsilon)$ using the demand projection data structures.

First of all, note that, by definition of $g(\boldsymbol{s}) = \frac{\frac{1}{s^+} - \frac{1}{s^-}}{\boldsymbol{r}}$,

$$\left\|\delta\sqrt{\boldsymbol{r}}g(\boldsymbol{s})\right\|_\infty \leq \delta \leq \varepsilon\,,$$

so this term can be ignored. Using Lemma 4.4.3, we get that

$$\delta\left\|\boldsymbol{R}^{-1/2}\boldsymbol{B}\boldsymbol{L}^{+}(\boldsymbol{B}^\top g(\boldsymbol{s}) - \boldsymbol{\pi}^C(\boldsymbol{B}^\top g(\boldsymbol{s})))\right\|_\infty \leq \delta\cdot\widetilde{O}\left(\beta^{-2}\right) \leq \varepsilon/2\,.$$

This means that the entries of the vector

$$\delta\boldsymbol{R}^{-1/2}\boldsymbol{B}\boldsymbol{L}^{+}\boldsymbol{\pi}^C(\boldsymbol{B}^\top g(\boldsymbol{s}))$$

that have magnitude $\leq \varepsilon$ do not correspond to the $\Omega(\varepsilon)$-congested edges that we are looking for.

Now, we set $T = |C\backslash C^0|$, where $C^0$ was the congestion reduction subset during the last call to

BatchUpdate, and apply Lemma 4.4.12. This shows that

$$\sqrt{\mathcal{E}_{\boldsymbol{r}}\left(\boldsymbol{\pi}^{old}-\boldsymbol{\pi}^{C}\left(\boldsymbol{B}^{\top}g(\boldsymbol{s})\right)\right)}\leq\widetilde{O}\left(\alpha^{1/2}\beta^{-2}\right)\cdot T\,.$$

Therefore, if we define

$$\boldsymbol{\rho}=-\delta\boldsymbol{R}^{-1/2}\boldsymbol{B}\boldsymbol{L}^{+}\boldsymbol{\pi}^{old}\,,$$

we conclude that

$$\|\boldsymbol{\rho}-\boldsymbol{\rho}^{*}\|_{\infty}\leq O(\varepsilon)+\delta\left\|\boldsymbol{R}^{-1/2}\boldsymbol{B}\boldsymbol{L}^{+}\left(\boldsymbol{\pi}^{old}-\boldsymbol{\pi}^{C}\left(\boldsymbol{B}^{\top}g(\boldsymbol{s})\right)\right)\right\|_{\infty}\leq O(\varepsilon)+\delta T\cdot\widetilde{O}\left(\alpha^{1/2}\beta^{-2}\right)\leq O(\varepsilon)\,,$$

where we used the fact that $T=\frac{\varepsilon}{\widehat{\varepsilon}\alpha^{1/2}}\leq\frac{\varepsilon}{\delta\beta^{-2}\alpha^{1/2}}$. Therefore it suffices to estimate $\boldsymbol{\rho}$ up to $O(\varepsilon)$-additive accuracy.

Now, note that, by definition, no edge $e\in E\backslash S$ is $\varepsilon/\alpha$-important with respect to $\boldsymbol{r}^{0}$ and $C^{0}$. By using Lemma 4.4.5, for each such edge we get

$$\begin{aligned}
&\delta\left|\boldsymbol{R}^{-1/2}\boldsymbol{B}\boldsymbol{L}^{+}\boldsymbol{\pi}^{old}\right|_{e}\\
&\leq\delta\sqrt{\mathcal{E}_{\boldsymbol{r}}\left(\boldsymbol{\pi}^{C^{0}}\left(\boldsymbol{B}^{\top}\frac{\mathbf{1}_{e}}{\sqrt{\boldsymbol{r}}}\right)\right)}\sqrt{\mathcal{E}_{\boldsymbol{r}}(\boldsymbol{\pi}^{old})}\\
&\leq\delta\alpha\sqrt{\mathcal{E}_{\boldsymbol{r}^{0}}\left(\boldsymbol{\pi}^{C^{0}}\left(\boldsymbol{B}^{\top}\frac{\mathbf{1}_{e}}{\sqrt{\boldsymbol{r}}}\right)\right)}\sqrt{\mathcal{E}_{\boldsymbol{r}^{0}}(\boldsymbol{\pi}^{old})}\\
&\leq\delta\alpha\cdot\frac{\varepsilon}{\alpha}\cdot O(\sqrt{m})\\
&=O(\varepsilon)\,,
\end{aligned}$$

where we also used the fact that $\mathcal{E}_{\boldsymbol{r}^{0}}(\boldsymbol{\pi}^{C^{0},\boldsymbol{r}^{0}}(g(\boldsymbol{s}^{0})))\leq O(\mathcal{E}_{\boldsymbol{r}^{0}}(g(\boldsymbol{s})))$. Therefore it suffices to approximate

$$\boldsymbol{\rho}'=\delta\boldsymbol{I}_{S}\boldsymbol{R}^{-1/2}\boldsymbol{B}\boldsymbol{L}^{+}\boldsymbol{\pi}^{old}\,.$$

Note that here we can replace $\boldsymbol{L}^{+}$ by $\begin{pmatrix}-\boldsymbol{L}_{FF}^{-1}\boldsymbol{L}_{FC}\\ \boldsymbol{I}\end{pmatrix}\widetilde{SC}^{+}$ where $\widetilde{SC}\approx_{1+\varepsilon}SC$ and only lose another additive $\varepsilon$ error, as

$$\begin{aligned}
&\delta\left|\left\langle\mathbf{1}_{e},\boldsymbol{R}^{-1/2}\boldsymbol{B}\boldsymbol{L}^{+}\boldsymbol{\pi}^{old}\right\rangle-\delta\left\langle\mathbf{1}_{e},\boldsymbol{R}^{-1/2}\boldsymbol{B}\begin{pmatrix}-\boldsymbol{L}_{FF}^{-1}\boldsymbol{L}_{FC}\\ \boldsymbol{I}\end{pmatrix}\widetilde{SC}^{+}\boldsymbol{\pi}^{old}\right\rangle\right|\\
&=\delta\left|\left\langle\boldsymbol{\pi}^{C}\left(\boldsymbol{B}^{\top}\frac{\mathbf{1}_{e}}{\sqrt{\boldsymbol{r}}}\right),\left(SC^{+}-\widetilde{SC}^{+}\right)\boldsymbol{\pi}^{old}\right\rangle\right|\\
&\leq O(\delta\varepsilon\cdot\sqrt{m})\\
&\leq O(\varepsilon)\,,
\end{aligned}$$

where we used the fact that

$$(1-\varepsilon)SC\preceq\widetilde{SC}\preceq(1+\varepsilon)SC\Rightarrow-O(\varepsilon)\widetilde{SC}^{+}\preceq SC^{+}-\widetilde{SC}^{+}\preceq O(\varepsilon)\widetilde{SC}^{+}\,.$$

and that

$$\begin{aligned}\sqrt{\mathcal{E}_{\boldsymbol{r}}\left(\boldsymbol{\pi}^{old}\right)} &\leq \sqrt{\mathcal{E}_{\boldsymbol{r}}\left(\boldsymbol{\pi}^{C}\left(\boldsymbol{B}^{\top}g(\boldsymbol{s})\right)\right)} + \sqrt{\mathcal{E}_{\boldsymbol{r}}\left(\boldsymbol{\pi}^{old} - \boldsymbol{\pi}^{C}\left(\boldsymbol{B}^{\top}g(\boldsymbol{s})\right)\right)} \\ &\leq O(m) + \widetilde{O}\left(\alpha^{1/2}\beta^{-2}\right)\cdot T \\ &\leq O(m)\,,\end{aligned}$$

where we used the fact that $T = \frac{\varepsilon}{\widehat{\varepsilon}\alpha^{1/2}} \leq \frac{\varepsilon}{\delta\beta^{-2}\alpha^{1/2}} \leq \frac{\sqrt{m}}{\beta^{-2}\alpha^{1/2}}$.

Now, we will use the sketching lemma (Lemma 5.1, [73] v2), which shows that in order to find all entries of

$$\boldsymbol{I}_S\boldsymbol{R}^{-1/2}\boldsymbol{B}\begin{pmatrix}-\boldsymbol{L}_{FF}\boldsymbol{L}_{FC}\\ \boldsymbol{I}\end{pmatrix}\widetilde{SC}^{+}\boldsymbol{\pi}^{old}$$

with magnitude $\Omega(\varepsilon)$, it suffices to compute the inner products

$$\begin{aligned}&\delta\left\langle \boldsymbol{B}^{\top}\frac{\boldsymbol{q}_S^i}{\sqrt{\boldsymbol{r}}}, \begin{pmatrix}-\boldsymbol{L}_{FF}^{-1}\boldsymbol{L}_{FC}\\ \boldsymbol{I}\end{pmatrix}\widetilde{SC}^{+}\boldsymbol{\pi}^{old}\right\rangle \\ &= \delta\left\langle \boldsymbol{\pi}^{C}\left(\boldsymbol{B}^{\top}\frac{\boldsymbol{q}_S^i}{\sqrt{\boldsymbol{r}}}\right), \widetilde{SC}^{+}\boldsymbol{\pi}^{old}\right\rangle\end{aligned}$$

for $i \in [\widetilde{O}\left(\varepsilon^{-2}\right)]$, up to additive accuracy

$$\varepsilon\cdot\left\|\delta\boldsymbol{I}_S\boldsymbol{R}^{-1/2}\boldsymbol{B}\begin{pmatrix}-\boldsymbol{L}_{FF}^{-1}\boldsymbol{L}_{FC}\\ \boldsymbol{I}\end{pmatrix}\widetilde{SC}^{+}\boldsymbol{\pi}^{old}\right\|_2^{-1} \geq \Omega(\varepsilon)\,,$$

where we used the fact that

$$\begin{aligned}&\left\|\delta\boldsymbol{I}_S\boldsymbol{R}^{-1/2}\boldsymbol{B}\begin{pmatrix}-\boldsymbol{L}_{FF}^{-1}\boldsymbol{L}_{FC}\\ \boldsymbol{I}\end{pmatrix}\widetilde{SC}^{+}\boldsymbol{\pi}^{old}\right\|_2^2 \\ &= \delta^2\langle\widetilde{SC}^{+}\boldsymbol{\pi}^{old}, \begin{pmatrix}-\boldsymbol{L}_{CF}\boldsymbol{L}_{FF}^{-1} & \boldsymbol{I}\end{pmatrix}\boldsymbol{L}\begin{pmatrix}-\boldsymbol{L}_{FF}^{-1}\boldsymbol{L}_{FC}\\ \boldsymbol{I}\end{pmatrix}\widetilde{SC}^{+}\boldsymbol{\pi}^{old}\rangle \\ &= \delta^2\left\langle\widetilde{SC}^{+}\boldsymbol{\pi}^{old}, \begin{pmatrix}-\boldsymbol{L}_{CF}\boldsymbol{L}_{FF}^{-1} & \boldsymbol{I}\end{pmatrix}\begin{pmatrix}\boldsymbol{L}_{FF} & \boldsymbol{L}_{FC}\\ \boldsymbol{L}_{CF} & \boldsymbol{L}_{CC}\end{pmatrix}\begin{pmatrix}-\boldsymbol{L}_{FF}^{-1}\boldsymbol{L}_{FC}\\ \boldsymbol{I}\end{pmatrix}\widetilde{SC}^{+}\boldsymbol{\pi}^{old}\right\rangle \\ &= \delta^2\left\langle\widetilde{SC}^{+}\boldsymbol{\pi}^{old}, SC\widetilde{SC}^{+}\boldsymbol{\pi}^{old}\right\rangle \\ &\leq 2\delta^2\left\langle\boldsymbol{\pi}^{old}, \widetilde{SC}^{+}\boldsymbol{\pi}^{old}\right\rangle \\ &\leq O(\delta^2 m) \\ &= O(1)\,.\end{aligned}$$

Now, for the second part of the proof, we would like to compute $\boldsymbol{v}$ such that $\|\boldsymbol{v} - \boldsymbol{v}^*\|_\infty \leq O(\varepsilon)$, where we remind that

$$\boldsymbol{v}^* = \left\langle \boldsymbol{\pi}^{C}\left(\boldsymbol{B}^{\top}\frac{\boldsymbol{q}_S^i}{\sqrt{\boldsymbol{r}}}\right), \widetilde{SC}^{+}\boldsymbol{\pi}^{old}\right\rangle.$$

Note that we already have estimates $\widetilde{\pi}^{C}\left(\boldsymbol{B}^{\top}\frac{\boldsymbol{q}_S^i}{\sqrt{\boldsymbol{r}}}\right)$ given by $\mathrm{DP}^i$ for all $i \in [\widetilde{O}\left(\varepsilon^{-2}\right)]$. We obtain these estimates by calling

$$\mathrm{DP}^i.\textsc{Output}()$$

each of which takes time $O(\beta m)$. By the guarantees of Definition 4.4.13, with high probability we have

$$\delta \left| \left\langle \widetilde{\pi}^C \left( \boldsymbol{B}^\top \frac{\boldsymbol{q}_S^i}{\sqrt{\boldsymbol{r}}} \right) - \boldsymbol{\pi}^C \left( \boldsymbol{B}^\top \frac{\boldsymbol{q}_S^i}{\sqrt{\boldsymbol{r}}} \right), \widetilde{SC}^+ \boldsymbol{\pi}^{old} \right\rangle \right| \leq \widehat{\varepsilon}\sqrt{\alpha} T \,,$$

where we used the fact that

$$E_{\boldsymbol{r}}(\delta \cdot \widetilde{SC}^+ \boldsymbol{\pi}^{old}) \leq O(1) \,.$$

Now, since by definition BatchUpdate is called every $\frac{\varepsilon}{\widehat{\varepsilon}\alpha^{1/2}}$ calls to Update, We have $T \leq \frac{\varepsilon}{\widehat{\varepsilon}\alpha^{1/2}}$ and so

$$\delta \left| \left\langle \widetilde{\pi}^C \left( \boldsymbol{B}^\top \frac{\boldsymbol{q}_S^i}{\sqrt{\boldsymbol{r}}} \right) - \boldsymbol{\pi}^C \left( \boldsymbol{B}^\top \frac{\boldsymbol{q}_S^i}{\sqrt{\boldsymbol{r}}} \right), \widetilde{SC}^+ \boldsymbol{\pi}^{old} \right\rangle \right| \leq \varepsilon \,.$$

This means that, running the algorithm from (Lemma 5.1, [73] v2), we can obtain an edge set of size $\widetilde{O}\left(\varepsilon^{-2}\right)$ that contains all edges such that $|\rho_e^*| \geq c \cdot \varepsilon$ for some constant $c > 0$. By rescaling $\varepsilon$ to get the right constant, we obtain all edges such that $|\rho_e^*| \geq \varepsilon/2$ with high probability. The runtime is dominated by the time to get $\widetilde{SC}$ and apply its inverse, and is $\widetilde{O}\left(\beta m \varepsilon^{-2}\right)$.

**Success probability** We will argue that $\mathcal{L}$ uses DynamicSC and the $\mathrm{DP}^i$ as an oblivious adversary. First of all, note that no randomness is injected into the inputs of DynamicSC, as they are all coming form the inputs of $\mathcal{L}$.

Regarding $\mathrm{DP}^i$, note that its only output is given by the call to $\mathrm{DP}^i$.Output. However, note that its output is only used to estimate the inner product

$$\left\langle \boldsymbol{\pi}^C \left( \boldsymbol{B}^\top \frac{\boldsymbol{q}_S^i}{\sqrt{\boldsymbol{r}}} \right), \widetilde{SC}^+ \boldsymbol{\pi}_{old} \right\rangle \,,$$

from which we obtain the set of congested edges and we directly return it from $\mathcal{L}$. Thus, it does not influence the state of $\mathcal{L}$, DynamicSC or any future inputs.

□

### 9.2.5 Deferred Proofs from Section 4.5

**Proof of Lemma 4.4.6**

*Proof.* Let $\mathcal{P}_v(u)$ be a random walk that starts from $u$ and stops when it hits $v$.

$$\begin{aligned} p_v^{C \cup \{v,w\}}(u) &= \Pr\left[\mathcal{P}_v(u) \cap C = \emptyset \text{ and } w \notin \mathcal{P}_v(u)\right] \\ &= \Pr\left[\mathcal{P}_v(u) \cap C = \emptyset\right] \cdot \Pr\left[w \notin \mathcal{P}_v(u) \mid \mathcal{P}_v(u) \cap C = \emptyset\right] \\ &= p_v^{C \cup \{v\}}(u) \cdot \Pr\left[w \notin \mathcal{P}_v(u) \mid \mathcal{P}_v(u) \cap C = \emptyset\right] \end{aligned}$$

Consider new resistances $\widehat{r}$, where $\widehat{r}_e = r_e$ for all $e \in E$ not incident to $C$ and $\widehat{r}_e = \infty$ for all $e \in E$ incident to $C$. Also, let $\widehat{p}$ be the hitting probability function for these new resistances. It is easy to see that

$$\Pr\left[w \notin \mathcal{P}_v(u) \mid \mathcal{P}_v(u) \cap C = \emptyset\right] = \widehat{p}_v^{\{v,w\}}(u) \,.$$

Therefore, we have

$$p_v^{C \cup \{v,w\}}(u) = p_v^{C \cup \{v\}}(u) \cdot \widehat{p}_v^{\{v,w\}}(u) \,.$$

Now we will bound $\widehat{p}_v^{\{v,w\}}(u)$. Let $\psi$ be electrical potentials for pushing 1 unit of flow from $v$ to $w$ with resistances $\widehat{r}$ and let $f$ be the associated electrical flow. We have that

$$|\psi_u - \psi_w| = |f_e|\widehat{r}_e \leq \widehat{r}_e = r_e \tag{9.48}$$

(because $|f_e| \leq 1$ and $e$ is not incident to $C$) and

$$|\psi_v - \psi_w| = \widehat{R}_{eff}(v,w) \geq R_{eff}(v,w) \tag{9.49}$$

Additionally, by well known facts that connect electrical potential embeddings with random walks, we have that

$$\psi_u = \psi_w + \widehat{p}_v^{\{v,w\}}(u)(\psi_v - \psi_w)\,,$$

or equivalently

$$\widehat{p}_v^{\{v,w\}}(u) = \frac{\psi_u - \psi_w}{\psi_v - \psi_w}\,.$$

Using (9.48) and (9.49), this immediately implies that

$$\widehat{p}_v^{\{v,w\}}(u) \leq \frac{r_e}{R_{eff}(v,w)}\,.$$

So we have proved that

$$p_v^{C\cup\{v,w\}}(u) \leq p_v^{C\cup\{v\}}(u)\frac{r_e}{R_{eff}(v,w)}$$

and symmetrically

$$p_v^{C\cup\{v,u\}}(w) \leq p_v^{C\cup\{v\}}(w)\frac{r_e}{R_{eff}(v,u)}\,.$$

Now, let's look at $\pi_v^{C\cup\{v\}}(B^\top \mathbf{1}_e) = p_v^{C\cup\{v\}}(u) - p_v^{C\cup\{v\}}(w)$. Note that

$$p_v^{C\cup\{v\}}(u) = p_v^{C\cup\{v,w\}}(u) + p_w^{C\cup\{v,w\}}(u)p_v^{C\cup\{v\}}(w)$$

which we re-write as

$$p_v^{C\cup\{v\}}(u) - p_v^{C\cup\{v\}}(w) = p_v^{C\cup\{v,w\}}(u) - (1 - p_w^{C\cup\{v,w\}}(u))p_v^{C\cup\{v\}}(w) \leq p_v^{C\cup\{v,w\}}(u)\,.$$

Symmetrically,

$$p_v^{C\cup\{v\}}(w) - p_v^{C\cup\{v\}}(u) \leq p_v^{C\cup\{v,u\}}(w)\,.$$

From these we conclude that

$$\begin{aligned}
\left|\pi_v^{C\cup\{v\}}(B^\top \mathbf{1}_e)\right| &= \left|p_v^{C\cup\{v\}}(u) - p_v^{C\cup\{v\}}(w)\right| \\
&\leq \max\left\{p_v^{C\cup\{v,w\}}(u), p_v^{C\cup\{v,u\}}(w)\right\} \\
&\leq \max\left\{p_v^{C\cup\{v\}}(u)\cdot\frac{r_e}{R_{eff}(v,w)}, p_v^{C\cup\{v\}}(w)\cdot\frac{r_e}{R_{eff}(v,u)}\right\} \\
&\leq (p_v^{C\cup\{v\}}(u) + p_v^{C\cup\{v\}}(w))\cdot\max\left\{\frac{r_e}{R_{eff}(v,w)}, \frac{r_e}{R_{eff}(v,u)}\right\}\,,
\end{aligned}$$

which, after dividing by $\sqrt{r_e}$ gives

$$\left|\pi_v^{C\cup\{v\}}(B^\top \mathbf{1}_e)\right| \leq (p_v^{C\cup\{v\}}(u) + p_v^{C\cup\{v\}}(w)) \cdot \max\left\{\frac{\sqrt{r_e}}{R_{eff}(v,w)}, \frac{\sqrt{r_e}}{R_{eff}(v,u)}\right\}.$$

□

**Proof of Lemma 4.5.3**

*Proof.* For each $u \in V\backslash C$ and $e \in S'$ with $u \in e$, we generate $Z$ random walks $P^1(u), \ldots, P^Z(u)$ from $u$ to $C \cup \{v\}$. We set

$$\begin{aligned}\widetilde{\pi}_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top \frac{\boldsymbol{q}_{S'}}{\sqrt{\boldsymbol{r}}}\right) &= \sum_{e=(u,w)\in S'} \sum_{z=1}^{Z} \frac{1}{Z}\frac{q_e}{\sqrt{r_e}}\left(1_{\{v\in P^z(u)\}} - 1_{\{v\in P^z(w)\}}\right) \\ &= \sum_{e=(u,w)\in S'} \sum_{z=1}^{Z} (X_{e,u,z} - X_{e,w,z}),\end{aligned}$$

where we have set $X_{e,u,z} = \frac{1}{Z}\frac{q_e}{\sqrt{r_e}} 1_{\{v\in P^z(u)\}}$ and $X_{e,w,z} = -\frac{1}{Z}\frac{q_e}{\sqrt{r_e}} 1_{\{v\in P^z(w)\}}$.

Note that $\mathop{\mathbb{E}}_{P^z(u)}[X_{e,u,z}] = \frac{1}{Z}\frac{q_e}{\sqrt{r_e}} p_v^{C\cup\{v\}}(u)$ and $\mathop{\mathbb{E}}_{P^z(w)}[X_{e,w,z}] = -\frac{1}{Z}\frac{q_e}{\sqrt{r_e}} p_v^{C\cup\{v\}}(w)$. This implies that our estimate is unbiased, as

$$\mathbb{E}\left[\widetilde{\pi}_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top \frac{\boldsymbol{q}_{S'}}{\sqrt{\boldsymbol{r}}}\right)\right] = \sum_{e=(u,w)\in S'} \frac{q_e}{\sqrt{r_e}}(p_v^{C\cup\{v\}}(u) - p_v^{C\cup\{v\}}(w)) = \pi_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top \frac{\boldsymbol{q}_{S'}}{\sqrt{\boldsymbol{r}}}\right).$$

We now need to show that our estimate is concentrated around the mean. To apply the concentration bound in Lemma 4.5.2, we need the following bounds:

$$\sum_{e=(u,w)\in S'} \sum_{z=1}^{Z} (|\mathbb{E}[X_{e,u,z}]| + |\mathbb{E}[X_{e,w,z}]|) = \sum_{e=(u,w)\in S'} \frac{|q_e|}{\sqrt{r_e}}(p_v^{C\cup\{v\}}(u) + p_v^{C\cup\{v\}}(w)) := E$$

$$\max_{\substack{e=(u,w)\in S' \\ z\in[Z]}} \max\{|X_{e,u,z}|, |X_{e,w,z}|\} \leq \max_{e\in S'} \frac{1}{Z\sqrt{r_e}} := M.$$

So now for any $t \in [0, E]$ we have

$$\begin{aligned}
&\Pr\left[\left|\widetilde{\pi}_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top \frac{\boldsymbol{q}_{S'}}{\sqrt{\boldsymbol{r}}}\right) - \pi_v^{C\cup\{v\}}(\boldsymbol{B}^\top \frac{\boldsymbol{q}_{S'}}{\sqrt{\boldsymbol{r}}})\right| > t\right] \\
&\leq 2\exp\left(-\frac{t^2}{6EM}\right) \\
&= 2\exp\left(-\frac{Zt^2}{6\sum_{e=(u,w)\in S'} \frac{|q_e|}{\sqrt{r_e}}(p_v^{C\cup\{v\}}(u) + p_v^{C\cup\{v\}}(w)) \max_{e\in S'} \frac{1}{\sqrt{r_e}}}\right) \\
&\leq 2\exp\left(-\frac{Zt^2}{6\sum_{e=(u,w)\in S'} (p_v^{C\cup\{v\}}(u) + p_v^{C\cup\{v\}}(w)) \max_{e\in S'} \frac{1}{r_e}}\right) \\
&\leq 2\exp\left(-\frac{Zt^2c^2R_{eff}(C,v)}{6\sum_{e=(u,w)\in S'} (p_v^{C\cup\{v\}}(u) + p_v^{C\cup\{v\}}(w))}\right) \\
&\leq 2\exp\left(-Zt^2c^2R_{eff}(C,v)/\widetilde{O}\left(\beta^{-2}\right)\right) \\
&\leq \frac{1}{n^{100}},
\end{aligned}$$

where the last inequality follows by setting $Z = \widetilde{O}\left(\frac{\log n \log \frac{1}{\beta}}{\delta_1'^2}\right)$ and $t = \frac{\delta_1'}{\beta c\sqrt{R_{eff}(C,v)}}$. Note that we have used the fact that $R_{eff}(C,v) \leq r_e/c^2$ for all $e \in S'$, as well as the congestion reduction property (Definition 4.4.1)

$$\sum_{e=(u,w)\in E\setminus E(C)} (p_v^{C\cup\{v\}}(u) + p_v^{C\cup\{v\}}(w)) \leq \widetilde{O}\left(1/\beta^2\right).$$

□

**Proof of Lemma 4.5.4**

*Proof.* In order to compute $\widetilde{\pi}_v^{C\cup\{v\}}(\boldsymbol{B}^\top \frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}})$ we use Lemma 4.5.3 with demand $\boldsymbol{B}^\top \frac{\boldsymbol{q}_{S'}}{\sqrt{\boldsymbol{r}}}$ and error parameter $\delta_1' > 0$, where

$$\{e \in S \mid R_{eff}(C,v) \leq r_e/(2c^2)\} \subseteq S' \subseteq \{e \in S \mid R_{eff}(C,v) \leq r_e/c^2\},$$

and $c > 0$ will be defined later. Note that such a set $S'$ can be trivially computed given our effective resistance estimate $\widetilde{R}_{eff}(C,v) \approx_2 R_{eff}(C,v)$. However, algorithmically we do not directly compute $S'$, but instead find its intersection with the edges from which a sampled random walk ends up at $v$. (Using the congestion reduction property of $C$, this can be done in $\widetilde{O}\left(\delta_1'^{-2}\beta^{-2}\log n \log \frac{1}{\beta}\right)$ time just by going through all random walks that contain $v$.)

Now, Lemma 4.5.3 guarantees that

$$\left|\widetilde{\pi}_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}_{S'}}{\sqrt{\boldsymbol{r}}}\right)-\pi_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}_{S'}}{\sqrt{\boldsymbol{r}}}\right)\right|\leq\frac{\delta_1'}{\beta c\sqrt{R_{eff}(C,v)}}$$

given access to $O(\delta'^{-2}\log n\log\frac{1}{\beta})$ random walks for each $u\in V\backslash C$, $e\in S'$ with $u\in e$.

Then, we set $\widetilde{\pi}_v^{C\cup\{v\}}(\boldsymbol{B}^\top\frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}):=\widetilde{\pi}_v^{C\cup\{v\}}(\boldsymbol{B}^\top\frac{\boldsymbol{q}_{S'}}{\sqrt{\boldsymbol{r}}})$, and we have that

$$\begin{aligned}&\left|\widetilde{\pi}_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right)-\pi_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right)\right|\\&\leq\left|\widetilde{\pi}_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}_{S'}}{\sqrt{\boldsymbol{r}}}\right)-\pi_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}_{S'}}{\sqrt{\boldsymbol{r}}}\right)\right|+\left|\pi_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}_S-\boldsymbol{q}_{S'}}{\sqrt{\boldsymbol{r}}}\right)\right|\\&\leq\frac{\delta_1'}{\beta c\sqrt{R_{eff}(C,v)}}+\left|\pi_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}_S-\boldsymbol{q}_{S'}}{\sqrt{\boldsymbol{r}}}\right)\right|.\end{aligned}\tag{9.50}$$

Now, to bound the second term, we use Lemma 4.4.6, which gives

$$\left|\pi_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}_S-\boldsymbol{q}_{S'}}{\sqrt{\boldsymbol{r}}}\right)\right|\leq\sum_{e=(u,w)\in S\backslash S'}\left(p_v^{C\cup\{v\}}(u)+p_v^{C\cup\{v\}}(w)\right)\frac{\sqrt{r_e}}{R_{eff}(v,e)}.$$

Now, note that for each $e\in S\backslash S'$, $e$ is close to $C$, but $v$ is far from $C$, so $R_{eff}(v,e)$ should be large. Specifically, by Lemma 4.2.8 we have $R_{eff}(v,e)\geq\frac{1}{2}\min\{R_{eff}(v,u),R_{eff}(v,w)\}$, and by the triangle inequality

$$\min\{R_{eff}(v,u),R_{eff}(v,w)\}\geq R_{eff}(C,v)-\max\{R_{eff}(C,u),R_{eff}(C,w)\}\geq R_{eff}(C,v)-2R_{eff}(C,e).$$

By the fact that $e$ is $\gamma$-important and that

$$e\notin S'\supseteq\{e\in S\mid R_{eff}(C,v)\leq r_e/(2c^2)\},$$

we have $R_{eff}(C,e)\leq r_e/\gamma^2\leq 2c^2R_{eff}(C,v)/\gamma^2$, so

$$\begin{aligned}\frac{\sqrt{r_e}}{R_{eff}(v,e)}&\leq\frac{1}{1/2-2c^2/\gamma^2}\frac{\sqrt{r_e}}{R_{eff}(C,v)}\\&\leq\frac{c}{1/2-2c^2/\gamma^2}\frac{1}{\sqrt{R_{eff}(C,v)}}.\end{aligned}$$

By using the congestion reduction property (Definition 4.4.1), we obtain

$$\left|\pi_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}_S-\boldsymbol{q}_{S'}}{\sqrt{\boldsymbol{r}}}\right)\right|\leq\frac{c}{1/2-2c^2/\gamma^2}\frac{1}{\sqrt{R_{eff}(C,v)}}\widetilde{O}\left(\frac{1}{\beta^2}\right).\tag{9.51}$$

Setting $c=\min\{\delta_1/\widetilde{O}\left(\beta^{-2}\right),\gamma/4\}$ and $\delta_1'=\beta c\cdot\delta_1/2$, (9.50) becomes

$$\left|\widetilde{\pi}_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right)-\pi_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right)\right|\leq\frac{\delta_1}{\sqrt{R_{eff}(C,v)}}.$$

Also, the number of random walks needed for each valid pair $(u, e)$ is

$$\widetilde{O}\left(\delta_1'^{-2}\log n\log\frac{1}{\beta}\right)=\widetilde{O}\left(\delta_1^{-2}\beta^{-2}c^{-2}\log n\log\frac{1}{\beta}\right)=\widetilde{O}\left(\left(\delta_1^{-4}\beta^{-6}+\delta_1^{-2}\beta^{-2}\gamma^{-2}\right)\log n\log\frac{1}{\beta}\right)$$

.

For the last part of the lemma, we let $S''=\{e\in S\mid R_{eff}(C,v)\le r_e/\left(\gamma/4\right)^2\}$ and write

$$\begin{aligned}&\left|\pi_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right)\right|\\&\le\left|\pi_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}_{S''}}{\sqrt{\boldsymbol{r}}}\right)\right|+\left|\pi_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}_S-\boldsymbol{q}_{S''}}{\sqrt{\boldsymbol{r}}}\right)\right|.\end{aligned}$$

For the first term,

$$\begin{aligned}\left|\pi_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}_{S''}}{\sqrt{\boldsymbol{r}}}\right)\right|&\le\sum_{e=(u,w)\in S''}\left(p_v^{C\cup\{v\}}(u)+p_v^{C\cup\{v\}}(w)\right)\frac{1}{\sqrt{r_e}}\\&\le\frac{1}{(\gamma/4)\sqrt{R_{eff}(C,v)}}\widetilde{O}\left(\frac{1}{\beta^2}\right),\end{aligned}$$

and for the second term we have already proved in (9.51) (after replacing $c$ by $\gamma/4$) that

$$\left|\pi_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}_S-\boldsymbol{q}_{S''}}{\sqrt{\boldsymbol{r}}}\right)\right|\le\frac{\gamma/4}{\sqrt{R_{eff}(C,v)}}\widetilde{O}\left(\frac{1}{\beta^2}\right).$$

Putting these together, we conclude that

$$\left|\pi_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right)\right|\le\frac{1}{\gamma\sqrt{R_{eff}(C,v)}}\cdot\widetilde{O}\left(\frac{1}{\beta^2}\right).$$

□

**Proof of Lemma 4.5.5**

*Proof.* For any $I\subseteq\mathbb{R}$, we define $F_I=\{i\in[n]\mid|\bar{\phi}_i|\in I\}$. For some $0<a<b$ to be defined later, we partition $[n]$ as

$$[n]=F_{I_0}\cup F_{I_1}\cup\cdots\cup F_{I_K}\cup F_{I_{K+1}},$$

where $I_0=[0,a)$, $I_{K+1}=[b,\infty)$, and $I_1,\ldots,I_K$ is a partition of $[a,b)$ into $K=O(\log\frac{b}{a})$ intervals such that for all $k\in[K]$ we have $\Phi_k:=\max_{i\in F_k}|\bar{\phi}_i|\le 2\cdot\min_{i\in F_k}|\bar{\phi}_i|$.

A union bound gives

$$\Pr\left[\left|\langle\widetilde{\pi}-\boldsymbol{\pi},\bar{\boldsymbol{\phi}}\rangle\right|>t\right]\le\sum_{k=0}^{K+1}\Pr\left[\left|\sum_{i\in I_k}(\widetilde{\pi}_i-\pi_i)\bar{\phi}_i\right|>t/(K+2)\right].$$

We first examine $I_0$ and $I_{K+1}$ separately. Note that

$$\left|\sum_{i\in F_{I_0}}(\widetilde{\pi}_i-\pi_i)\bar{\phi}_i\right|\le\|\widetilde{\pi}-\boldsymbol{\pi}\|_1 a\le 2a$$

and

$$\left|\sum_{i\in F_{I_{K+1}}}(\widetilde{\pi}_i-\pi_i)\bar{\phi}_i\right|\le\sum_{i\in F_{I_{K+1}}}\widetilde{\pi}_i\left|\bar{\phi}_i\right|+\sum_{i\in F_{I_{K+1}}}\pi_i\left|\bar{\phi}_i\right|\le\frac{1}{b}\sum_{i\in F_{I_{K+1}}}|\widetilde{\pi}_i-\pi_i|\bar{\phi}_i^2\le\frac{1}{b}\sum_{i\in F_{I_{K+1}}}\widetilde{\pi}_i|\bar{\phi}_i|+\frac{\left\|\bar{\phi}\right\|_{\boldsymbol{\pi},2}^2}{b}$$

But note that by picking $b\ge\max\left\{\frac{(K+2)Var_{\boldsymbol{\pi}}(\bar{\boldsymbol{\phi}})}{t},\sqrt{Var_{\boldsymbol{\pi}}(\bar{\boldsymbol{\phi}})\cdot n^{101}}\right\}$, we have $\frac{Var_{\boldsymbol{\pi}}(\bar{\phi})}{b}\le t/(K+2)$ and also for any $i\in F_{K+1}$ we have $\pi_i\le\frac{Var_{\boldsymbol{\pi}}(\bar{\boldsymbol{\phi}})}{b^2}\le\frac{1}{n^{101}}$. This means that $\Pr[\widetilde{\pi}_i\neq 0]\le\frac{1}{n^{101}}$, and so by union bound

$$\Pr\left[\sum_{i\in F_{I_{K+1}}}\widetilde{\pi}_i|\bar{\phi}_i|\neq 0\right]\le\frac{1}{n^{100}}\,.$$

Now, we proceed to $F_1,\ldots,F_K$. We draw $Z$ samples $x_1,\ldots,x_Z$ from $\boldsymbol{\pi}$. Then, we also define the following random variables for $z\in[Z]$ and $i\in[n]$:

$$X_{z,i}=\begin{cases}1 & \text{if } x_z=i\\ 0 & \text{otherwise}\end{cases}$$

for $i\in[n]$ and

$$Y_{z,k}=\frac{1}{Z}\sum_{i\in F_k}X_{z,i}\bar{\phi}_i$$

This allows us to write $\sum\limits_{i\in F_k}\widetilde{\pi}_i\bar{\phi}_i=\sum\limits_{z=1}^{Z}Y_{z,k}$.

Fix $k\in[K]$. We will apply Lemma 4.5.2 on the random variable $\sum\limits_{z=1}^{Z}Y_{z,k}$. We first compute

$$\sum_{z=1}^{Z}|\mathbb{E}[Y_{z,k}]|=\left|\sum_{i\in F_k}\pi_i\bar{\phi}_i\right|\le\sum_{i\in F_k}\pi_i|\bar{\phi}_i|:=E_k$$

and

$$\max_{z\in[Z]}|Y_{z,k}|\le\frac{\Phi_k}{Z}:=M_k\,.$$

Therefore we immediately have $E_k M_k \leq \frac{2}{Z} \sum_{i \in F_k} \pi_i \bar{\phi}_i^2 \leq \frac{2}{Z} \cdot Var_\pi(\bar{\phi})$. By Lemma 4.5.2,

$$\begin{aligned}
&\Pr\left[\left|\sum_{z=1}^{Z} Y_{z,k} - \mathbb{E}\left[\sum_{z=1}^{Z} Y_{z,k}\right]\right| > t/(K+2)\right] \\
&\leq 2\exp\left(-\frac{t^2}{6E_k M_k (K+2)^2}\right) \\
&\leq 2\exp\left(-\frac{Zt^2}{12 \cdot Var_\pi(\bar{\phi})(K+2)^2}\right).
\end{aligned}$$

Summarizing, and using the fact that $K = \widetilde{O}\left(\log(n \cdot Var_{\boldsymbol{\pi}}(\bar{\boldsymbol{\phi}})/t^2)\right)$, we get

$$\Pr\left[\left|\langle \widetilde{\pi} - \boldsymbol{\pi}, \bar{\boldsymbol{\phi}} \rangle\right| > t\right] \leq \widetilde{O}\left(\frac{1}{n^{100}}\right) + 2\widetilde{O}\left(\log\left(n \cdot Var_{\boldsymbol{\pi}}(\bar{\boldsymbol{\phi}})/t^2\right)\right) \exp\left(-\frac{Zt^2}{12 \cdot \widetilde{O}\left(Var_{\boldsymbol{\pi}}(\bar{\boldsymbol{\phi}}) \log^2 n\right)}\right).$$

□

**Proof of Lemma 4.5.6**

*Proof.* Let $S_0 = \emptyset$ and and for each $k \in \mathbb{N}$ let

$$S_k = \{i \in [n] \backslash S_{k-1} \ : \ \phi_i^2 \leq 2^{k+1} R_{eff}(C, v)\}.$$

Fix some $k \geq 2$. Note that $\phi_i^2 > 2^k R_{eff}(C, v)$ for all $i \in S_k$, implying $\frac{2^k R_{eff}(C,v)}{R_{eff}(S_k,v)} < E_{\boldsymbol{r}}(\boldsymbol{\phi}) \leq 1$, and so $R_{eff}(S_k, v) > 2^k R_{eff}(C, v) \geq 4R_{eff}(C, v)$. As

$$R_{eff}(C, v) \geq \frac{1}{4} \min\{R_{eff}(S_k, v), R_{eff}(C \backslash S_k, v)\} > \min\{R_{eff}(C, v), \frac{1}{4} R_{eff}(C \backslash S_k, v)\},$$

we have $R_{eff}(C \backslash S_k, v) < 4R_{eff}(C, v)$. This implies that $\left\| \pi_{S_k}^C(\mathbf{1}_v) \right\|_1 \leq \frac{R_{eff}(C \backslash S_k, v)}{R_{eff}(S_k, v)} < \frac{1}{2^{k-2}}$.

So we conclude that $Var_{\boldsymbol{\pi}}(\boldsymbol{\phi}) = \sum_{i \in S_k} \pi_i \phi_i^2 \leq \frac{1}{2^{k-2}} \cdot 2^{k+1} R_{eff}(C, v) = 8R_{eff}(C, v)$. □

**Proof of Lemma 4.5.1**

*Proof.* The first part of the statement is given by applying Lemma 4.5.4, and we see that it requires $\widetilde{O}\left(\delta_1^{-4}\beta^{-6} + \delta_1^{-2}\beta^{-2}\gamma^{-2}\right)$ random walks for each $u \in V \backslash C$ and $e \in E \backslash E(C)$ with $u \in e$.

For the second part we use the fact that the change in the demand projection after inserting $v$ into $C$ is given by

$$\boldsymbol{\pi}^{C \cup \{v\}}\left(\boldsymbol{B}^\top \frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right) - \boldsymbol{\pi}^C\left(\boldsymbol{B}^\top \frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right) = \pi_v^{C \cup \{v\}}\left(\boldsymbol{B}^\top \frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right) \cdot (\mathbf{1}_v - \boldsymbol{\pi}^C(\mathbf{1}_v)),$$

and therefore we can estimate this update via

$$\widetilde{\pi}_v^{C \cup \{v\}}\left(\boldsymbol{B}^\top \frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right) \cdot (\mathbf{1}_v - \widetilde{\pi}^C(\mathbf{1}_v)).$$

where $\widetilde{\pi}_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right)$ is the estimate we computed using Lemma 4.5.4 and $\widetilde{\pi}^C(\mathbf{1}_v)$ is obtained by applying Lemma 4.5.7.

Let us show that this estimation indeed introduces only a small amount of error. For any $\boldsymbol{\phi}$, such that $E_r(\boldsymbol{\phi}) \leq 1$, we can write

$$\begin{aligned}
&\left|\left\langle \widetilde{\pi}_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right)\cdot(\mathbf{1}_v-\widetilde{\pi}^C(\mathbf{1}_v))-\pi_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right)\cdot(\mathbf{1}_v-\boldsymbol{\pi}^C(\mathbf{1}_v)),\boldsymbol{\phi}\right\rangle\right| \\
&\leq\left|\left\langle \pi_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right)\cdot(\boldsymbol{\pi}^C(\mathbf{1}_v)-\widetilde{\pi}^C(\mathbf{1}_v)),\boldsymbol{\phi}\right\rangle\right| \\
&+\left|\left\langle \left(\widetilde{\pi}_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right)-\pi_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right)\right)\cdot(\mathbf{1}_v-\boldsymbol{\pi}^C(\mathbf{1}_v)),\boldsymbol{\phi}\right\rangle\right| \\
&+\left|\left\langle \left(\widetilde{\pi}_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right)-\pi_v^{C\cup\{v\}}\left(\boldsymbol{B}^\top\frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}\right)\right)\cdot(\boldsymbol{\pi}^C(\mathbf{1}_v)-\widetilde{\pi}^C(\mathbf{1}_v)),\boldsymbol{\phi}\right\rangle\right|.
\end{aligned}$$

At this point we can bound these quantities using Lemmas 4.5.4 and 4.5.7. It is important to notice that they require that $S$ is a set of $\gamma$-important edges, for some parameter $\gamma$. Our congestion reduction subset $C$ keeps increasing due to vertex insertions. This, however, means that effective resistances between any vertex in $V\setminus C$ and $C$ can only decrease, and therefore the set of important edges can only increase. Thus we are still in a valid position to apply these lemmas.

Using $E_{\boldsymbol{r}}(\boldsymbol{\phi}) \leq 1$, which allows us to write:

$$\langle \mathbf{1}_v-\boldsymbol{\pi}^C(\mathbf{1}_v)),\boldsymbol{\phi}\rangle \leq \mathcal{E}_{\boldsymbol{r}}\left(\mathbf{1}_v-\boldsymbol{\pi}^C(\mathbf{1}_v)\right)=R_{eff}(v,C)\,,$$

we can continue to upper bound the error by:

$$\begin{aligned}
&\frac{\widetilde{O}\left(\gamma^{-1}\beta^{-2}\right)}{\sqrt{R_{eff}(C,v)}}\cdot\delta_2\sqrt{R_{eff}(C,v)}+\frac{\delta_1}{\sqrt{R_{eff}(C,v)}}\cdot\sqrt{R_{eff}(C,v)}+\frac{\delta_1}{\sqrt{R_{eff}(C,v)}}\cdot\delta_2\sqrt{R_{eff}(C,v)} \\
&=\delta_2\cdot\widetilde{O}\left(\gamma^{-1}\beta^{-2}\right)+\delta_1+\delta_1\delta_2\,.
\end{aligned}$$

Setting $\delta_1=\widehat{\varepsilon}/2$ and $\delta_2=\widehat{\varepsilon}\beta^2\gamma/\widetilde{O}\left(1\right)$, we conclude that w.h.p. each operation introduces at most $\widehat{\varepsilon}$ additive error in the maintained estimate for $\left\langle\widetilde{\pi}^C(\boldsymbol{B}^\top\frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}}),\boldsymbol{\phi}\right\rangle$.

Per Lemma 4.5.4, estimating one coordinate of the demand projection requires

$$\widetilde{O}\left(\delta_1^{-4}\beta^{-6}+\delta_1^{-2}\beta^{-2}\gamma^{-2}\right)=\widetilde{O}\left(\widehat{\varepsilon}^{-4}\beta^{-6}+\widehat{\varepsilon}^{-2}\beta^{-2}\gamma^{-2}\right)$$

random walks, and estimating $\widetilde{\pi}^C(\boldsymbol{B}^\top\frac{\boldsymbol{q}_S}{\sqrt{\boldsymbol{r}}})$, per Lemma 4.5.7, requires

$$\widetilde{O}\left(\delta_2^{-2}\right)=\widetilde{O}\left(\widehat{\varepsilon}^{-2}\beta^{-4}\gamma^{-2}\right)$$

random walks. This concludes the proof.

□

### 9.2.6 The Checker Data Structure

**Theorem 9.2.12** (Theorem 3, [73]). *There is a* Checker *data structure supporting the following operations with the given runtimes against oblivious adversaries, for parameters* $0<\beta_{\textsc{Checker}},\varepsilon<1$ *such that* $\beta_{\textsc{Checker}}\geq\widetilde{\Omega}\left(\varepsilon^{-1/2}/m^{1/4}\right)$.

- INITIALIZE($\boldsymbol{f}, \varepsilon, \beta_{\text{CHECKER}}$): *Initializes the data structure with slacks* $\boldsymbol{s}^+ = \boldsymbol{u} - \boldsymbol{f}$, $\boldsymbol{s}^- = \boldsymbol{f}$, *and resistances* $\boldsymbol{r} = \frac{1}{(\boldsymbol{s}^+)^2} + \frac{1}{(\boldsymbol{s}^-)^2}$. *Runtime:* $\widetilde{O}\left(m\beta_{\text{CHECKER}}^{-4}\varepsilon^{-4}\right)$.

- UPDATE($e, \boldsymbol{f}'$): *Set* $s_e^+ = u_e - f'_e$, $s_e^- = f'_e$, *and* $r_e = \frac{1}{(s_e^+)^2} + \frac{1}{(s_e^-)^2}$. *Runtime: Amortized* $\widetilde{O}\left(\beta_{\text{CHECKER}}^{-2}\varepsilon^{-2}\right)$.

- TEMPORARYUPDATE($e, \boldsymbol{f}'$): *Set* $s_e^+ = u_e - f'_e$, $s_e^- = f'_e$, *and* $r_e = \frac{1}{(s_e^+)^2} + \frac{1}{(s_e^-)^2}$. *Runtime: Worst case* $\widetilde{O}\left((K\beta_{\text{CHECKER}}^{-2}\varepsilon^{-2})^2\right)$, *where* $K$ *is the number of* TEMPORARYUPDATE*s that have not been rolled back using* ROLLBACK. *All* TEMPORARYUPDATE*s should be rolled back before the next call to* UPDATE.

- ROLLBACK(): *Rolls back the last* TEMPORARYUPDATE *if it exists. The runtime is the same as the original operation.*

- CHECK($e, \boldsymbol{\pi}_{old}$): *Returns* $\widetilde{f}_e$ *such that* $\sqrt{r_e}\left|\widetilde{f}_e - \widetilde{f}_e^*\right| \leq \varepsilon$, *where*

$$\widetilde{\boldsymbol{f}}^* = \delta g(\boldsymbol{s}) - \delta \boldsymbol{R}^{-1}\boldsymbol{B}\left(\boldsymbol{B}^\top \boldsymbol{R}^{-1}\boldsymbol{B}\right)^+ \boldsymbol{B}^\top g(\boldsymbol{s})\,,$$

  *for* $\delta = 1/\sqrt{m}$. *Additionally, a vector* $\boldsymbol{\pi}_{old}$ *that is supported on* $C$ *such that*

$$\mathcal{E}_{\boldsymbol{r}}\left(\boldsymbol{\pi}_{old} - \boldsymbol{\pi}^C\left(\boldsymbol{B}^\top g(\boldsymbol{s})\right)\right) \leq \varepsilon^2 m/4$$

  *is provided, where* $C$ *is the vertex set of the dynamic sparsifier in the* DYNAMICSC *that is maintained internally. Runtime: Worst case* $\widetilde{O}\left(\left(\beta_{\text{CHECKER}}m + (K\beta_{\text{CHECKER}}^{-2}\varepsilon^{-2})^2\right)\varepsilon^{-2}\right)$, *where* $K$ *is the number of* TEMPORARYUPDATE*s that have not been rolled back. Additionally, the output of* CHECK($e$) *is independent of any previous calls to* CHECK.

*Finally, all calls to* CHECK *return valid outputs with high probability. The total number of* UPDATE*s and* TEMPORARYUPDATE*s that have not been rolled back should always be* $O(\beta_{\text{CHECKER}}m)$.

This theorem is from [73]. The only difference is in the guarantee of CHECK. We will now show how it can be implemented. Let

$$\widetilde{\boldsymbol{f}}^* = \delta g(\boldsymbol{s}) - \delta \boldsymbol{R}^{-1}\boldsymbol{B}\boldsymbol{L}^+\boldsymbol{B}^\top g(\boldsymbol{s})\,.$$

Let DYNAMICSC be the underlying Schur complement data structure. We first add the endpoints $u, w$ of $e$ as terminals by calling

$$\text{DYNAMICSC.TEMPORARYADDTERMINALS}(\{u, w\})$$

so that the new Schur complement is on the vertex set $C' = C \cup \{u, w\}$. Then, we set

$$\boldsymbol{\phi} = -\widetilde{SC}^+\boldsymbol{\pi}_{old}$$

and $\widetilde{f}_e = (\phi_u - \phi_w)/\sqrt{r_e}$, where $\widetilde{SC}$ is the output of DYNAMICSC.$\widetilde{SC}$(). Equivalently, note that

$$\widetilde{f}_e = \delta \cdot \mathbf{1}_e^\top \boldsymbol{R}^{-1}\boldsymbol{B}\boldsymbol{L}^+\boldsymbol{\pi}_{old}\,.$$

We will show that $\sqrt{r_e}\left|\widetilde{f}_e - \widetilde{f}_e^*\right| \leq \varepsilon$.

We write

$$\begin{aligned}
&\sqrt{r_e}\left|\widetilde{f}_e - \widetilde{f}_e^*\right| \\
&\leq \left\|\delta\sqrt{\boldsymbol{r}}g(\boldsymbol{s})\right\|_\infty + \left\|\delta\boldsymbol{R}^{-1/2}\boldsymbol{B}\boldsymbol{L}^+\left(\boldsymbol{B}^\top g(\boldsymbol{s}) - \boldsymbol{\pi}^{C'}\left(\boldsymbol{B}^\top g(\boldsymbol{s})\right)\right)\right\|_\infty \\
&+ \left\|\delta\boldsymbol{R}^{-1/2}\boldsymbol{B}\boldsymbol{L}^+\left(\boldsymbol{\pi}^{C'}\left(\boldsymbol{B}^\top g(\boldsymbol{s})\right) - \boldsymbol{\pi}^{C}\left(\boldsymbol{B}^\top g(\boldsymbol{s})\right)\right)\right\|_\infty + \left\|\delta\boldsymbol{R}^{-1/2}\boldsymbol{B}\boldsymbol{L}^+\left(\boldsymbol{\pi}^{C}\left(\boldsymbol{B}^\top g(\boldsymbol{s})\right) - \boldsymbol{\pi}_{old}\right)\right\|_\infty .
\end{aligned}$$

For the first term,

$$\left\|\delta\sqrt{\boldsymbol{r}}g(\boldsymbol{s})\right\|_\infty = \left\|\delta\frac{\frac{1}{\boldsymbol{s}^+} - \frac{1}{\boldsymbol{s}^-}}{\sqrt{\frac{1}{(\boldsymbol{s}^+)^2} + \frac{1}{(\boldsymbol{s}^-)^2}}}\right\|_\infty \leq \delta \leq \varepsilon/10\,.$$

Now, by the fact that $C'$ is a $\beta_{\textsc{Checker}}$-congestion reduction subset by definition in DynamicSC, Lemma 4.4.3 immediately implies that the second term is $\leq \delta \cdot \widetilde{O}\left(\beta_{\textsc{Checker}}^{-2}\right) \leq \varepsilon/10$.

For the third term, we apply Lemma 4.4.12, which shows that

$$\begin{aligned}
&\left\|\delta\boldsymbol{R}^{-1/2}\boldsymbol{B}\boldsymbol{L}^+\left(\boldsymbol{\pi}^{C'}\left(\boldsymbol{B}^\top g(\boldsymbol{s})\right) - \boldsymbol{\pi}^{C}\left(\boldsymbol{B}^\top g(\boldsymbol{s})\right)\right)\right\|_\infty \\
&\leq \delta \cdot \sqrt{\mathcal{E}_{\boldsymbol{r}}\left(\boldsymbol{\pi}^{C'}\left(\boldsymbol{B}^\top g(\boldsymbol{s})\right) - \boldsymbol{\pi}^{C}\left(\boldsymbol{B}^\top g(\boldsymbol{s})\right)\right)} \\
&\leq \delta \cdot \widetilde{O}\left(\beta_{\textsc{Checker}}^{-2}\right) \\
&\leq \varepsilon/10\,,
\end{aligned}$$

as the resistances don't change and we only have two terminal insertions from $C$ to $C'$.

Finally, the fourth term is

$$\left\|\delta\boldsymbol{R}^{-1/2}\boldsymbol{B}\boldsymbol{L}^+\left(\boldsymbol{\pi}^{C}\left(\boldsymbol{B}^\top g(\boldsymbol{s})\right) - \boldsymbol{\pi}_{old}\right)\right\|_\infty \leq \delta \cdot \sqrt{\mathcal{E}_{\boldsymbol{r}}\left(\boldsymbol{\pi}^{C}\left(\boldsymbol{B}^\top g(\boldsymbol{s})\right) - \boldsymbol{\pi}_{old}\right)} \leq \delta \cdot \varepsilon\sqrt{m}/2 = \varepsilon/2\,.$$

We conclude that $\sqrt{r_e}\left|\widetilde{f}_e - \widetilde{f}_e^*\right| \leq \varepsilon$. Finally, we call DynamicSC.Rollback to undo the terminal insertions.

The runtime of this operation is dominated by the call to DynamicSC.$\widetilde{SC}()$, which takes time $\widetilde{O}\left((\beta_{\textsc{Checker}}m + (K\beta_{\textsc{Checker}}^{-2}\varepsilon^{-2})^2)\varepsilon^{-2}\right)$.

## 9.3 Appendix for Chapter 5

### 9.3.1 Approximating Submodular Set Functions with Graph Cuts

It is shown in [45] that submodular set functions defined on a ground set of $n$ elements can be $O\left(n^2\right)$ approximated by directed graph cuts. We state this fact as a lemma, and we include the proof for completeness in Section 9.3.1 below.

**Definition 9.3.1.** *Given a submodular set function $F : 2^V \to \mathbb{R}$, such that $F(\emptyset) = F(V) = 0$, and a weighted directed graph $G = (V, E, c)$, we say that the cut function of $G$ $\alpha$-approximates $F$ if*

$$\frac{1}{\alpha}c^+(A) \leq F(A) \leq c^+(A)\,, \text{for all } A \subseteq V\,.$$

**Lemma 9.3.2.** *Let $V = \{1, \ldots, n\}$, and let $F : V \to \mathbb{R}$ be a non-negative submodular set function, satisfying $F(\emptyset) = F(V) = 0$. Using $O\left(n^2\right)$ calls to a minimization oracle which can compute for all*

**Algorithm 22** Approximate non-negative submodular function $F = F^0 - w^0$ by graph cuts, where $F^0 : 2^V \to \mathbb{R}_{\geq 0}$ is the initial submodular function and the shift vector $w^0 : V \to \mathbb{R}$ is given as input

**function** GRAPHAPPROX($w^0 : V \to \mathbb{R}$)
    Call GRAPHAPPROXSHIFTED($F^0 - w^0$)
**function** GRAPHAPPROXSHIFTED($F : 2^V \to \mathbb{R}_{\geq 0}$)
    Let $E = \{(u, v) \in V \times V : u \neq v\}$.
    **for** $u, v \in V : u \neq v$ **do**
        Compute $w_{uv} = \min_{\substack{A \subseteq V: \\ u \in A, v \notin A}} F(A)$.
        $c_{uv} = w_{uv}$.
    **return** $G = (V, E, c)$

*pairs* $u, v \in V$

$$\min_{\substack{A \subseteq V \\ u \in A, v \notin A}} F(A)$$

*one can compute a weighted directed graph* $G(V, E, c)$ *such that its cut function*

$$c^+(A) := \sum_{\substack{(u,v) \in E: \\ u \in A, v \notin A}} c_{uv}$$

$(n^2/4)$*-approximates* $G$*. In other words, for any* $A \subseteq V$ *the size of the graph cut satisfies:*

$$\frac{1}{n^2/4} \cdot c^+(A) \leq F(A) \leq c^+(A) \ .$$

*Furthermore, if* $F$ *takes only values that are discrete multiples of* $\Delta$*, i.e.* $F(A) \in \Delta \cdot \mathbb{Z}_{\geq 0}$ *for all* $A$*, then all elements of* $c$ *are discrete multiples of* $\Delta$*.*

As a consequence, we obtain a good approximation by graph cuts for decomposable submodular functions where each component in the decomposition acts on few elements, i.e., when $F_i(A) = F_i(A \cap V_i)$ for some $V_i \subseteq V$.

**Lemma 9.3.3.** *Let* $V = \{1, \ldots, n\}$*, and let* $F_i : V_i \to \mathbb{R}$*,* $V_i \subseteq V$ *be non-negative submodular set functions, with* $F_i(\emptyset) = F_i(V_i)$*, for* $i = 1, \ldots, r$*. In the time required to compute for all pairs* $u \neq v \in V$ *and for all* $1 \leq i \leq r$

$$\min_{\substack{A \subseteq V_i \\ u \in A, v \notin A}} F_i(A)$$

*one can compute a weighted directed graph* $G(V, E, c)$ *such that its cut function* $(M^2/4)$*-approximates* $\sum_{i=1}^r F_i$*, where* $M = \max_{i=1,r} |V_i|$.

*Proof.* For each $i$ compute the corresponding graph as in Lemma 9.3.2. Then take the union of edges over the same vertex set. □

We showed that the function $F(A)$ is well approximated by the cut function $c^+(A)$ for the graph we constructed. Note that $c^+$ is only defined on internal vertices of the graph, excluding $s$ and $t$. However this does not affect its submodularity. Therefore the submodular base polytopes for the two function approximate each other well.

**Lemma 9.3.4.** *Let $F, G$ be two submodular functions defined over the same vertex set $V$ such that $F(\emptyset) = G(\emptyset) = 0$, $F(V) = G(V) = 0$, and for any $A \subseteq V$, $\frac{1}{\alpha}G(A) \leq F(A) \leq G(A)$. Then their submodular base polytopes satisfy:*

$$\frac{1}{\alpha}B(G) \subseteq B(F) \subseteq B(G)\,.$$

*Proof.* Let any $w \in B(F)$. Then $w(V) = F(V) = G(V)$. Furthermore for any set $A \subseteq V$, we have $w(A) \leq F(A) \leq G(A)$. Similarly for any $w \in G(A)$, we have $w(A) \leq G(A) \leq \alpha F(A)$, so $B(G) \subseteq \alpha B(F)$, which yields the claim. □

At this point we can prove that the submodular base polytope of the cut function created in Lemma 9.3.3 approximates the submodular base polytope of the decomposable function $\sum_{i=1}^{r} F_i$.

**Lemma 9.3.5.** *Let $V = \{1, \ldots, n\}$, let $F_i : V_i \to \mathbb{R}$, $V_i \subseteq V$ be non-negative submodular set functions, with $F_i(\emptyset) = F_i(V_i)$, for $i = 1, \ldots, r$, and let $F = \sum_{i=1}^{r} F_i$. In the time required to solve $\min_{A \subseteq V_i : u \in A, v \notin A} F_i(A)$ for all $u, v \in V_i$ and all $i$, we can compute a weighted directed graph $G = (V, E, c)$ such that the submodular base polytope of the cut function $c^+(A)$ satisfies $\frac{1}{M^2/4}B(c^+) \subseteq B(F) \subseteq B(c^+)$, where $M = \max_{i=1,r} |V_i|$.*

*Proof.* The proof follows directly from applying Lemma 9.3.3, followed by Lemma 9.3.4. □

**Proof of Lemma 9.3.2**

*Proof.* To simplify notation let us denote by

$$w_{uv} = \min_{\substack{A \subseteq V \\ u \in A, v \notin A}} F\left(A\right)\,,$$

and let $T_{uv}$ be the set achieving this minimum.

Consider the graph defined as follows. For every $u, v \in V$, create an arc $(u, v)$ with weight $c_{uv} = w_{uv}$. By construction all capacities are discrete multiples of $\Delta$.

Now we can prove the lower bound on $F$. We have that

$$c^+\left(A\right) = \sum_{u \in A, v \notin A} c_{uv} \leq \sum_{u \in A, v \notin A} c_{uv} = \sum_{u \in A, v \notin A} F(T_{uv}) \leq \sum_{u \in A, v \notin A} F\left(A\right) \leq \left(n^2/4\right) F(A)\,.$$

We used the fact that $F\left(A\right)$ upper bounds $c_{uv}$ for all $u \in A, v \notin A$. Now we prove the upper bound. For any nonempty set $A \subset V$ we can write $A = \bigcup_{u \in A} \left(\bigcap_{v \in V \setminus A} T_{uv}\right)$. By twice applying Lemma 9.3.6, we obtain that

$$F\left(A\right) \leq \sum_{u \in A} \sum_{v \notin A} F\left(T_{uv}\right) = c^+\left(A\right)\,.$$

Additionally, we have by construction that

$$c^+(\emptyset) = c^+(V) = 0\,.$$

□

**Lemma 9.3.6.** *Let $F$ be a non-negative submodular set function $F : 2^V \to \mathbb{R}$, an let $A_1, \ldots, A_t$ be subsets of $V$. Then*

$$F\left(\bigcup_{i=1}^{t} A_i\right) \leq \sum_{i=1}^{t} F(A_i)$$

*and*

$$F\left(\bigcap_{i=1}^{t} A_i\right) \leq \sum_{i=1}^{t} F(A_i)\,.$$

*Proof.* We prove by induction on $t$. If $t = 1$, both inequalities are equalities. Otherwise, suppose they hold for $t-1$. Let $S = \bigcup_{i=1}^{t-1} A_i$. By submodularity, $F(S \cup A_t) \leq F(S) + F(A_t) - F(S \cap A_t)$. Since $F$ is non-negative, so is $F(S \cap A_t)$, and therefore $F(S \cup A_t) \leq F(S) + F(A_t)$. Applying the induction hypothesis this concludes the first part of the proof.

Similarly, let $S = \bigcap_{i=1}^{t-1} A_i$. By submodularity, $F(S \cap A_t) \leq F(S) + F(A_t) - F(S \cup A_t)$. Since $F$ is non-negative, so is $F(S \cup A_t)$, and therefore $F(S \cap A_t) \leq F(S) + F(A_t)$. Again, applying the induction hypothesis this concludes the second part of the proof. □

### 9.3.2 Parametric Submodular Minimization via Optimization on the Base Polytope

In this section, for completeness, we provide a proof of Lemma 5.2.3, which is based on [15] (see Chapter 8). In addition, we provide error analysis for reductions between approximate solutions to the combinatorial parametric submodular minimization problem, its continuous version involving the Lovász extension, and the dual formulation on the base polytope.

*Proof of Lemma 5.2.3.* Given any point $x$, let $\beta \leq \min\{0, \min_i x_i\}$. Applying the definition of the Lovász extension, and the fundamental theorem of calculus, we can write:

$$\begin{aligned} f(x) + \sum_{i \in V} \psi_i(x_i) &= \int_0^\infty F(\{i : x_i \geq t\})dt + \int_\beta^0 \left(F(\{i : x_i \geq t\}) - F(V)\right) dt \\ &\quad + \sum_{i \in V} \psi_i(\beta) + \int_\beta^\infty \sum_{i : x_i \geq t} \psi_i'(t)dt \\ &= \int_\beta^\infty \left(F(\{i : x_i \geq t\}) + \sum_{i : x_i \geq t} \psi_i'(t)\right) dt + \sum_{i \in V} \psi_i(\beta) - \beta F(V)\,. \end{aligned}$$

Note that we crucially used the fact that the parametric term $\sum_i \psi_i'(t)$ is separable.

Next we show that if the optimal sets $A^\alpha$ were different from those defined in (5.3), then we could obtain a different iterate $x'$ such that $f(x') + \sum_{i \in V} \psi_i(x') \leq f(x^*) + \sum_{i \in V} \psi_i(x^*)$. However, since $\psi$ is strictly convex, the minimizer of $f(x) + \sum_{i \in V} \psi_i(x)$ is unique. This gives a contradiction leading us to the desired conclusion.

Indeed, let $x_i' = \sup_{i \in A^\alpha} \alpha$. By the strict convexity property of $\psi_i$, we have that for any $\alpha > \beta$, $A^\alpha \subseteq A^\beta$, which we reprove for completeness in Lemma 9.3.7.

Using this fact, we know that if $A^\alpha$ are the optimizers of $F(A) + \sum_{i \in V} \psi_i(\alpha)$, then we can write:

$$\int_\beta^\infty \left(F(A^t) + \sum_{i \in A^t} \psi_i'(t)\right) dt = \int_\beta^\infty \left(F(\{i : x_i' \geq t\}) + \sum_{i : x_i' \geq t} \psi_i'(t)\right) dt\,.$$

Since by the optimality of $A^t$ we have that

$$f(A^t) + \sum_{i \in A_t} \psi_i(t) \leq F(\{i : x_i \geq t\}) + \sum_{i : x_i \geq t} \psi_i'(t)\,,$$

it means that letting $\beta = \min\{0, \min_i x_i', \min_i x_i^*\}$,

$$\int_\beta^\infty \left( F(\{i : x_i' \geq t\}) + \sum_{i : x_i' \geq t} \psi_i'(t) \right) dt \leq \int_\beta^\infty \left( F(\{i : x_i^* \geq t\}) + \sum_{i : x_i^* \geq t} \psi_i'(t) \right) dt$$

and therefore

$$f(x') + \sum_{i \in V} \psi_i(x') \leq f(x^*) + \sum_{i \in V} \psi_i(x^*)\,,$$

which concludes the proof. □

**Lemma 9.3.7.** *Let $F : 2^V \to \mathbb{R}$ be a submodular set function, and let $\psi_i : \mathbb{R} \to \mathbb{R}$ be a family of strictly convex functions, for $i \in V$. Let $F_\alpha(A) = F(A) + \sum_{i \in A} \psi_i'(\alpha)$, and $A^\alpha = \arg\min_{A \subseteq V} F_\alpha(A)$. If $\alpha > \beta$, then $A^\alpha \subseteq A^\beta$.*

*Proof.* By optimality we have that

$$F(A^\alpha) + \sum_{i \in A^\alpha} \psi_i'(\alpha) \leq F(A^\alpha \cap A^\beta) + \sum_{i \in A^\alpha \cap A^\beta} \psi_i'(\alpha)$$

and

$$F(A^\beta) + \sum_{i \in A^\beta} \psi_i'(\beta) \leq F(A^\alpha \cup A^\beta) + \sum_{i \in A^\alpha \cup A^\beta} \psi_i'(\beta)\,.$$

Summing up we obtain that

$$\begin{aligned}
&\sum_{i \in A^\alpha} \psi_i'(\alpha) - \sum_{i \in A^\alpha \cap A^\beta} \psi_i'(\alpha) + \sum_{i \in A^\beta} \psi_i'(\beta) - \sum_{i \in A^\alpha \cup A^\beta} \psi_i'(\beta) \\
&\leq F(A^\alpha \cap A^\beta) + F(A^\alpha \cup A^\beta) - F(A^\alpha) - F(A^\beta) \\
&\leq 0\,,
\end{aligned}$$

where we used submodularity in the last step. Therefore

$$\sum_{i \in A^\alpha \setminus A^\beta} (\psi_i'(\alpha) - \psi_i'(\beta)) \leq 0\,.$$

Hence we conclude that $A^\alpha \setminus A^\beta = \emptyset$, since by strict convexity we have that for all $i$, $\psi_i'(\alpha) > \psi_i'(\beta)$, which would make the term above strictly positive had there been any elements in the set difference. □

Next we perform a careful error analysis to bound the total error we incur in the case where the iterate we consider is not an exact minimizer of (5.2), but has some small error in norm.

**Lemma 9.3.8.** *Under the conditions from Lemma 5.2.3, let $\widetilde{x} \in \mathbb{R}$ be a point satisfying $\|\widetilde{x} - x^*\| \leq \varepsilon$, where $x^*$ is the minimizer of (5.2). Let the sets*

$$\widetilde{A}^\alpha = \{i : \widetilde{x}_i \geq \alpha\}\,.$$

*If $\psi_i$ is $\sigma$-strongly convex, for all $i$, and $\max_{A\subseteq V} F(A) - \min_{A'\subseteq V} F(A') \leq M$, then:*

$$F(\widetilde{A}^\alpha) + \sum_{i\in\widetilde{A}^\alpha} \psi_i'(\alpha) \leq F(A^\alpha) + \sum_{i\in A^\alpha} \psi_i'(\alpha) + Mn^{3/2}\varepsilon + \beta\varepsilon^2/2\,.$$

*Proof.* First, using the smoothness of $\psi_i$ we prove that

$$f(\widetilde{x}) + \sum_i \psi_i(\widetilde{x}_i) \leq f(x^*) + \sum_i \psi_i(x_i^*) + Mn^{3/2}\varepsilon + \beta\varepsilon^2/2\,.$$

To prove this we first note that $f$ is Lipschitz, since we can use the fact that entries of the gradient of the Lovász extension consist of differences between $F$ evaluated at different subsets of $V$. Hence for any $x$, $|\nabla_i f(x)| \leq \max_{A\subseteq V} F(A) - \min_{A'\subseteq V} F(A') \leq M$, and thus $\|\nabla f(x)\| \leq M\sqrt{n}$. Therefore

$$f(\widetilde{x}) - f(x^*) \leq M\sqrt{n}\|\widetilde{x} - x^*\| \leq M\sqrt{n}\varepsilon\,.$$

Secondly, we use the smoothness of $\psi_i$, to obtain that

$$\psi_i(\widetilde{x}_i) \leq \psi_i(x_i^*) + \psi_i'(x_i^*)(\widetilde{x}_i - x_i^*) + \frac{\beta}{2}(\widetilde{x}_i - x_i^*)^2$$

Using Lemma 9.3.10 we see that $\psi_i'(x_i^*) = -w_i^*$, where $w^*$ is the optimizer of a certain function over the base polytope $B(F)$. By the definition of $B(F)$ we have $w_i^* \leq F(\{i\}) \leq M$ and $-w_i^* + \sum_{j\neq i} w_j^* = F(V)$, so $-w_i^* \geq F(V) - \sum_{j\neq i} F(\{j\}) \geq -M(n-1)$. Thus, by applying Cauchy-Schwharz, we have

$$\sum_{i\in V} \psi_i'(x_i^*)(\widetilde{x}_i - x_i^*) \leq \max_i |\psi_i'(x_i^*)|\sqrt{n}\cdot\|\widetilde{x} - x^*\| \leq M(n-1)n^{1/2}\varepsilon\,,$$

and thus

$$\sum_{i\in V} \psi_i'(\widetilde{x}_i) - \psi_i'(x_i^*) \leq M(n-1)n^{1/2}\varepsilon + \beta\varepsilon^2/2\,.$$

Combining with the bound on $f(\widetilde{x})$, we obtain our claimed error in function value.

Now we can finalize the argument. Following the proof of Lemma 5.2.3 we write $f(\widetilde{x}) + \sum_{i\in V} \psi_i'(\widetilde{x}_i)$ as an integral, and similarly for $x^*$, to conclude that for $\beta = \min\{0, \min_i \widetilde{x}_i, \min_i x_i^*\}$,

$$\int_\beta^\infty \left(F(\widetilde{A}^t) + \sum_{i\in\widetilde{A}^t} \psi_i'(t)\right) dt \leq \int_\beta^\infty \left(F(A^t) + \sum_{i\in A^t} \psi_i'(t)\right) dt + Mn^{3/2}\varepsilon + \beta\varepsilon^2/2\,.$$

Since by definition $A^t$ minimizes $\sum_{i\in A^t} \psi_i'(t)$, we conclude that for all $t$,

$$F(\widetilde{A}^t) + \sum_{i\in\widetilde{A}^t} \psi_i'(t) \leq F(A^t) + \sum_{i\in A^t} \psi_i'(t) + Mn^{3/2}\varepsilon + \beta\varepsilon^2/2\,.$$

□

We can also show that if we obtain an approximate minimizer of the dual problem (5.4) over $B(F)$, we can use it to recover an approximate minimizer of the primal problem (5.2).

**Lemma 9.3.9.** *Let $w^*$ be the minimizer of the dual problem (5.4), and and let $x^*$ be the minimizer*

*of the primal problem (5.2). If $w \in B(F)$ such that*

$$\sum_{i \in V} \psi_i^*(-w_i) \leq \sum_{i \in V} \psi_i^*(-w_i^*) + \varepsilon \,,$$

*then the point $x \in \mathbb{R}^n$ where $x_i = (\psi_i^*)'(-w_i)$ satisfies*

$$\|x - x^*\| \leq \sqrt{\frac{2L\varepsilon}{\sigma^2}} \,.$$

*Proof.* By Lemma 9.3.10 we know that $x^*$ and $w^*$ are related via $x_i^* = (\psi_i^*)'(-w_i^*)$. Therefore we can write

$$|x_i - x_i^*| = |(\psi_i^*)'(-w_i) - (\psi_i^*)'(-w_i^*)| \leq \frac{1}{\sigma} |w_i - w_i^*| \,,$$

where in the last inequality we used the fact that $\psi_i$ is $\sigma$-strongly convex, and hence $\psi_i^*$ is $1/\sigma$-smooth [145, 92]. Next we show that $|w_i - w_i^*|$ is bounded by a function of $\varepsilon$.

Since by assumption $\psi_i$ is $L$-smooth, its dual $\psi_i^*$ is $1/L$-strongly convex. Therefore we have that, for all $i$:

$$\psi_i^*(-w_i) \geq \psi_i^*(-w_i^*) + (\psi_i^*)'(-w_i^*) \cdot (-w_i - (-w_i^*)) + \frac{\sigma}{2}(w_i^* - w_i)^2 \,.$$

Furthermore, since $w^*$ is an optimizer over $B(F)$, we know by first-order optimality that for any $w \in B(F)$:

$$\sum_{i \in V} (\psi_i^*)'(-w_i^*) \cdot (-w_i - (-w_i^*)) \geq 0 \,,$$

i.e. slightly moving the point from $-w^*$ towards $-w$ can only increase function value. Thus we obtain that

$$\sum_{i \in V} \psi_i^*(-w_i) \geq \sum_{i \in V} \psi_i^*(-w_i^*) + \frac{1}{2L} \sum_{i \in V} (w_i^* - w_i)^2 \,.$$

Combining with the hypothesis, this implies that

$$\frac{1}{2L} \sum_{i \in V} (w_i^* - w_i)^2 \leq \varepsilon \,,$$

and therefore

$$\|x - x^*\|^2 \leq \frac{1}{\sigma^2} \sum_{i \in V} (w_i - w_i^*)^2 \leq \frac{2L\varepsilon}{\sigma^2} \,,$$

which implies the claimed result. □

As a corollary of the previous lemmas, we see that an approximate solution to the dual problem (5.4) yields an approximate solution to the original parametric problem (5.1).

**Corollary 9.3.1.** *Let $F : 2^V \to \mathbb{R}$ be a non-negative submodular set function, and let the the family of parametric problems defined in (5.1). Let $w \in B(F)$ such that*

$$\sum_{i \in V} \psi_i^*(-w_i) \leq \sum_{i \in V} \psi_i^*(-w_i^*) + \varepsilon \,,$$

*where $w^*$ is the true minimizer of the dual problem (5.4). Then for any $\alpha$, the set*

$$\widetilde{A}^\alpha = \{i : \psi_i^*(-w_i) \geq \alpha\}$$

*satisfies*

$$F_\alpha(\widetilde{A}^\alpha) \leq F_\alpha(A^\alpha) + \sqrt{\varepsilon} \cdot Mn^{3/2}\sqrt{2L/\sigma^2} + \varepsilon \cdot (L/\sigma)^2\,,$$

*where* $A^\alpha = \arg\min_{A \subseteq V} F_\alpha(A)$.

*Proof.* From Lemma 9.3.9 we know that the hypothesis implies that the point $x$ where $x_i = (\psi_i^*)(-w_i)$ satisfies $\|x - x^*\| \leq \sqrt{2L\varepsilon/\sigma^2}$. Applying Lemma 9.3.8 we thus obtain that the sets constructed satisfy

$$F_\alpha(\widetilde{A}^\alpha) \leq F_\alpha(A^\alpha) + Mn^{3/2}\sqrt{2L\varepsilon/\sigma^2} + L/2 \cdot (2L\varepsilon/\sigma^2)\,,$$

which yields our claim. □

The following helper lemma shows that we can efficiently convert between (exact) solutions to the primal and dual problems (5.2) and (5.4). Using standard techniques we can prove that these also enable us to convert between suboptimal solutions, while satisfying certain error bounds.

**Lemma 9.3.10.** *Let $x^*$ be the (unique) minimizer of (5.2), and let $w^*$ be the minimizer of (5.4). Then $w_i^* = -\psi_i'(x_i)$ and $(\psi_i^*)'(-w_i) = x_i$, for all $i \in V$.*

*Proof.* We the dual characterization of $f$ and Sion's theorem, to write

$$\min_{x\in\mathbb{R}^n} f(x) + \sum_{i\in V} \psi_i(x_i) = \min_{x\in\mathbb{R}^n} \max_{w\in B(F)} \langle w, x\rangle + \sum_{i\in V} \psi_i(x_i) = \max_{w\in B(F)} \min_{x\in\mathbb{R}^n} \langle w, x\rangle + \sum_{i\in V} \psi_i(x_i)\,.$$

Since each $\psi_i$ acts on a different coordinate we can write the inner minimization problem as

$$\min_{x\in\mathbb{R}^n} \sum_{i\in V} (w_i x_i + \psi_i(x_i)) = -\sum_{i\in V} \psi_i^*(-w_i)\,,$$

where we applied the definition of the Fenchel dual. Furthermore by standard convex analysis [23, 140], as $\psi_i'$ ranges from $-\infty$ to $\infty$ for each $i$ we have that $(\psi_i^*)'(-w_i) = x_i$, and similarly $\psi_i'(x_i) = -w_i$.

Thus we can equivalently write (5.2) as

$$\max_{w\in B(F)} -\sum_{i\in V} \psi_i^*(-w_i)\,.$$

By the previous observation, the optima are thus related via $(\psi_i^*)'(-w_i^*) = x_i^*$, and similarly $\psi_i'(x_i^*) = -w_i^*$. □

### 9.3.3 Parametric $s$-$t$ Cuts

In this section we show how to solve the parametric minimum cut problem by efficiently using a maximum flow oracle. In Section 5.4 we show how to convert the solution obtained by this combinatorial routine to a nearly-optimal solution to a related optimization problem on the submodular base polytope of the corresponding cut function.

In the parametric min $s,t$-cut problem, we are given a directed network $G = (V, E)$ with two distinguished vertices: a *source* $s \in V$, and a *sink* $t \in V$, $s \neq t$. The capacities of individual edges of $G$ are nonnegative functions of a real parameter $\lambda$ in some possibly infinite domain $\mathbb{D} \subseteq \mathbb{R}$ (as opposed to constants in the classical setting of min $s,t$-cut). Following [72], we assume that the capacities of edges $sv \in E$ are nondecreasing in $\lambda$ and the capacities of edges $vt \in E$ are nonincreasing in $\lambda$. The capacities of all other edges of $G$ are constant.

We denote by $c_\lambda(uv) : \mathbb{D} \to \mathbb{R}$ the capacity function of an edge $uv \in E$. Moreover, we assume that these edge capacity functions can be evaluated for arbitrary $\lambda$ in constant time.

Roughly speaking, the goal of the parametric min $s,t$-cut problems is to compute a representation of min $s,t$-cut for all the possible parameters $\lambda$. Before we precisely define what this means, let us introduce some more notation and state some useful properties of (parametric) min-cuts.

Denote by $\mathrm{cap}(G)$ the capacity of a min $s,t$-cut in $G$. Let $G_{\lambda'}$ be the graph with all the parameterized capacities replaced with the corresponding values for $\lambda = \lambda'$. For any $S$, $s \in S \subseteq V \backslash \{t\}$, let $c_\lambda(S)$ be the capacity function of $S$, i.e., the sum of capacity functions $c_\lambda(uv)$ through all $uv$ with $u \in S$ and $v \in V \setminus S$.

**Lemma 9.3.2** ([62])**.** *For any $G$, there exists a unique* minimal *minimum $s,t$-cut $(S,T)$ with $|S|$ smallest possible, such that for any min $s,t$-cut $(S',T')$ of $G$ we have $S \subseteq S'$. Given any maximum $s,t$-flow $f$ in $G$, such a cut can be computed from $f$ in $O(m)$ time.*

*Proof.* Let $G_f$ be the residual network associated with flow $f$. We let $S$ be the set of vertices reachable from $s$ in $G_f$ (via edges with positive capacity). As proven by Ford and Fulkerson [62, Theorem 5.5], $S$ defined this way does not depend on the chosen maximum flow $f$, and $S \subseteq S'$ holds. Clearly, given $f$, $S$ can be found using any graph search algorithm. □

Ford and Fulkerson [62, Corollary 5.4] showed that for any two min $s,t$-cuts $(S_1,T_1),(S_2,T_2)$ of $G$, $(S_1 \cap S_2, T_1 \cup T_2)$ is also a min $s,t$-cut of $G$. Gallo et al. [72, Lemma 2.8] gave the following generalization of this property to parametric min $s,t$-cuts.

**Lemma 9.3.3** ([72])**.** *Let $\lambda_1 \leq \lambda_2$. For $i = 1,2$, let $(S_{\lambda_i}, T_{\lambda_i})$ be some min $s,t$-cut in $G_{\lambda_i}$. Then $(S_{\lambda_1} \cap S_{\lambda_2}, T_{\lambda_1} \cup T_{\lambda_2})$ is a min $s,t$-cut in $G_{\lambda_1}$.*

Our algorithm will use the following crucial property of parametric minimal min $s,t$-cuts.

**Lemma 9.3.4.** *Let $\lambda_1 \leq \lambda_2$. For $i = 1,2$, let $(S_{\lambda_i}, T_{\lambda_i})$ be the unique minimal min $s,t$-cut in $G_{\lambda_i}$. Then $S_{\lambda_1} \subseteq S_{\lambda_2}$.*

*Proof.* The uniqueness of $S_{\lambda_1}$ and $S_{\lambda_2}$ follows by Lemma 9.3.2 applied to $G_{\lambda_1}$ and $G_{\lambda_2}$, respectively. By Lemma 9.3.3, $(S_{\lambda_1} \cap S_{\lambda_2}, T_{\lambda_1} \cup T_{\lambda_2})$ is a min $s,t$-cut in $G_{\lambda_1}$. By Lemma 9.3.2, we have $S_{\lambda_1} \subseteq S_{\lambda_1} \cap S_{\lambda_2}$. It follows that $S_{\lambda_1} \subseteq S_{\lambda_2}$. □

Now given Lemma 9.3.4, we can formally state our goal in this section, which is to compute a parametric min $s,t$-cut defined as follows. Let $\lambda_{\min} \in \mathbb{D}$ be such that the minimal min $s,t$-cuts of $G_{\lambda_{\min}}$ and $G_{\lambda'}$ are equal for all $\lambda' \in \mathbb{D}$, $\lambda' < \lambda_{\min}$. Similarly, let $\lambda_{\max} \in \mathbb{D}$ be such that the minimal min $s,t$-cuts of $G_{\lambda_{\max}}$ and $G_{\lambda'}$ are equal for all $\lambda' \in \mathbb{D}$ with $\lambda' > \lambda_{\max}$. We will consider $\lambda_{\min}$ and $\lambda_{\max}$ additional inputs to our problem.

For simplicity, in the remaining part of this section we denote by $S_\lambda$ and $T_\lambda$ the $s$-side and the $t$-side (resp.) of *the minimal* min-$s,t$-cut of $G_\lambda$.

**Definition 9.3.11** (Parametric min $s,t$-cut)**.** *Let $\Lambda = \{\lambda_1, \ldots, \lambda_k\} \subseteq \mathbb{D}$, where $k \leq n-1$ and $\lambda_{\min} < \lambda_1 < \ldots < \lambda_k \leq \lambda_{\max}$. Let $\lambda_0 = \lambda_{\min}$. Let $\tau : V \to \Lambda \cup \{\lambda_{\min}, \infty\}$ be such that $\tau(s) = \lambda_{\min}$ and $\tau(t) = \infty$. Let $S(z) = \{v \in V : \tau(v) \leq z\}$. A pair $(\Lambda, \tau)$ is a parametric min $s,t$-cut of $G$ if:*

1. *For $i = 0, \ldots, k-1$, $S(\lambda_i)$ is a minimal min $s,t$-cut of $G_{\lambda'}$ for all $\lambda' \in [\lambda_i, \lambda_{i+1}) \cap \mathbb{D}$.*
2. *$S(\lambda_k)$ is a minimal min $s,t$-cut of $G_{\lambda_{\max}}$.*
3. *For $i = 0, \ldots, k-1$, $S(\lambda_i) \subsetneq S(\lambda_{i+1})$.*

It will also prove useful to define an approximate version of parametric min $s,t$-cut.

**Definition 9.3.12** ($\varepsilon$-approximate parametric min $s,t$-cut)**.** *Let $\Lambda$, $\tau$, and $S : \mathbb{D} \to 2^V$ be as in Definition 9.3.11. A pair $(\Lambda, \tau)$ is called an $\varepsilon$-approximate parametric min $s,t$-cut of $G$ if:*

1. *For $i = 0, \ldots, k-1$, $S(\lambda_i)$ is a minimal min $s,t$-cut of $G_{\lambda'}$ for all $\lambda' \in [\lambda_i, \lambda_{i+1} - \varepsilon) \cap \mathbb{D}$.*
2. *$S(\lambda_k)$ is a minimal min $s,t$-cut of $G_{\lambda_{\max}}$.*
3. *For $i = 0, \ldots, k-1$, $S(\lambda_i) \subsetneq S(\lambda_{i+1})$.*

**Lemma 9.3.5.** *Let $(\Lambda, \tau)$ be the parametric min $s,t$-cut of $G$. Let $(\Lambda_\varepsilon, \tau_\varepsilon)$ be an $\varepsilon$-approximate parametric min $s,t$-cut of $G$. Then for all $v \in V$, $\tau(v) \le \tau_\varepsilon(v) \le \tau(v) + \varepsilon$.*

*Proof.* Let $S(z) = \{v \in V : \tau(v) \le z\}$, and $S_\varepsilon(z) = \{v \in V : \tau_\varepsilon(v) \le z\}$. First of all, $\tau(v) = \infty$ if and only if $\tau_\varepsilon(v) = \infty$. This is because each of those is equivalent to $v \notin S_{\lambda_{\max}}$. In this case the lemma holds trivially.

So in the following let us assume that $\tau(v)$ and $\tau_\varepsilon(v)$ are both finite. We first prove $\tau_\varepsilon(v) \ge \tau(v)$. If $\tau(v) = \lambda_{\min}$ then this follows by $\tau_\varepsilon(v) \ge \lambda_{\min}$. So suppose $\tau(v) = \lambda$ for some $\lambda \in \Lambda$. Then by item (1) of Definition 9.3.11, for any $\lambda' < \lambda$, $S(\lambda')$ is a minimal min $s,t$-cut of $G_{\lambda'}$ and $v \notin S(\lambda')$. If we had $\tau_\varepsilon(v) < \tau(v)$, then $S_\varepsilon(\tau_\varepsilon(v))$ would be a minimal min $s,t$-cut of $G_{\tau_\varepsilon(v)}$ such that $v \in S_{\tau_\varepsilon(v)}$ and $\tau_\varepsilon(v) < \lambda$, a contradiction.

Now let us prove $\tau_\varepsilon(v) \le \tau(v) + \varepsilon$. To this end, suppose $\tau_\varepsilon(v) > \tau(v) + \varepsilon$. If $\tau_\varepsilon(v) = \lambda_{\min}$, then we have $\lambda_{\min} > \tau(v) + \varepsilon \ge \lambda_{\min} + \varepsilon$, a clear contradiction. So let us assume that $\tau_\varepsilon(v) \in \Lambda_\varepsilon$ and let $\lambda^*$ be the element preceding $\tau_\varepsilon(v)$ in $\Lambda_\varepsilon$, or $\lambda^* = \lambda_{\min}$ if no such element exists. We have $v \notin S_{\lambda^*}$ and $S_{\lambda^*}$ is a minimal min $s,t$-cut in $G_{\lambda'}$ for $\lambda' = \lambda^*$ and all $\lambda' \in [\lambda^*, \tau_\varepsilon(v) - \varepsilon)$. As a result, for any $\lambda'' < \tau_\varepsilon(v) - \varepsilon$, the minimal min $s,t$-cut of $G_{\lambda''}$ does not contain $v$ in the $s$-side. But $\tau(v) < \tau_\varepsilon(v) - \varepsilon$, $v \in S(\tau(v))$, and $S(\tau(v))$ is a minimal min $s,t$-cut of $G_{\tau(v)}$, a contradiction. □

Our main result in this section is the following theorem.

**Theorem 5.3.2.** *Let $R = \lambda_{\max} - \lambda_{\min}$ be an integral multiple of $\varepsilon > 0$. Let $T_{\text{maxflow}}(n', m') = \Omega(m' + n')$ be a convex function bounding the time needed to compute maximum flow in a graph with $n'$ vertices and $m'$ edges obtained from $G_\lambda$ by edge/vertex deletions and/or edge contractions (with merging parallel edges by summing their capacities) for any $\lambda = \lambda_{\min} + \ell\varepsilon$ and any integer $\ell \in [0, R/\varepsilon]$. Then, $\varepsilon$-approximate parametric min $s,t$-cut in $G$ can be computed in $O(T_{\text{maxflow}}(n, m \log n) \cdot \log \frac{R}{\varepsilon} \cdot \log n)$ time.*

The rest of this section is devoted to proving Theorem 5.3.2. For a connected subset $X \subseteq V(G)$, $\{s,t\} \not\subseteq X$, let $G/X$ denote $G$ after merging the vertex set $X$ into a single vertex. If the contracted vertex set $X$ contains $s$ ($t$), then the resulting vertex inherits the identity of $s$ ($t$, resp.).

**Lemma 9.3.6.** *Let $\lambda$ be arbitrary and let $(S_\lambda, T_\lambda)$ be the minimal min $s,t$-cut in $G_\lambda$. Then:*

1. *For any $\lambda' \ge \lambda$, $cap(G_{\lambda'}) = cap(G_{\lambda'}/S_\lambda)$.*
2. *For any $\lambda' \le \lambda$, $cap(G_{\lambda'}) = cap(G_{\lambda'}/T_\lambda)$.*

*Proof.* We only prove item 1, as item 2 is analogous. Since merging vertices is equivalent to connecting them with infinite capacity edges, it cannot decrease the min $s,t$-cut capacity, i.e., $\text{cap}(G_{\lambda'}) \le \text{cap}(G_{\lambda'}/S_\lambda)$. On the other hand, by Lemma 9.3.4, the minimal $s,t$ min-cut $(S_{\lambda'}, T_{\lambda'})$ in $G_{\lambda'}$ satisfies $S_\lambda \subseteq S_{\lambda'}$. Hence, the capacity of the $s,t$-cut $(S_{\lambda'}/S_\lambda, T_{\lambda'})$ in $G_{\lambda'}/S_\lambda$ is the same as the capacity $\text{cap}(G_{\lambda'})$ of $(S_{\lambda'}, T_{\lambda'})$ in $G_{\lambda'}$. Consequently, $\text{cap}(G_{\lambda'}) \ge \text{cap}(G_{\lambda'}/S_\lambda)$. □

**Remark 9.3.7.** *If $(S_\lambda, T_\lambda)$ is a minimal min $s,t$-cut in $G_\lambda$, then $G[S_\lambda]$ is connected by construction (Lemma 9.3.2). However, $G[T_\lambda]$ might in general consist of several connected components if $G_\lambda$ contains zero-capacity edges. In that case, we can still obtain $G_{\lambda'}/T_\lambda$ above using edge/vertex deletions and edge contractions. Namely, we contract only the connected component $A$ of $T_\lambda$ that contains $t$. For any other component $C_i$ ($i = 1, \ldots, q$) of $T_\lambda$, its incoming edges start in $S_\lambda$ and all have capacity 0 in $G_\lambda$, and thus also in $G_{\lambda'}$ for $\lambda' < \lambda$. Consequently, removing the vertices of $\bigcup_{i=1}^{q} C_i$ and subsequently contracting $A$ has the same effect on $G_{\lambda'}$ as merging the entire $T_\lambda$, i.e., $G_{\lambda'}/T_\lambda = G_{\lambda'}[V \setminus \bigcup_{i=1}^{q} C_i]/A$.*

We use a recursive "divide-and-conquer" algorithm. The input to a recursive procedure ApxParametricMinCut is a graph $G = (V, E)$ with $n$ vertices, $m$ edges, source $s$ and sink $t$, the parametric capacity function $c_\lambda : E \to \mathbb{D} \to \mathbb{R}$, and two parameters $\lambda_{\min}, \lambda_{\max}$ such that $\varepsilon$ evenly divides $\lambda_{\max} - \lambda_{\min}$. The output of the procedure is an $\varepsilon$-approximate parametric min $s,t$-cut $(\{\lambda_1, \ldots, \lambda_k\}, \tau)$ as in Definition 9.3.12. By Lemma 9.3.4, $k \leq |V(G)| - 1$.

**Algorithm 23** Computing an $\varepsilon$-approximate parametric min $s,t$-cut.

Let $s, t, \varepsilon$ be globally defined.
**function** ApxParametricMinCut($G = (V, E), c_\lambda : E \to \mathbb{D} \to \mathbb{R}, \lambda_{\min} \in \mathbb{D}, \lambda_{\max} \in \mathbb{D}$)
  **if** $|V| \leq 2$ **then return** $(\emptyset, \{s \to \lambda_{\min}, t \to \infty\})$
  For any $\lambda' \in \mathbb{D}$, let $c_\lambda[\lambda = \lambda']$ the capacity function $E \to \mathbb{R}$ of $G_{\lambda'}$
  $S_{\lambda_{\min}} =$ MinimalMinCut$(G, c_\lambda[\lambda = \lambda_{\min}])$
  $S_{\lambda_{\max}} =$ MinimalMinCut$(G, c_\lambda[\lambda = \lambda_{\max}])$
  **if** $|S_{\lambda_{\min}}| > |V|/2$ **then return** ApxParametricMinCut(Contract$(G, c_\lambda, S_{\lambda_{\min}}), \lambda_{\min}, \lambda_{\max})$
  **if** $|S_{\lambda_{\max}}| < |V|/2$ **then return** ApxParametricMinCut(Contract$(G, c_\lambda, V \setminus S_{\lambda_{\max}}), \lambda_{\min}, \lambda_{\max})$
  $(\lambda_1, \lambda_2) := (\lambda_{\min}, \lambda_{\max})$
  **while** $\lambda_2 - \lambda_1 > \varepsilon$ **do**
    $\lambda' := \lambda_1 + \lfloor (\lambda_2 - \lambda_1)/2\varepsilon \rfloor \cdot \varepsilon$
    $S_{\lambda'} =$ MinimalMinCut$(G, c_\lambda[\lambda = \lambda'])$
    **if** $|S_{\lambda'}| \geq |V|/2$ **then**
      $\lambda_2 := \lambda'$
    **else**
      $\lambda_1 := \lambda'$
  For $i = 1, 2$, $S_{\lambda_i} :=$ MinimalMinCut$(G, c_\lambda[\lambda = \lambda_i])$
  $(\Lambda_1, \tau_1) =$ ApxParametricMinCut(Contract$(G, c_\lambda, V \setminus S_{\lambda_1}), \lambda_{\min}, \lambda_1)$
  $(\Lambda_2, \tau_2) =$ ApxParametricMinCut(Contract$(G, c_\lambda, S_{\lambda_2}), \lambda_2, \lambda_{\max})$
  $\Lambda :=$ **if** $|S_{\lambda_1}| = |S_{\lambda_2}|$ **then** $\Lambda_1 \cup \Lambda_2$ **else** $\Lambda_1 \cup \{\lambda_2\} \cup \Lambda_2$
  $\tau := \{v \in S_{\lambda_1} \to \tau_1(v), v \in V \setminus S_{\lambda_2} \to \tau_2(v), v \in S_{\lambda_2} \setminus S_{\lambda_1} \to \lambda_2\}$ **return** $(\Lambda, \tau)$

The main idea of the procedure ApxParametricMinCut is to find the (approximately) most balanced minimal $s,t$-cuts $S_{\lambda_1}$ and $S_{\lambda_2}$ and use them to reduce the problem size in the recursive calls significantly. Specifically, we want to find such $\lambda_1 \leq \lambda_2$ that $|S_{\lambda_1}| \leq n/2$, $|S_{\lambda_2}| \geq n/2$ and $\lambda_2 - \lambda_1 = \varepsilon$.

Suppose $n > 2$ as otherwise the problem is trivial. First, we compute minimal min-cuts in $G_{\lambda_{\min}}$ and $G_{\lambda_{\max}}$. This takes two max-flow runs, i.e., $T_{\text{maxflow}}(n, m)$ time, plus $O(m)$ time by Lemma 9.3.2.

It might happen that $|S_{\lambda_{\min}}| \leq |S_{\lambda_{\max}}| < n/2$ or $n/2 < |S_{\lambda_{\min}}| \leq |S_{\lambda_{\max}}|$. In these special cases we can immediately reduce the vertex set by a factor of at least two by contracting $T_{\lambda_{\max}}$ or $S_{\lambda_{\min}}$

```
22: function MinimalMinCut(G = (V, E), c : E → ℝ)
23:     f = MaxFlow(G, s, t, c) return {v ∈ V : v reachable from s in the residual network G_f}

24: function Contract(G = (V, E), c_λ : E → 𝔻 → ℝ, X ⊆ V)          ▷ |X ∩ {s, t}| = 1
25:     w* := if s ∈ X then s else t
26:     V' := V \ X ∪ {w*}
27:     E' := ∅
28:     c'_λ := E' → 𝔻 → ℝ
29:     for uv ∈ E do
30:         u' := if u ∈ X then w* else u
31:         v' := if v ∈ X then w* else v
32:         if (u', v') ≠ (s, t) then
33:             if u'v' ∉ E' then
34:                 E' := E' ∪ {u'v'}
35:                 c'_λ(u'v') := c_λ(uv)
36:             else
37:                 c'_λ(u'v') := c'_λ(u'v') + c_λ(uv)          ▷ We add functions here.
        return (G' = (V', E'), c'_λ)
```

respectively, and recurse on the reduced graph. By Lemma 9.3.6 and the definition of $\lambda_{\min}, \lambda_{\max}$ this reduction does not influence the structure of parametric cuts.

So suppose $|S_{\lambda_{\min}}| \le n/2$ and $|S_{\lambda_{\max}}| \ge n/2$. Set $\lambda_1 = \lambda_{\min}$ and $\lambda_2 = \lambda_{\max}$. So we have $|S_{\lambda_1}| \le n/2$ and $|S_{\lambda_2}| \ge n/2$ initially. We maintain this invariant and gradually shrink the interval $[\lambda_1, \lambda_2]$ until its length gets precisely $\varepsilon$ in a binary search-like way. We repeatedly try the pivot $\lambda' = \lambda_1 + \lfloor(\lambda_2 - \lambda_1)/2\varepsilon\rfloor \cdot \varepsilon$ and compute $S_{\lambda'}$. If $|S_{\lambda'}| \ge n/2$, we set $\lambda_2 = \lambda'$, and otherwise we set $\lambda_1 = \lambda'$. Note that $\lambda_2 - \lambda_1$ remains an integer multiple of $\varepsilon$ at all times. The whole process costs $O(\log[(\lambda_{\max} - \lambda_{\min})/\varepsilon]) = O(\log(R/\varepsilon))$ max-flow executions.

Let $G_1 = G/T_{\lambda_1}$ and $G_2 = G/S_{\lambda_2}$. Note that $G_1, G_2$ may contain parallel edges or a direct $st$ edge as a result of contraction. Hence, these graphs are first preprocessed by (1) removing self-loops and direct $st$ edges, (2) merging parallel edges by summing their cost functions. The contraction and preprocessing is performed using the procedure Contract. Note that none of these preprocessing steps change the minimal cuts of $G_{1,\lambda}$ or $G_{2,\lambda}$ for any $\lambda$: the direct $st$ edges cross *all* $s, t$-cuts.

Next, we recursively compute $\varepsilon$-approximate parametric min $s, t$-cut in graphs $G_1 = G/T_{\lambda_1}$ and $G_2 = G/S_{\lambda_2}$, The recursive call on $G_1$ is made with $(\lambda_{\min}, \lambda_{\max})$ set to $(\lambda_{\min}, \lambda_1)$, whereas the recursive call on $G_2$ uses $(\lambda_{\min}, \lambda_{\max}) := (\lambda_2, \lambda_{\max})$. Note that indeed we have $G_{1,\lambda_1} = G_{1,\lambda'}$ for $\lambda' > \lambda_1$ as required since the $t$-side of the minimal min $s, t$-cut in $G_{1,\lambda_1}$ contains only $t$. Similarly, $G_{2,\lambda_2} = G_{2,\lambda'}$ for all $\lambda' < \lambda_2$.

Let $(\Lambda_1, \tau_1) = (\{\lambda_{1,1}, \dots, \lambda_{1,a}\}, \tau_1)$ and $(\Lambda_2, \tau_2) = (\{\lambda_{2,1}, \dots, \lambda_{2,b}\}, \tau_2)$ be the returned $\varepsilon$-approximate parametric min $s, t$-cuts of $G_1$ and $G_2$ respectively. We return $(\Lambda, \tau)$ as the $\varepsilon$-approximate parametric min-$s, t$-cut of $G$, where

$$\Lambda = \begin{cases} \Lambda_1 \cup \Lambda_2 & \text{if } |S_{\lambda_1}| = |S_{\lambda_2}| = n/2, \\ \Lambda_1 \cup \{\lambda_2\} \cup \Lambda_2 & \text{otherwise.} \end{cases} \qquad \tau(v) = \begin{cases} \tau_1(v) & \text{if } v \in S_{\lambda_1}, \\ \tau_2(v) & \text{if } v \in T_{\lambda_2}, \\ \lambda_2 & \text{otherwise.} \end{cases}$$

Let us now prove the correctness of this algorithm. We proceed by induction on $n$. For $n \le 2$

this is trivial, so suppose $n > 3$ and that recursive calls are made. Clearly, $\lambda_{1,a} \leq \lambda_1 < \lambda_2 < \lambda_{2,1}$.

Item (3) of Definition 9.3.12 follows easily by induction and the definition of $\lambda_1, \lambda_2$. Let $\Lambda = \{\lambda'_1, \ldots, \lambda'_k\}$. That $S(\lambda'_i) = \{v \in V : \tau(v) \leq \lambda'_i\}$ is a minimal min $s,t$-cut of $G_{\lambda'_i}$ for all $\lambda'_i \in \Lambda$ (i.e., item (2) of Definition 9.3.12) follows directly by Lemma 9.3.6 the definitions of $\lambda_1, \lambda_2$.

Now consider item (1) of Definition 9.3.12. For some $j < k$ we have $\lambda'_j = \lambda_{1,a}$. For all $i = 0, \ldots, k-1$, $i \neq j$, item (1), i.e., that $S_{\lambda'_i}$ is a minimal min $s,t$-cut for all $\lambda' \in [\lambda'_i, \lambda'_{i+1})$, follows directly inductively.

If $|S_{\lambda_1}| = |S_{\lambda_2}| = n/2$, then $S_{\lambda_{1,a}} = S_{\lambda_2}$. By induction it follows that $S_{\lambda_{1,a}}$ is a minimal min $s,t$-cut of $G_{\lambda'}$ for all $\lambda' \in [\lambda_2, \lambda_{2,1} - \varepsilon)$, and thus also for all $\lambda' \in [\lambda_{1,a}, \lambda_{2,1} - \varepsilon) = [\lambda'_j, \lambda'_{j+1} - \varepsilon)$.

If, on the other hand, $S_{\lambda_1} \subsetneq S_{\lambda_2}$, then $\lambda'_{j+1} = \lambda_2$. Since $S_{\lambda_{1,a}} = S_{\lambda_1}$, $S_{\lambda_{1,a}}$ is indeed a minimal min $s,t$-cut for all $\lambda' \in [\lambda_{1,a}, \lambda_2 - \varepsilon) = [\lambda'_j, \lambda'_{j+1} - \varepsilon)$ as $\lambda_2 - \varepsilon = \lambda_1$.

Note that the input graph of each of the recursive calls has at most $n/2 + 1$ vertices. Moreover, by merging the parallel edges (and summing their costs) after the contraction we can guarantee that $|E(G_1)| + |E(G_2)| \leq |E(G)| + n/2$. Indeed, observe that the only edges of $G_2$ that can also appear in $G_1$ are those incident to $s$ in $G_2$, and there are at most $|T_{\lambda_2}| \leq n/2$ of them.

There is one important technical detail here: even though the individual functions $c_\lambda(uv)$ (for the edges $uv$ of the original input graph $G$) can be evaluated in constant time, after summing $k$ of such functions in the process this cost can be as much as $\Theta(k)$. We now argue that this cannot happen in our case due to preprocessing $G_1$ and $G_2$, and the evaluation cost is $O(1)$ for all edges in all recursive calls. More concretely, one can show that each edge capacity function in a recursive call can be expressed as the sum of at most one original capacity function $c_\lambda(xy)$ and a real number. Indeed, suppose this is the case for some call with input $G$. Then, each edge $uv$ of $G_1$ either (1) is contained in $G$ and has not resulted from merging some parallel edges after contraction, (2) has $u \notin \{s,t\}$ and $v = t$ and resulted from merging edges $uz_1, \ldots, uz_l$ such that $z_1, \ldots, z_l \in T_{\lambda_1}$. The former case is trivial. In the latter case, for at least $l-1$ of these $z_i$ we have $z_i \neq t$, so $c_\lambda(uz_i)$ is a constant function. Fot at most one $z_j$ is of the form $c_\lambda(xy) + \Delta$ for some original capacity function $c_\lambda(xy)$ and $\Delta \in \mathbb{R}$. We conclude that the capacity function of $uv$ in $G_1$ is of the same form and equals $c_\lambda(xy) + \Delta'$, where $\Delta' = \Delta + \sum_{i \neq j} c_\lambda(uz_i) \in \mathbb{R}$. The proof for $G_2$ is analogous.

Now let us analyze the running time of the algorithm. One can easily inductively prove that:

- Each graph at the $i$-th level of the recursion tree has less than $n/2^i + 2$ vertices; hence, there are no more than $\log_2 n + 1$ levels in the tree.
- The sum $n_i$ of numbers of vertices through all the graphs at the $i$-th level is less than $2^i(n/2^i + 2) \leq n + 2^{i+1} \leq 3n$.
- Since, the sum $m_i$ of numbers of edges in graphs at level $i > 0$ satisfies $m_i \leq m_{i-1} + n_{i-1}/2 \leq m_{i-1} + 3n/2$, we have $m_i \leq m + 3in/2 = O(m + n \log n)$.

By the above, and since the function $T_{\text{maxflow}}$ is convex, we conclude the total time cost at the $i$-th level is $O\left(T_{\text{maxflow}}(n, m + n\log n)\log(R/\varepsilon)\right)$. Recall that there are $O(\log n)$ levels and therefore the total time is $O\left(T_{\text{maxflow}}(n, m + n\log n)\log(R/\varepsilon)\log n\right)$.

### Exact Parametric Min $s,t$-Cut

In this section we show how Theorem 5.3.2 implies new bounds on computing *exact* parametric min $s,t$-cuts in a few interesting settings.

**Integer Polynomial Costs.** Suppose all the parametric costs $c_\lambda(uv)$ are of the form $c_\lambda(uv) = Q_{uv}(\lambda)$, where each $Q_{uv}$ is a (possibly different) constant-degree polynomial with integer coefficients bounded in the absolute value by an integer $U > 0$ and take nonnegative values on $\mathbb{D}$. Recall $Q_{uv}$ can have a positive degree only if $u = s$ or $v = t$. Moreover, if $u = s$, then $Q_{uv}$ is increasing, whereas when $v = t$, then $Q_{uv}$ is decreasing.

Observe that the parametric capacity $c_\lambda(S)$ of any $S$, $s \in S \subseteq V \setminus \{t\}$ is a constant-degree polynomial with integer coefficients bounded by $nU$ in absolute value. The same applies to a difference polynomial $c_\lambda(S) - c_\lambda(S')$ for any two such sets $S, S'$.

It is known that for a constant-degree polynomials $Q$ with integer coefficients bounded by $W$:

- The roots of $Q$ are of absolute value $O(\text{poly}(W))$ (e.g., [168]).
- Any two distinct roots of $Q$ are at least $\Omega(1/\text{poly}(W))$ apart. [119]

This means that by setting $\lambda_{\min} = -R/2$ and $\lambda_{\max} = R/2$ (or slightly less aggressively, if e.g., $R/2 \notin \mathbb{D}$) for a sufficiently large even integer $R = O(\text{poly}\, nU)$ such that $R/2$ exceeds the maximum possible absolute value of a root of any polynomial of the form $c_\lambda(S) - c_\lambda(S')$, we will indeed have $G_{\lambda_{\min}} = G_{\lambda'}$ for all $\lambda' < \lambda_{\min}$ and $G_{\lambda_{\max}} = G_{\lambda'}$ for all $\lambda' > \lambda_{\max}$.

Moreover, assume we compute an $\varepsilon$-approximate parametric min $s,t$-cut $(\Lambda, \tau)$, where $\Lambda = \{\lambda_1, \ldots, \lambda_k\}$. Suppose for some $i$ there exists $\lambda'$, $\max(\lambda_i, \lambda_{i+1} - 2\varepsilon) \leq \lambda' < \lambda_{i+1}$, such that the minimal $s,t$-cut $S_{\lambda'}$ in $G_{\lambda'}$ satisfies $S_{\lambda_i} \subsetneq S_{\lambda'} \subsetneq S_{\lambda_{i+1}}$. Let $\lambda_i^* = \max(\lambda_i, \lambda_{i+1} - 2\varepsilon)$. Note that $S_{\lambda_i} = S_{\lambda_i^*}$ by Definition 9.3.12. Since $S_{\lambda'}$ is minimal, $c_\lambda(S_{\lambda_i})(\lambda_i^*) - c_\lambda(S_{\lambda'})(\lambda_i^*) \leq 0$ and $c_\lambda(S_{\lambda_i})(\lambda') - c_\lambda(S_{\lambda'})(\lambda') > 0$. So the polynomial $c_\lambda(S_{\lambda_i}) - c_\lambda(S_{\lambda'})$ is non-zero and has a root in the interval $[\lambda_i^*, \lambda')$. Similarly one can prove that the polynomial $c_\lambda(S_{\lambda'}) - c_\lambda(S_{\lambda_{i+1}})$ is non-zero and has a root in the interval $[\lambda', \lambda_{i+1})$. We conclude that the product polynomial $[c_\lambda(S_{\lambda_i}) - c_\lambda(S_{\lambda'})] \cdot [c_\lambda(S_{\lambda'}) - c_\lambda(S_{\lambda_{i+1}})]$ is non-zero, has constant degree, has integer coefficients of order $O(\text{poly}\, nU)$, and has two distinct roots in the interval $[\lambda_i^*, \lambda_{i+1})$, i.e., less than $2\varepsilon$ apart. Therefore, if we set $\varepsilon$ so that $1/\varepsilon$ is a sufficiently large integer but still polynomial in $nU$, the assumption $S_{\lambda_i} \subsetneq S_{\lambda'} \subsetneq S_{\lambda_{i+1}}$ leads to a contradiction. As a result, for all such $\lambda'$, $S_{\lambda'}$ equals either $S_{\lambda'}$ or $S_{\lambda''}$.

In other words, computing an $\varepsilon$-approximate min $s,t$-cut $(\Lambda, \tau)$, where $\Lambda = \{\lambda_1, \ldots, \lambda_k\}$, gives as the structure of all possible minimal min $s,t$-cuts $S_\lambda$ in the following sense. Suppose $\Lambda^* = \{\lambda_1^*, \ldots, \lambda_l^*\}$ is an *exact* parametric min $s,t$-cut. Then $k = l$ and $S(\lambda_i) = \{v \in V : \tau(v) \leq \lambda_i) = S_{\lambda_i^*}$ for all $i = 1, \ldots, k$. To compute $\lambda_i^*$, it is hence enough to find the *unique* $\lambda_i^* \in (\lambda_{i-1}, \lambda_i]$ such that $c_\lambda(S(\lambda_{i-1}))(\lambda_i^*) = c_\lambda(S(\lambda_i))(\lambda_i^*)$ which boils down to solving a polynomial equation of constant degree. It is well known that such equations can be solved exactly in constant time for degrees at most 4.

Observe that if we run our $\varepsilon$-approximate parametric min $s,t$-cut algorithm with $\lambda_{\min}, \lambda_{\max}, \varepsilon$ set as described above, maximum flow is always invoked on some minor $H$ of $G_\lambda$ for $\lambda$ that is an integer multiple of $\varepsilon$. Since, $1/\varepsilon$ is an integer, by multiplying edge costs in $H$ by $1/\varepsilon$, we only need a maximum flow algorithm that can handle integer edge capacities of order $O(\text{poly}(nU))$. By plugging in the best-known algorithms for computing max flow with integral capacities, we obtain the following.

**Theorem 9.3.13.** *Let $G$ be a graph whose parameterized capacities are constant-degree polynomials with integer coefficients in $[-U, U]$. The structure of parametric min $s,t$-cut on $G$ can be computed in:*

- $O(m \cdot \min(m^{1/2}, n^{2/3}) \cdot \text{polylog}\{n, U\})$ *time using a combinatorial algorithm [75],*

- $O((m + n^{1.5}) \cdot \text{polylog}\{n, U\})$ *time using the algorithm of [25],*
- $O(m^{1.497} \text{polylog}\{n, U\})$ *time using the algorithm of [73].*

*The cut function can be found exacly in additional* $O(n)$ *time if the degrees of capacity polynomials are at most* 4.

**Discrete Domains.** Let us now consider the case when $\mathbb{D}$ is discrete and has $\ell$ elements. Suppose all parametric costs are arbitrary functions meeting the requirements of the parametric min $s, t$-cut problem. Then, we can make the $\varepsilon$-approximate algorithm exact by employing the following simple modifications. We start with $\lambda_{\min} = \min \mathbb{D}$ and $\lambda_{\max} = \max \mathbb{D}$. In the binary-search like step, we always choose the middle element of $\mathbb{D} \cap [\lambda_1, \lambda_2]$ as the next pivot. This way, all the max-flow computations are performed on minors of $G_\lambda$ for $\lambda \in \mathbb{D}$.

**Theorem 9.3.14.** *Let* $G$ *be a graph with arbitrary parameterized capacities* $\mathbb{D} \to \mathbb{R}$ *for a discrete domain* $\mathbb{D} \subseteq \mathbb{R}$*, where* $\ell = |\mathbb{D}|$*. Let* $T_{\text{maxflow}}(n', m')$ *be defined as in Theorem 5.3.2. Then exact parametric min* $s, t$*-cut on* $G$ *can be computed in* $O(T_{\text{maxflow}}(n, m \log n) \cdot \log \ell \cdot \log n)$ *time.*

**Planar Graphs.** By Remark 9.3.7, all the max-flow computations in the algorithm of Theorem 5.3.2 are performed on minors of $G$. As a result, if the input graph $G$ is planar, we can use state-of-the-art planar max $s, t$-flow algorithms to obtain better bounds on the parametric min-$s, t$-cut algorithms on planar graphs. Since maximum $s, t$-flow for planar graphs can be computed in $O(n \log n)$ time even for real capacities [22, 58], planar parametric min $s, t$-cut can be solved exactly:

- in $O(n \, \text{polylog}\{n, U\})$ time when parameterized capacities are constant degree polynomials with integer coefficients in $[-U, U]$,
- in $O(n \log^3 n \log \ell)$ time for discrete domains $\mathbb{D} \subseteq \mathbb{R}$ of size $\ell$.

What may be surprising, our reduction is powerful enough to obtain an interesting subquadratic *strongly polynomial* exact algorithm computing parametric min $s, t$-cut in a planar graph with capacity functions that are arbitrary polynomials of degree no more than 4 and real coefficients.

We now sketch this algorithm. It is based on the *parametric search* technique [123] (see also [3]). Suppose we want to solve some decision problem $\mathcal{P}(\alpha)$, where $\alpha \in \mathbb{R}$, such that if $\mathcal{P}(\alpha_0)$ is a yes instance, then $\mathcal{P}(\alpha')$ for all $\alpha'' < \alpha_0$ is also a yes instance. We wish to find the maximum $\alpha^*$ for which $\mathcal{P}(\alpha^*)$ is a yes instance. An example problem $\mathcal{P}(\alpha)$ could be *"does an* $s, t$*-flow of value* $\alpha$ *exist in* $G$*?"*. Then, $\alpha^*$ clearly equals the maximum flow in $G$.

Suppose we have an efficient strongly polynomial algorithm solving the decision problem. Then, in practice one could find $\alpha^*$ via binary search given some initial interval containing $\alpha^*$; however, in general this would not lead to an exact algorithm for real values of $\alpha^*$. Parametric search is a technique for converting a strongly polynomial *parallel* decision algorithm into a *sequential or parallel* strongly polynomial *optimization* algorithm as explained above. The only requirement to keep in mind is that the decision algorithm is governed by comparisons, each of which amounts to testing the sign of some low-degree (say, no more than 4) polynomial in $\alpha$. Specifically, suppose we have a parallel decision algorithm $\mathcal{A}$ that uses $W_{\mathcal{A}}$ work and $D_{\mathcal{A}}$ depth, and also a (possibly the same) another parallel decision algorithm $\mathcal{B}$ with $W_{\mathcal{B}}$ work and $D_{\mathcal{B}}$ depth. Suppose for simplicity all these quantities are polynomial in $n$. Then, parametric search yields a strongly-polynomial optimization algorithm computing $\alpha^*$ in $\tilde{O}(D_{\mathcal{A}} \cdot W_{\mathcal{B}} + W_{\mathcal{A}})$ work and $\tilde{O}(D_{\mathcal{A}} \cdot D_{\mathcal{B}})$ depth.

Now back to planar graphs. We will use parametric search in a nested way. First of all, we will need a decent parallel max flow algorithm for planar graphs. It is well-known (e.g., [58]) that

the decision variant of the max $s,t$-flow problem on planar graphs is reducible to negative cycle detection in the dual graph (which is also planar). There exists a parallel negative cycle detection algorithm on planar graphs with $\tilde{O}(n + n^{3/2}/d^3)$ work and $\tilde{O}(d)$ depth for any $d \geq 1$ [93]. Hence, by using that algorithm as both $\mathcal{A}$ (for $d = D^{3/7}$) and $\mathcal{B}$ (for $d = D^{4/7}$), where $D$ is a parameter, we have $W_{\mathcal{A}} = \tilde{O}(n + n^{3/2}/D^{9/7})$, $D_{\mathcal{A}} = \tilde{O}(D^{3/7})$, $W_{\mathcal{B}} = \tilde{O}(n + n^{3/2}/D^{12/7})$, $D_{\mathcal{B}} = \tilde{O}(D^{4/7})$. So parametric search yields a strongly-polynomial parallel max-flow algorithm for planar graphs with work $\tilde{O}(n + n^{3/2}/D^{9/7})$ and depth $\tilde{O}(D)$ for any $D \geq 1$.

Given a parallel algorithm for max flow in planar graphs, we can use parametric search (instead of binary search) once again when computing the pair $\lambda_1, \lambda_2$ in our recursive algorithm. More specifically, we would like $\lambda_1$ to be the largest such that $|S_{\lambda_1}| \leq n/2$, whereas $\lambda_2$ to be the smallest such that $|S_{\lambda_2}| \geq n/2$. It is easy to see that $\lambda_1$ and $\lambda_2$ are precisely neighboring (or the same) breakpoints of the cut function, i.e., belong to $\Lambda$ from Definition 9.3.11. To actually compute $\lambda_1, \lambda_2$, we use parametric search with $\mathcal{A}$ set to the obtained parallel max-flow algorithm[2], and $\mathcal{B}$ to the best known algorithm that computes a minimum min $s,t$-cut in a planar graph, i.e., a combination of the max-flow algorithm of [22, 58], and linear time graph search. So, in the outer parametric search instance we have$W_{\mathcal{A}} = \tilde{O}(n + n^{3/2}/D^{9/7})$, $D_{\mathcal{A}} = \tilde{O}(D)$, and $W_{\mathcal{B}} = \tilde{O}(n)$. Therefore, the obtained algorithm runs in $\tilde{O}(n^{3/2}/D^{9/7} + Dn)$ sequential time. By setting $D = n^{7/32}$, we obtain $\tilde{O}(n^{1+7/32}) = \tilde{O}(n^{1.21875})$ time.

We stress that all the algorithms [22, 58, 93] used above proceed by only adding and comparing edge weights. Adding polynomials cannot increase their degrees, so indeed when these algorithms are run "generically" for some $\lambda$, the control flow depends only on signs of some small degree polynomials.

**Theorem 9.3.15.** *Let $G$ be a planar graph whose parameterized capacities are all polynomials of degree at most 4 with real coefficients. There exists a strongly polynomial algorithm computing parametric min $s,t$-cut in $G$ exactly in $\tilde{O}(n^{1+7/32})$ time.*

### 9.3.4 Removing Assumptions on $F_i$

In this section we argue why the assumptions on the functions $F_i$ introduced in Section 5.2 are valid without loss of generality. More precisely, we assumed that for all $i$, $F_i(\emptyset) = F_i(V_i) = 0$ and $F_i(S) \geq 0$ for all $S$. Here we show that a simple preprocessing step can enforce all of these conditions.

Without changing the original problem we can shift each $F_i$ such that it evaluates to 0 on $\emptyset$, by defining $\overline{F}_i(S) = F_i(S) - F_i(\emptyset)$. This only changes $F$ by a constant term without affecting the sets that minimize the parametric problem (5.1).

For each $i$, we use Lemma 9.3.16 to find a point $w_i \in B(\overline{F}_i)$. Using this point, we define $\overline{\overline{F}}_i(S) = \overline{F}_i(S) - w_i(S)$. Since by definition $w_i(V_i) = \overline{F}_i(V_i)$, we have that $\overline{\overline{F}}_i(V_i) = 0$. Also, we have $\overline{\overline{F}}_i(\emptyset) = \overline{F}_i(\emptyset) = 0$. Finally, since $w_i(S) \leq \overline{F}_i(S)$ for all $S$, we also have $\overline{\overline{F}}_i(S) \geq 0$.

[2] Actually, it is computing max-flow followed by a graph search to determine the minimal min $s,t$-cut. However, this latter step does not involve any comparisons on capacities, so its depth can be ignored.

Now we can equivalently rewrite the parametric problem

$$\begin{aligned} F_\alpha(A) &= F(A) + \sum_{j\in A} \psi'_j(\alpha) \\ &= \overline{F}(A) + \sum_{j\in A} \psi'_j(\alpha) + \left(\sum_{i=1}^{m} F_i(\emptyset)\right) \\ &= \overline{\overline{F}}(A) + \sum_{j\in A} \left(\psi'_j(\alpha) + \sum_{i=1}^{r} w_i(j)\right) + \left(\sum_{i=1}^{m} F_i(\emptyset)\right) . \end{aligned}$$

Now we can solve the problem on $\overline{\overline{F}} = \sum_{i=1}^{r} \overline{\overline{F}}_i$ with the parametric penalties $\overline{\overline{\psi}}'_j(\alpha) = \psi'_j(\alpha) + \sum_{i=1}^{r} w_i(j)$, which maintain the validity of Assumption 5.2.2.

To compute a point in the base polytope of a submodular function we use the following folklore lemma, which shows that the running time of our initialization procedure is $O(\sum_{i=1}^{r} |V_i| \cdot \mathrm{EO}_i)$:

**Lemma 9.3.16** ([70])**.** *Let $F : 2^V \to \mathbb{Z}$ be a submodular set function, with $F(\emptyset) = 0$, and let $B(F)$ be its base polytope. Given any $x \in \mathbb{R}^{|V|}$, one can compute*

$$\arg\max_{w\in B(F)} \langle x, w\rangle$$

*using $O(|V|)$ calls to an evaluation oracle for $F$. Furthermore $w$ is integral.*

### 9.3.5 Deferred Proofs

#### Proof of Lemma 5.4.3

We define the primal and dual optima of this problem, which will be useful for the proof.

**Definition 9.3.17** (Graph subproblem minimizers)**.** *Let $\widetilde{x}^*$ be the minimizer of*

$$\min_x g(x) + \phi(x)\,, \tag{9.52}$$

*and $\widetilde{w}^*$ be the minimizer of*

$$\min_{w\in B(G)} \phi^*(-w)\,. \tag{9.53}$$

The main tool that we will use for this proof will be the following two structural statements, which can be extracted from Propositions 4.2 and 8.3 in [15].

**Lemma 9.3.18** ([15])**.** *Consider any submodular function $F : 2^V \to \mathbb{R}$.*

1. *Fix some $x \in \mathbb{R}^n$. For any $w \in \mathbb{R}^n$, $w$ is an optimizer of $\max_{w\in B(F)} \langle w, x\rangle$ if and only if there exists a permutation $\pi$ of $[n]$ such that $x_{\pi_1} \geq x_{\pi_2} \geq \cdots \geq x_{\pi_n}$ and for all $u \in V$ we have*

$$w_u = \begin{cases} F(\{\pi_1\}) & \text{if } u = 1 \\ F(\{\pi_1, \pi_2, \ldots, \pi_u\}) - F(\{\pi_1, \pi_2, \ldots, \pi_{u-1}\}) & \text{if } u \geq 2 \end{cases} .$$

2. *Given a function $\phi$ that satisfies the conditions in Definition 5.2.2, the optimal solution $\widetilde{x}^*$ to the problem*

$$\min_x f(x) + \phi(x)\,,$$

*where $f$ is the Lovász extension of $F$, is given by*

$$\widetilde{x}^*_u = -\inf(\{\lambda \in \mathbb{R} \ : \ u \in S(\lambda)\})$$

*for all $u \in V$, where*

$$S(\lambda) = \operatorname*{argmin}_{S \subseteq V} F(S) + \sum_{u \in S} \phi'_u(-\lambda)\,.$$

Additionally, we present two simple lemmas which will be useful in the proof. The first one upper bounds the $\ell_1$ diameter of a base polytope, and the second one upper bounds the $\ell_1$ norm of the gradient of a function in the base polytope.

**Lemma 9.3.8.** *For any submodular function $F : 2^V \to \mathbb{R}_{\geq 0}$ and $F(S) \leq F_{\max}$ for all $S \subseteq V$, we have that*

$$\max_{w \in B(F)} \|w\|_1 \leq 2nF_{\max}\,.$$

*Proof.* By definition of $B(F)$, for all $u \in V$ we have $w_u \leq F(\{u\}) \leq F_{\max}$, so $\sum_{u \in V : w_u \geq 0} w_u \leq nF_{\max}$. Also, $\sum_{u \in V} w_u = F(V)$, so we conclude that

$$\begin{aligned}
\|w\|_1 &= \sum_{u \in V : w_u \geq 0} w_u - \sum_{u \in V : w_u < 0} w_u \\
&= 2 \sum_{u \in V : w_u \geq 0} w_u - F(V) \\
&\leq 2nF_{\max} - F(V) \\
&\leq 2nF_{\max}
\end{aligned}$$

□

**Lemma 9.3.9.** *For any submodular function $F : 2^V \to \mathbb{R}_{\geq 0}$, $F(S) \leq F_{\max}$ for all $S \subseteq V$, and function $\psi : \mathbb{R}^n \to \mathbb{R}$ satisfying the conditions of Definition 5.2.2 we have that*

$$\max_{w \in B(F)} \|\nabla \psi^*(-w)\|_1 \leq \frac{2nF_{\max}}{\sigma} + \|\nabla \psi^*(0)\|_1\,.$$

*Proof.*

$$\begin{aligned}
\|\nabla \psi^*(-w)\|_1 &\leq \|\nabla \psi^*(-w) - \nabla \psi^*(0)\|_1 + \|\nabla \psi^*(0)\|_1 \\
&\leq \frac{1}{\sigma}\|w\|_1 + \|\nabla \psi^*(0)\|_1 \\
&\leq \frac{2nF_{\max}}{\sigma} + \|\nabla \psi^*(0)\|_1\,,
\end{aligned}$$

where we used the triangle inequality, the $\frac{1}{\sigma}$-smoothness of the $\psi_u^*$'s, and Lemma 9.3.8. □

We are now ready to proceed with the proof.

*Proof of Lemma 5.4.3.* We let $\Lambda = \{\lambda_1, \ldots, \lambda_k\}$, where $\lambda_1 < \cdots < \lambda_k$, and define $S(\lambda)$ to be a minimal set in

$$\operatorname*{argmin}_{S \subseteq V} G(S) + \sum_{u \in S} \phi_u'(-\lambda)$$

for all $\lambda \in \mathbb{R}$. Note that this can be equivalently written as

$$\begin{aligned}
&\operatorname*{argmin}_{S \subseteq V} G(S) + \sum_{u \in S} \max\{0, \phi_u'(-\lambda)\} - \sum_{u \in S} \max\{0, -\phi_u'(-\lambda)\} \\
&= \operatorname*{argmin}_{S \subseteq V} G(S) + \sum_{u \in S} \max\{0, \phi_u'(-\lambda)\} + \sum_{u \in V \setminus S} \max\{0, -\phi_u'(-\lambda)\} - \sum_{u \in V} \max\{0, -\phi_u'(-\lambda)\} \\
&= \operatorname*{argmin}_{S \subseteq V} G(S) + \sum_{u \in S} \max\{0, \phi_u'(-\lambda)\} + \sum_{u \in V \setminus S} \max\{0, -\phi_u'(-\lambda)\} \\
&= \operatorname*{argmin}_{S \subseteq V} c_\lambda^+(S \cup \{s\}),
\end{aligned}$$

by the definition of the parametric capacities $c_\lambda$, where $c_\lambda^+(S \cup \{s\}) = \sum_{\substack{u \in S \cup \{s\} \\ v \in V \setminus (S \cup \{s\})}} c_\lambda(u, v)$. Additionally, we denote $\varepsilon = \frac{1}{3L}$ for convenience. By the second item of Lemma 9.3.18, we know that the minimizer of $\min_x f(x) + \phi(x)$ is defined as

$$\widetilde{x}_u^* = -\inf\{\lambda \in \mathbb{R} \ : \ u \in S(\lambda)\}.$$

For all $u \in V$, let $i_u = \operatorname*{argmin}_{i \in [k]} \{\lambda_i \mid u \in S(\lambda_i)\}$ and $\widetilde{x}_u = -\lambda_{i_u}$. We will first prove that $\|\widetilde{x} - \widetilde{x}^*\|_\infty \leq \varepsilon$. Now, by definition we have that $\widetilde{x}_u^* \geq \widetilde{x}_u$. Additionally, setting $\lambda_0 = -\infty$ for convenience, we have $u \notin S(\lambda_{i_u - 1})$, and $u \in S(-\widetilde{x}_u^*)$, so $S(\lambda_{i_u - 1}) \subset S(-\widetilde{x}_u^*)$. By the first item of Definition 9.3.12, this implies that

$$-\widetilde{x}_u^* \geq \lambda_{i_u} - \varepsilon = -\widetilde{x}_u - \varepsilon \Leftrightarrow \widetilde{x}_u^* \leq \widetilde{x}_u + \varepsilon.$$

Therefore, we have concluded that $|\widetilde{x}_u - \widetilde{x}_u^*| \leq \varepsilon$ for all $u \in V$, i.e. $\|\widetilde{x} - \widetilde{x}^*\|_\infty \leq \varepsilon$.

We define a dual solution $\widehat{w} = -\nabla \phi(\widetilde{x})$. We will show that $\widetilde{w}^*$ can be retrieved by rounding $\widehat{w}$. Using the fact that $\phi_u$'s are $L$-smooth and the optimality condition $\widetilde{w}^* = -\nabla \phi(\widetilde{x}^*)$ from Lemma 9.3.10, we get that

$$\|\widehat{w} - \widetilde{w}^*\|_\infty = \|\nabla \phi(\widetilde{x}) - \nabla \phi(\widetilde{x}^*)\|_\infty \leq L \|\widetilde{x} - \widetilde{x}^*\|_\infty \leq L\varepsilon = 1/3.$$

On the other hand, by optimality of $\widetilde{w}^*$, it is a maximizer of $\max_{w \in B(G)} \langle w, \widetilde{x}^* \rangle$. By the first item of Lemma 9.3.18, there exists a permutation $\pi_1, \ldots, \pi_n$ of $V$ such that $\widetilde{w}_u^* = G(\{\pi_1, \ldots, \pi_u\}) - G(\{\pi_1, \ldots, \pi_{u-1}\})$ for $u \in V$. As $G$ takes integral values, we have $\widetilde{w}_u^* \in \Delta \cdot \mathbb{Z}$ for all $u \in V$, and since $|\widehat{w}_u - \widetilde{w}_u^*| < 1/2$, we can exactly recover $\widetilde{w}^*$ by rounding each entry of $\widehat{w}$ to the closest integer.

Our next goal is to compute a $G$-decomposition of $\widetilde{w}^*$, which we will do by computing an exact primal solution and then again applying the first item of Lemma 9.3.18. Given $\widetilde{w}^*$, we can easily

recover the primal optimum $\widetilde{x}^* = \nabla\phi(-\widetilde{w}^*)$. In order to recover a decomposition $\widetilde{w}^* = \sum_{i=1}^{r} \widetilde{w}^{*i}$, we use the well-known fact [54] that

$$\max_{w \in B(G)} \langle w, x \rangle = \max_{w^i \in B(G_i)} \sum_{i=1}^{r} \langle w^i, x \rangle \,,$$

so for any $i \in [r]$, $\widetilde{w}^{*i}$ necessarily maximizes

$$\max_{w^i \in B(G_i)} \langle w^i, \widetilde{x}^* \rangle \,.$$

Therefore, by the first item of Lemma 9.3.18, $\widetilde{w}^{*i}$ can be recovered by sorting the entries of $\widetilde{x}^*$ in decreasing order, such that $\widetilde{x}^*_{\pi_1} \geq \widetilde{x}^*_{\pi_2} \geq \cdots \geq \widetilde{x}^*_{\pi_n}$ for some permutation $\pi$ of $V$, and then setting

$$\widetilde{w}^{*i}_u = G_i(\{\pi_1, \ldots, \pi_u\}) - G_i(\{\pi_1, \ldots, \pi_{u-1}\}) \tag{9.54}$$

for all $u \in V$. Note that $\widetilde{w}^{*i}$'s are in $\mathbb{Z}^n$.

The runtime is dominated by the computation of the decomposition in (9.54), which involves computing prefix cuts for each $G_i$ and by Lemma 9.3.10 takes time $O\left(\sum_{i=1}^{r} |V_i|^2\right)$. Therefore, the total runtime is $O\left(n + \sum_{i=1}^{r} |V_i|^2\right)$. □

**Lemma 9.3.10** (Computing all prefix cut values)**.** *Given a graph $G(V, E, c)$ with $V = \{1, 2, \ldots, n\}$, we can compute the values $c^+([u])$ for all $u \in [n]$ in time $O(n^2)$.*

*Proof.* We note that $c^+(\emptyset) = 0$ and for any $u \geq 1$ we have

$$c^+([u]) = c^+([u-1]) + \sum_{v=u+1}^{n} c_{uv} - \sum_{v=1}^{u-1} c_{vu} \,. \tag{9.55}$$

Therefore $c^+([u])$ can be computed in $O(n)$ given $c^+([u-1])$. As we apply (9.55) $n$ times, the total runtime is $O(n^2)$. □

**Proof of Lemma 5.4.4**

We first prove the following lemma, which helps us bound the range of parameters for parametric min $s,t$-cut.

**Lemma 9.3.11.** *Consider a graph $G(V \cup \{s,t\}, E, c \geq 0)$ and a function $\phi(x) = \sum_{u \in V} \phi_u(x_u)$ that satisfies Assumption 5.2.2. Additionally, let $G(S) = c^+(S \cup \{s\})$ for all $S \subseteq V$ be the cut function associated with the graph. For any $\lambda \in \mathbb{R}$, we set $S(\lambda)$ to be the smallest set that minimizes*

$$\min_{S \subseteq V} G(S) + \sum_{u \in S} \phi'_u(-\lambda) \,.$$

*Let $\rho = \max_{u \in V} |\phi'_u(0)|$ and $G_{\max} = \max_{S \subseteq V} G(S)$. Then, $S(\lambda_{\min}) = \emptyset$ and $S(\lambda_{\max}) = V$, where $\lambda_{\min} = -2\frac{\rho + G_{\max}}{\sigma}$ and $\lambda_{\max} = 2\frac{\rho + G_{\max}}{\sigma}$.*

*Proof.* We first note that by the $\sigma$-strong convexity of the $\phi_u$'s, and since $-\lambda_{\min} > 0 > -\lambda_{\max}$, we have that

$$\phi'_u(-\lambda_{\min}) \geq \phi'_u(0) + \sigma|\lambda_{\min}|\,.$$

and

$$\phi'_u(-\lambda_{\max}) \leq \phi'_u(0) - \sigma|\lambda_{\max}|\,.$$

Therefore for any $\emptyset \neq S \subseteq V$ we have

$$\begin{aligned}
G(S) + \sum_{u\in S} \phi'_u(-\lambda_{\min}) &\geq G(S) + \sum_{u\in S} (\phi'_u(0) + \sigma|\lambda_{\min}|) \\
&\geq G(S) - |S|\rho + |S|\sigma|\lambda_{\min}| \\
&= G(S) - |S|\rho + 2|S|\sigma\frac{\rho + G_{\max}}{\sigma} \\
&> G(S) + G_{\max} \\
&\geq G(\emptyset)\,,
\end{aligned}$$

where we used the fact that $G(S) \geq 0$ and $G(\emptyset) \leq G_{\max}$, so $S(\lambda_{\min}) = \emptyset$. Similarly, for any $S \subset V$ we have

$$\begin{aligned}
G(S) + \sum_{u\in S} \phi'_u(-\lambda_{\max}) &= G(S) + \sum_{u\in V} \phi'_u(-\lambda_{\max}) - \sum_{u\in V\setminus S} \phi'_u(-\lambda_{\max}) \\
&\geq G(S) + \sum_{u\in V} \phi'_u(-\lambda_{\max}) - \sum_{u\in V\setminus S} \left(\phi'_u(0) - \sigma|\lambda_{\max}|\right) \\
&\geq G(S) + \sum_{u\in V} \phi'_u(-\lambda_{\max}) + |V\setminus S|\left(\sigma|\lambda_{\max}| - \rho\right) \\
&= G(S) + \sum_{u\in V} \phi'_u(-\lambda_{\max}) + |V\setminus S|\left(2\sigma\frac{\rho + G_{\max}}{\sigma} - \rho\right) \\
&> G(S) + \sum_{u\in V} \phi'_u(-\lambda_{\max}) + G_{\max} \\
&\geq G(V) + \sum_{u\in V} \phi'_u(-\lambda_{\max})\,,
\end{aligned}$$

where we used the fact that $G(S) \geq 0$ and $G(V) \leq G_{\max}$, so $S(\lambda_{\max}) = V$. □

We are now ready for the proof.

*Proof of Lemma 5.4.4.* We first shift the polytope $B(F)$ so that $w$ is translated to 0. Specifically, for all $S \subseteq V$, we let $\widehat{F}(S) = F(S) - w(S)$ and $\widehat{F}_i(S) = F_i(S) - w^i(S)$ for all $i \in [r]$. As we are just subtracting a linear function, $\widehat{F}$ and the $\widehat{F}_i$'s are still submodular functions, and $B(\widehat{F}) = B(F) - w$, $B(\widehat{F}_i) = B(F_i) - w^i$ for all $i \in [r]$. Note that $w^i \in B(F_i)$ implies that the $\widehat{F}_i$'s (and thus also $\widehat{F}$) are non-negative, since

$$\widehat{F}_i(S) = F_i(S) - w(S) \geq 0\,,$$

and additionally $\widehat{F}_i(\emptyset) = F_i(\emptyset) = 0$ and $\widehat{F}_i(V_i) = F_i(V_i) - w^i(V_i) = 0$ for all $i \in [r]$ (also implying $\widehat{F}(\emptyset) = \widehat{F}(V) = 0$).

We run the algorithm from Lemma 9.3.5 on the $\widehat{F}_i$'s to obtain directed graphs $G_i(V, E, c^i \geq 0)$ whose ($V_i$-restricted) cut functions $G_i(S) = c^{i+}(S)$ $\alpha$-approximate $\widehat{F}_i(S)$, where $\alpha = \max_{i\in[r]} \{|V_i|^2/4 +$

$|V_i|\}$. More specifically,

$$\frac{1}{\alpha}\widehat{F}_i(S) \le G_i(S) \le \widehat{F}_i(S) \text{ for all } S \subseteq V_i,\ G_i(V_i) = \widehat{F}_i(V_i) \tag{9.56}$$

and

$$\frac{1}{\alpha}\left(B(F_i) - w^i\right) = \frac{1}{\alpha}B(\widehat{F}_i) \subseteq B(G_i) \subseteq B(\widehat{F}_i) = B(F_i) - w^i\,, \tag{9.57}$$

We also define the graph $G(V, E, c \ge 0)$, where $c = \sum_{i=1}^{r} c^i$ and has cut function $G(S) = \sum_{i=1}^{r} G_i(S)$ for all $S \subseteq V$. Then,

$$\frac{1}{\alpha}\widehat{F}(S) \le G(S) \le \widehat{F}(S) \text{ for all } S \subseteq V,\ G(V) = \widehat{F}(V) \tag{9.58}$$

and

$$\frac{1}{\alpha}\left(B(F) - w\right) \subseteq B(G) \subseteq B(F) - w\,. \tag{9.59}$$

We absorb the linear term that we subtracted from $F$ into the parametric function. Concretely, we define $\phi$ as $\phi(x) = \psi(x) + \langle w, x\rangle$ for all $x \in \mathbb{R}^n$. It is easy to see that $\phi(x)$ is coordinate-wise separable, as $\phi(x) = \sum_{u\in V} \phi_u(x_u)$ where $\phi_u(x_u) = \psi_u(x_u) + w_u x_u$ for all $u \in V$ and that it satisfies Assumption 5.2.2, since $\phi_u''(x_u) = \psi_u''(x_u)$ and

$$|\phi_u'(0)| = |\psi_u'(0) + w_u| \le |\psi_u'(0)| + F(\{u\}) \le |\psi_u'(0)| + F_{\max} = n^{O(1)}\,.$$

Additionally, the Fenchel dual of $\phi_u$ is a shifted version of $\psi_u$, i.e.

$$\phi_u^*(z) = \max_{y\in\mathbb{R}} zy - \phi(y) = \max_{w\in\mathbb{R}} zy - \psi(y) - w_u y = \max_{w\in\mathbb{R}} (z - w_u)y - \psi(y) = \psi_u^*(z - w_u)\,.$$

We will now run the algorithm from Lemma 5.4.3 on graphs $G_i$ and parametric function $\phi$ to obtain a dual solution vector $\widetilde{w}$. We note that, as $F_i$'s and $w^i$'s take integer values, $G_i$'s take integer values too. This algorithm takes a $\frac{1}{3L}$-approximate parametric min $s,t$-cut as input, which we first compute using Theorem 5.3.2, with range of parameters $[\lambda_{\min}, \lambda_{\max}]$ given by Lemma 9.3.11. We have

$$\begin{aligned}
\lambda_{\max} - \lambda_{\min} &= 4\frac{\max_{u\in V}|\phi_u'(0)| + \max_{S\subseteq V} G(S)}{\sigma} \\
&\le 4\frac{\max_{u\in V}|\phi_u'(0)| + \widehat{F}_{\max}}{\sigma} \\
&\le 4\frac{\max_{u\in V}|\phi_u'(0)| + F_{\max} + \|w\|_1}{\sigma} \\
&\le 4\frac{\max_{u\in V}|\phi_u'(0)| + (2n+1)F_{\max}}{\sigma} \\
&= n^{O(1)}\,,
\end{aligned}$$

where we used Lemma 9.3.8 and the fact that the quantities $|\phi_u'(0)|, F_{\max}, \frac{1}{\sigma}$ are bounded by $n^{O(1)}$ (Assumption 5.2.2).

Based on Theorem 5.3.2, the time to obtain the $\frac{1}{3L}$-approximate parametric min $s,t$-cut will be

$$O\left(T_{\text{maxflow}}(n, |E'|\log n)\log\frac{\lambda_{\max}-\lambda_{\min}}{1/3L}\log n\right) = \widetilde{O}\left(T_{\text{maxflow}}\left(n, n+\sum_{i=1}^{r}|V_i|^2\right)\right).$$

So, by applying Lemma 5.4.3, we obtained a dual solution $\widetilde{w} = \sum_{i=1}^{r}\widetilde{w}^i$ for which $\widetilde{w}^i \in B(G_i)$ and

$$\widetilde{w} = \operatorname*{argmin}_{\widetilde{w}\in B(G)} \phi^*(-\widetilde{w}) = \operatorname*{argmin}_{\widetilde{w}\in B(G)} \psi^*(-w-\widetilde{w})\,. \tag{9.60}$$

For all $u \in V$ and $i \in [r]$, we set $w'^i_u = w^i_u + \widetilde{w}^i_u$ and $w'_u = \sum_{i=1}^{r} w'^i_u$. Note that these quantities are still integral. We now prove the two parts of the lemma statement (feasibility and optimality) separately.

**Feasibility.** For any $i \in [r]$ and $S \subseteq V_i$, we have that

$$w'^i(S) = w^i(S) + \widetilde{w}^i(S) \le w^i(S) + G_i(S) \le w^i(S) + \widehat{F}_i(S) = w^i(S) + F_i(S) - w^i(S) = F^i(S)\,,$$

where the second inequality follows from the fact that $\widetilde{w}^{*i} \in G_i$. Similarly, we have

$$w'^i(V_i) = w^i(V_i) + \widetilde{w}^i(V_i) = w^i(V_i) + G_i(V_i) = w^i(V_i) + \widehat{F}_i(V_i) = F_i(V_i)\,.$$

So we conclude that $w'^i \in B(F_i)$ for all $i \in [r]$.

**Optimality.** Let's set $h(z) := \psi^*(-z)$ for all $z \in \mathbb{R}^n$ for notational convenience, so that our goal is to prove that

$$h(w') - h(w^*) \le \left(1-\frac{1}{\alpha}\right)(h(w) - h(w^*))\,.$$

We will prove a slightly different statement where $w'$ is replaced by a solution on the path from $w$ to $w^*$, which is enough because $w'$ is optimal in $w + B(G)$. Concretely, we set $\bar{w} = \frac{1}{\alpha}(w^* - w)$ and instead will prove

$$h(w+\bar{w}) - h(w^*) \le \left(1-\frac{1}{\alpha}\right)(h(w) - h(w^*))\,.$$

Now, $h$ is a convex function, so applying convexity twice we have

$$\begin{aligned} h(w) &\ge h(w+\bar{w}) + \langle \nabla h(w+\bar{w}), -\bar{w}\rangle \\ &= h(w+\bar{w}) - \frac{1}{\alpha}\langle \nabla h(w+\bar{w}), w^* - w\rangle \end{aligned} \tag{9.61}$$

and

$$\begin{aligned} h(w^*) &\ge h(w+\bar{w}) + \langle \nabla h(w+\bar{w}), w^* - w - \bar{w}\rangle \\ &= h(w+\bar{w}) + \frac{\alpha-1}{\alpha}\langle \nabla h(w+\bar{w}), w^* - w\rangle\,. \end{aligned} \tag{9.62}$$

We divide (9.62) by $\alpha - 1$ and then sum it with (9.61), getting

$$h(w) + \frac{1}{\alpha-1}h(w^*) \ge h(w+\bar{w}) + \frac{1}{\alpha-1}h(w+\bar{w})\,.$$

Equivalently,

$$\frac{\alpha}{\alpha-1}\left(h(w+\bar{w})-h(w^*)\right)\leq h(w)-h(w^*)\,.$$

So by rearranging,

$$h(w+\bar{w})-h(w^*)\leq\left(1-\frac{1}{\alpha}\right)\left(h(w)-h(w^*)\right)\,. \tag{9.63}$$

Thus we can equivalently write that

$$\psi^*(-w-\bar{w})-\psi^*(-w^*)\leq\left(1-\frac{1}{\alpha}\right)\left(\psi^*(-w)-\psi^*(-w^*)\right)\,. \tag{9.64}$$

Now, since by (9.59) we have $\frac{1}{\alpha}(B(F)-w)\subseteq B(G)$ and $w^*\in B(F)$, we have $\bar{w}=\frac{1}{\alpha}(w^*-w)\in B(G)$. Combinging the fact that $\widetilde{w}$ is a minimizer of $\min_{\widetilde{w}^*\in B(G)}\psi^*(-w-\widetilde{w}^*)$ with (9.60) and the fact that $\bar{w}\in B(G)$, we have

$$\psi^*(-w')=\psi^*(-w-\widetilde{w})\leq\psi^*(-w-\bar{w})\,.$$

Combining this with (9.64), we obtain the desired claim:

$$\psi^*(-w')-\psi^*(-w^*)\leq\left(1-\frac{1}{\alpha}\right)\left(\psi^*(-w)-\psi^*(-w^*)\right)\,. \tag{9.65}$$

The running time to compute graphs $G_i$ is $O\left(\sum_{i=1}^{r}|V_i|^2\mathcal{O}_i\right)$ and the time to run the algorithm from Lemma 5.4.3 is $\widetilde{O}\left(n+\sum_{i=1}^{r}|V_i|^2\right)$, so the total running time is

$$\widetilde{O}\left(\sum_{i=1}^{r}|V_i|^2\mathcal{O}_i+T_{\text{maxflow}}\left(n,n+\sum_{i=1}^{r}|V_i|^2\right)\right)\,.$$

□

**Algorithm 24** Finding all minimum cuts

1: **function** FindMinCuts($G(V,E,c),\phi,\varepsilon$)
2: $V'=V\cup\{s,t\}$, $E'=E\cup\bigcup_{u\in V}\{(u,t)\}\cup\bigcup_{u\in V}\{(s,u)\}$
3: Define parametric capacities

$$c_\lambda(u,v)=\begin{cases}\max\{0,\phi'_u(-\lambda)\} & \text{if } u\in V, v=t\\ \max\{0,-\phi'_u(-\lambda)\} & \text{if } u=s, v\in V\\ c_{uv} & \text{otherwise}\end{cases}$$

4: Set $(\Lambda,\tau)=$ApxParametricMinCut$(G'(V',E'),c_\lambda,\lambda_{\min}=-n^{O(1)},\lambda_{\max}=n^{O(1)})$
5: Set $\widetilde{w}_u=-\phi'^*_u(-\tau(u))$ for all $u\in V$ **return** $\widetilde{w}$

## 9.4 Appendix for Chapter 6

### 9.4.1 Deferred Proofs

#### Proof of Lemma 6.3.8

*Proof.* $\Phi^t$ is a quadratic restricted on $R^t$

$$\begin{aligned}
&\Phi(y) - \Phi(x) - \nabla\Phi(x)^T(y-x) \\
&= \frac{\rho_2^+}{2}\left(\|y_{R^t}\|_2^2 - \|x_{R^t}\|_2^2 - 2x_{R^t}^T(y_{R^t} - x_{R^t})\right) \\
&= \frac{\rho_2^+}{2}\|y_{R^t} - x_{R^t}\|_2^2 \\
&\in \left[0, \frac{\rho_2^+}{2}\|y-x\|_2^2\right]
\end{aligned}$$

and so for any $x, y$ with $|\mathrm{supp}(y-x)| \le s + s^*$ (resp. $|\mathrm{supp}(y-x)| \le 1$) we have

$$\begin{aligned}
&g(y) - g(x) - \nabla g(x)^T(y-x) \\
&= f(y) - f(x) - \nabla f(x)^T(y-x) + \Phi(y) - \Phi(x) - \nabla\Phi(x)^T(y-x) \\
&\ge \frac{\rho_{s+s^*}^-}{2}\|y-x\|_2^2 \,(\text{resp.} \le \rho_2^+\|y-x\|_2^2\,).
\end{aligned}$$

□

#### Proof of Lemma 6.3.3

*Proof.* By definition, and setting $\tau = \frac{1}{s}$, in each Type 1 iteration we have

$$\begin{aligned}
g(x^t) - g(x^{t+1}) &\ge \tau\left(g(x^t) - \mathrm{opt}\right) \\
\Rightarrow g(x^{t+1}) - \mathrm{opt} &\le (1-\tau)(g(x^t) - \mathrm{opt})
\end{aligned}$$

and in each Type 2 iteration we have

$$g(x^{t+1}) - \mathrm{opt} \le g(x^t) - \mathrm{opt}$$

(since $g$ can only decrease when unregularizing), therefore

$$\begin{aligned}
f(x^T) - \mathrm{opt} &\le g(x^T) - \mathrm{opt} \\
&\le (1-\tau)^{T_1}(g(x^0) - \mathrm{opt}) \\
&\le e^{-\tau T_1}(g(x^0) - \mathrm{opt}) \\
&\le \varepsilon\,,
\end{aligned}$$

where we used the fact that $T_1 = \frac{1}{\tau}\log\frac{g(x^0)-\mathrm{opt}}{\varepsilon}$. □

```python
import numpy as np

def IHT(n, s):
  x = np.zeros(n)
  for _ in range(T):
    x_new = x - eta * grad(x)
    x_new[np.argsort(np.abs(x_new))[:-s]] = 0
    x = x_new
  return x

def RegIHT(n, s):
  x, w = np.zeros(n), np.ones(n)
  for _ in range(T):
        x_new = (1 - 0.5 * w) * x - 0.5 * eta * grad(x)
        x_new[np.argsort(np.abs(x_new))[:-s]] = 0
        reg = np.sum(w * x**2)
        if reg != 0:
            w = w * (1 - c * w * x**2 / reg)
            w[w <= round_th] = 0
        x = x_new
  return x
```

Figure 9-1: Our python implementations of IHT, RegIHT, where grad is the gradient function, n is the total number of features, s is the desired sparsity level, eta is the step size, c is the weight step size, and round_th is the weight rounding threshold, which we set to 0.5. Note that grad(x) = np.dot(A.T, np.dot(A, x) - b) for linear regression and grad(x) = np.dot(A.T, expit(np.dot(A,x)) - b) for logistic regression, where expit is the sigmoid function.

**Proof of Lemma 6.4.1**

*Proof.* We have

$$\begin{aligned}\left(\sqrt{f(x^t)-f(\widetilde{x}^t)}+\sqrt{f(x^*)-f(\widetilde{x}^t)}\right)^2 &\geq \frac{\rho^-}{2}\left(\left\|x^t-\widetilde{x}^t\right\|_2+\left\|x^*-\widetilde{x}^t\right\|_2\right)^2\\ &\geq \frac{\rho^-}{2}\left\|x^t-x^*\right\|_2^2\\ &\geq \frac{\rho^-}{2}\left(\left\|x^*_{S^*\setminus S^t}\right\|_2^2+\left\|x^t_{S^t\setminus S^*}\right\|_2^2\right),\end{aligned}$$

where the first inequality follows by applying strong convexity to lower bound $f(x^t)-f(\widetilde{x}^t)$ and $f(x^*)-f(\widetilde{x}^t)$ combined with the fact that by definition of $\widetilde{x}^t$, $\nabla_{S^t\cup S^*}f(\widetilde{x}^t)=\vec{0}$, and the second is a triangle inequality. □

## 9.5 Appendix for Chapter 7

### 9.5.1 Python implementation

In Figure 9-1 we have python implementations of the IHT and regularized IHT algorithms that we use for our experiments. As can be seen, both implementations are pretty short.

### 9.5.2 Proof of Theorem 7.1.1

**Theorem 7.1.1** (Regularized IHT)**.** *Let $f \in \mathbb{R}^n \to \mathbb{R}$ be a convex function that is $\beta$-smooth and $\alpha$-strongly convex, with condition number $\kappa = \beta/\alpha$, and $\boldsymbol{x}^*$ be an (unknown) $s$-sparse solution with support $S^*$. Then, running Algorithm 16 with $\eta = (2\beta)^{-1}$ and $c = s'/(4T)$ for*

$$T = O\left(\kappa \log \frac{f(\boldsymbol{x}^0) + (\beta/2)\left\|\boldsymbol{x}^0\right\|_2^2 - f(\boldsymbol{x}^*)}{\varepsilon}\right)$$

*iterations starting from an arbitrary $s' = O(s\kappa)$-sparse solution $\boldsymbol{x}^0$, the algorithm returns an $s'$-sparse solution $\boldsymbol{x}^T$ such that $f(\boldsymbol{x}^T) \leq f(\boldsymbol{x}^*) + \varepsilon$. Furthermore, each iteration requires $O(1)$ evaluations of $f$, $\nabla f$, and $O(n)$ additional time.*

*Proof.* We repeatedly apply Lemma 7.3.1 for

$$T = 64(\kappa+1)\log \frac{f(\boldsymbol{x}^0) + (\beta/2)\left\|\boldsymbol{x}^0\right\|_2^2 - f(\boldsymbol{x}^*)}{\varepsilon}$$

iterations. We define the regularized function

$$g^t(\boldsymbol{x}) := f(\boldsymbol{x}) + (\beta/2)\left\|\boldsymbol{x}\right\|_{\boldsymbol{w}^t,2}^2 ,$$

where $\boldsymbol{w}^t$ are the weights before iteration $t \in [0, T-1]$. Specifically, for each $t$ we apply Lemma 7.3.1 on the current solution $\boldsymbol{x}^t$ and obtain the solution $\boldsymbol{x}^{t+1}$.

Before moving forward, we give an intuitive summary of the proof and the role of Lemma 7.3.1. As long as IHT makes "sufficient" progress on the regularized function $g^t$, this is satisfactory for the original function $f$ as well, because $f(\boldsymbol{x}) \leq g^t(\boldsymbol{x})$ for all $\boldsymbol{x}$. This is the case of the first bullet of Lemma 7.3.1. If it stops making sufficient progress, this means we are at an (approximate) sparse optimum for $g^t$, although it is not necessarily a good sparse solution for $f$, which is the objective we are aiming to minimize. This is where the second bullet of Lemma 7.3.1 comes in, which gives necessary conditions for the above non-progress phenomenon (in other words, a partial characterization of the local minima encountered when running IHT on a regularized function). Specifically, the following condition is central to our approach:

$$\left\|\boldsymbol{x}^t_{S^*}\right\|_{(\boldsymbol{w}^t)^2,2}^2 \geq (4\kappa+6)^{-1}\left\|\boldsymbol{x}^t\right\|_{\boldsymbol{w}^t,2}^2 .$$

We use this condition in the second part of the proof (after Case 2) to motivate a weight update from $\boldsymbol{w}^t$ to $\boldsymbol{w}^{t+1}$, and show that, exactly because of this condition, a lot of the weight decrease is concentrated inside the optimal support $S^*$. As the total weight decrease in $S^*$ is bounded by $s$, this gives a bound on the total number of iterations with insufficient decrease of $g^t$. If not for this condition, we would not be able to bound the number of such iterations and would have potentially remained forever stuck at a local minimum.

Now we are ready to move to the technical proof. In order to make sure that $g^{t+1}(\boldsymbol{x}^{t+1}) \leq g^{t+1}(\boldsymbol{x}^t)$, we revert to the previous solution if the one returned by Lemma 7.3.1 has a larger value of $g^{t+1}$. This is exactly what the conditional in Algorithm 16 is for. The property that $g^{t+1}(\boldsymbol{x}^{t+1}) \leq g^{t+1}(\boldsymbol{x}^t)$ is only used in the very last part of the proof.

Let us assume that $g^t(\boldsymbol{x}^t) > f(\boldsymbol{x}^*)$ at all times, as otherwise the statement holds by the fact that $g^t(\boldsymbol{x}^t)$ is non-increasing as a function of $t$ and upper bounds $f(\boldsymbol{x}^t)$ for all $t$. We have $g^{t+1}(\boldsymbol{x}^{t+1}) \leq g^t(\boldsymbol{x}^t)$ by the fact that $\boldsymbol{w}^{t+1} \leq \boldsymbol{w}^t$ and $g^t(\boldsymbol{x}^{t+1}) \leq g^t(\boldsymbol{x}^t)$ by the guarantees of Lemma 7.3.1.

If the first bullet of Lemma 7.3.1 holds, we have that the value of $g$ decreases considerably on iteration $t$, i.e.

$$\begin{aligned}
&g^{t+1}(\boldsymbol{x}^{t+1}) \\
&\leq g^t(\boldsymbol{x}^{t+1}) \\
&\leq g^t(\boldsymbol{x}^t) - (16\kappa)^{-1}(g^t(\boldsymbol{x}^t) - f(\boldsymbol{x}^*))\,.
\end{aligned}$$

Let us call these iterations *progress iterations*, and the other ones (where the second bullet of Lemma 7.3.1 holds) *weight iterations*. Now, since $g^t(\boldsymbol{x}^t)$ is non-increasing as a function of $t$, after $16\kappa \log \frac{g^0(\boldsymbol{x}^0)-f(\boldsymbol{x}^*)}{\varepsilon}$ progress iterations we will have

$$f(\boldsymbol{x}^t) \leq g^t(\boldsymbol{x}^t) \leq f(\boldsymbol{x}^*) + \varepsilon\,,$$

and so we will be done. From now on let us assume this is not the case, so there are at least

$$T - 16\kappa \log \frac{g^0(\boldsymbol{x}^0) - f(\boldsymbol{x}^*)}{\varepsilon} \geq 3T/4$$

weight iterations.

We remind that in each weight iteration, we have

$$\left\|\boldsymbol{x}^t_{S^*}\right\|^2_{(\boldsymbol{w}^t)^2,2} \geq (4\kappa+6)^{-1}\left\|\boldsymbol{x}^t\right\|^2_{\boldsymbol{w}^t,2} \tag{9.66}$$

$$(\beta/2)\left\|\boldsymbol{x}^t_{S^*}\right\|^2_{(\boldsymbol{w}^t)^2,2} \geq (8\kappa+8)^{-1}\left(g^t(\boldsymbol{x}^t) - f(\boldsymbol{x}^*)\right)\,. \tag{9.67}$$

In words, (9.66) roughly implies that at least an $\Omega(1/\kappa)$ fraction of the mass of $(\boldsymbol{w}^t\boldsymbol{x}^t)^2$ lies inside $S^*$. Therefore, if we decrease $\boldsymbol{w}^t$ by a quantity proportional to $(\boldsymbol{w}^t\boldsymbol{x}^t)^2$, the total sum of weights will decrease at most $O(\kappa)$ times faster than the sum of weights inside $S^*$. As the latter quantity can only decrease by $s$ overall, the total decrease of weights will be $O(s\kappa)$.

Concretely, after each iteration we update the regularization weights as follows:

$$\boldsymbol{w}^{t+1} = \left(\boldsymbol{w}^t - c\cdot(\boldsymbol{w}^t\boldsymbol{x}^t)^2/\left\|\boldsymbol{x}^t\right\|^2_{\boldsymbol{w}^t,2}\right)_{\geq 1/2}\,,$$

for some $c > 0$ to be determined later. First of all, note that the weights are non-increasing. Now, if not for the thresholding operation, it is easy to see that the total weight decrease is at most $c$. The thresholding operation can only double this weight decrease to $2c$. Concretely, for all $t$ we define a vector $\bar{\boldsymbol{w}}^t$ such that

$$\bar{w}^t_i = \begin{cases} w^t_i & \text{if } w^t_i \geq 1/2 \\ 1/2 & \text{if } w^t_i = 0\,. \end{cases}$$

Clearly, $\frac{1}{2}\left(\mathbf{1}-\boldsymbol{w}^t\right) \leq \mathbf{1}-\bar{\boldsymbol{w}}^t \leq \mathbf{1}-\boldsymbol{w}^t$. Now, we have

$$\begin{aligned}
&\left\|\bar{\boldsymbol{w}}^t\right\|_1 - \left\|\bar{\boldsymbol{w}}^{t+1}\right\|_1 \\
&\leq c\left\|\boldsymbol{x}^t\right\|^2_{(\boldsymbol{w}^t)^2,2} / \left\|\boldsymbol{x}^t\right\|^2_{\boldsymbol{w}^t,2} \\
&\leq c\,,
\end{aligned}$$

and, summing up for all $t$ we get

$$\left\|\mathbf{1} - \bar{\boldsymbol{w}}^T\right\|_1 = \left\|\bar{\boldsymbol{w}}^0\right\|_1 - \left\|\bar{\boldsymbol{w}}^T\right\|_1 \leq cT\,.$$

Therefore,

$$\left\|\mathbf{1} - \boldsymbol{w}^T\right\|_1 \leq 2\left\|\mathbf{1} - \bar{\boldsymbol{w}}^T\right\|_1 \leq 2cT\,,$$

and so $\left\|\boldsymbol{w}^T\right\|_1 \geq n - 2cT$.

Therefore, the condition $\left\|\boldsymbol{w}^t\right\|_1 \geq n - s'/2$ of Lemma 7.3.1 is satisfied for all $t$ as long as $c \leq s'/(4T)$. In order to bound the number of iterations, we distinguish two cases for the sum of weights inside $S^*$.

**Case 1:** The sum of weights inside $S^*$ decreases by $\geq 4s/T$.

This case cannot happen more than $T/4$ times since the sum of weights inside $S^*$ can only decrease by $s$ in total. Therefore, case 2 below happens at least $T/2$ times.

**Case 2:** The sum of weights inside $S^*$ decreases by $< 4s/T$.

Note that the decrease in the sum of weights in $S^*$ is exactly equal to

$$\sum_{i\in S^*} \begin{cases} c\cdot (w_i^t x_i^t)^2 / \left\|\boldsymbol{x}^t\right\|^2_{\boldsymbol{w}^t,2} & \text{if this is} \leq w_i^t - 1/2 \\ w_i^t & \text{otherwise}\,. \end{cases}$$

Let $T^*$ be the set of indices $i \in S^*$ for which the second case is true, i.e.

$$c\cdot (w_i^t x_i^t)^2 / \left\|\boldsymbol{x}^t\right\|^2_{\boldsymbol{w}^t,2} > w_i^t - 1/2\,.$$

The total weight decrease from elements in $S^*\backslash T^*$ is then

$$\begin{aligned} &\sum_{i\in S^*\backslash T^*} c\cdot (w_i^t x_i^t)^2 / \left\|\boldsymbol{x}^t\right\|^2_{\boldsymbol{w}^t,2} \\ &= c\left\|\boldsymbol{x}^t_{S^*\backslash T^*}\right\|^2_{(\boldsymbol{w}^t)^2,2} / \left\|\boldsymbol{x}^t\right\|^2_{\boldsymbol{w}^t,2} \\ &\geq \frac{c}{4\kappa+6}\left\|\boldsymbol{x}^t_{S^*\backslash T^*}\right\|^2_{(\boldsymbol{w}^t)^2,2} / \left\|\boldsymbol{x}^t_{S^*}\right\|^2_{(\boldsymbol{w}^t)^2,2}\,, \end{aligned}$$

where we used (9.66). As we have assumed that this decrease is less than $4s/T$, we have that

$$\begin{aligned} \left\|\boldsymbol{x}^t_{T^*}\right\|^2_{(\boldsymbol{w}^t)^2,2} &= \left\|\boldsymbol{x}^t_{S^*}\right\|^2_{(\boldsymbol{w}^t)^2,2} - \left\|\boldsymbol{x}^t_{S^*\backslash T^*}\right\|^2_{(\boldsymbol{w}^t)^2,2} \\ &\geq \left(1 - \frac{4s(4\kappa+6)}{cT}\right)\left\|\boldsymbol{x}^t_{S^*}\right\|^2_{(\boldsymbol{w}^t)^2,2} \\ &\geq (1/2)\left\|\boldsymbol{x}^t_{S^*}\right\|^2_{(\boldsymbol{w}^t)^2,2}\,, \end{aligned} \tag{9.68}$$

as long as $c \geq 8s(4\kappa+6)/T$. We can pick such a $c$ as long as

$$8s(4\kappa+6)/T \leq c \leq s'/(4T) \Leftrightarrow s' \geq 32(4\kappa+6)s\,.$$

Now, to deal with the fact that the sum weights in $T^*$ might not decrease sufficiently, note that all

the weights in $T^*$ are being set to 0, i.e. $w_i^{t+1} = 0$ for all $i \in T^*$. Together with (9.68) and (9.67) this means that we can make significant progress in function value. To see this, note that

$$\begin{aligned}
&g^{t+1}(\boldsymbol{x}^{t+1}) \\
&\leq g^{t+1}(\boldsymbol{x}^t) \\
&\leq g^t(\boldsymbol{x}^t) - (\beta/2)\left\|\boldsymbol{x}^t_{T^*}\right\|^2_{\boldsymbol{w}^t,2} \\
&\leq g^t(\boldsymbol{x}^t) - (\beta/2)\left\|\boldsymbol{x}^t_{T^*}\right\|^2_{(\boldsymbol{w}^t)^2,2} \\
&\leq g^t(\boldsymbol{x}^t) - (\beta/4)\left\|\boldsymbol{x}^t_{S^*}\right\|^2_{(\boldsymbol{w}^t)^2,2} \\
&\leq g^t(\boldsymbol{x}^t) - (16\kappa+16)^{-1}\left(g^t(\boldsymbol{x}^t) - f(\boldsymbol{x}^*)\right) ,
\end{aligned}$$

which can happen at most

$$16(\kappa+1)\log\frac{g^0(\boldsymbol{x}^0) - f(\boldsymbol{x}^*)}{\varepsilon} \leq T/4$$

times.

□

### 9.5.3 Proof of Lemma 7.3.1

**Lemma 7.3.1** (Regularized IHT step progress)**.** *Let $f \in \mathbb{R}^n \to \mathbb{R}$ be a convex function that is $\beta$-smooth and $\alpha$-strongly convex, $\kappa = \beta/\alpha$ be its condition number, and $\boldsymbol{x}^*$ be any $s$-sparse solution.*

*Given any $s'$-sparse solution $\boldsymbol{x} \in \mathbb{R}^n$ where*

$$s' \geq (128\kappa + 2)s$$

*and a weight vector $\boldsymbol{w} \in (\{0\} \cup [1/2, 1])^n$ such that $\|\boldsymbol{w}\|_1 \geq n - s'/2$, we make the following update:*

$$\boldsymbol{x}' = H_{s'}\left((\mathbf{1} - 0.5\boldsymbol{w})\boldsymbol{x} - (2\beta)^{-1}\nabla f(\boldsymbol{x})\right) .$$

*Then, at least one of the following two conditions holds:*

- *Updating $\boldsymbol{x}$ makes regularized progress:*

  $$g(\boldsymbol{x}') \leq g(\boldsymbol{x}) - (16\kappa)^{-1}(g(\boldsymbol{x}) - f(\boldsymbol{x}^*)) ,$$

  *where*

  $$g(\boldsymbol{x}) := f(\boldsymbol{x}) + (\beta/2)\|\boldsymbol{x}\|^2_{\boldsymbol{w},2}$$

  *is the $\ell_2$-regularized version of $f$ with weights given by $\boldsymbol{w}$. Note: The regularized progress statement is true as long as $\boldsymbol{x}$ is suboptimal, i.e. $g(\boldsymbol{x}) > f(\boldsymbol{x}^*)$. Otherwise, we just have $g(\boldsymbol{x}') \leq g(\boldsymbol{x})$.*

- *$\boldsymbol{x}$ is significantly correlated to the optimal support $S^* := \operatorname{supp}(\boldsymbol{x}^*)$:*

  $$\|\boldsymbol{x}_{S^*}\|^2_{\boldsymbol{w}^2,2} \geq (4\kappa+6)^{-1}\|\boldsymbol{x}\|^2_{\boldsymbol{w},2} ,$$

  *and the regularization term restricted to $S^*$ is non-negligible:*

  $$(\beta/2)\|\boldsymbol{x}_{S^*}\|^2_{\boldsymbol{w}^2,2} \geq (8\kappa+8)^{-1}(g(\boldsymbol{x}) - f(\boldsymbol{x}^*)) .$$

*Proof.* By using the fact that $f$ is $\beta$-smooth, and so $g$ is $2\beta$-smooth due to $\boldsymbol{w} \leq \mathbf{1}$, for any $\boldsymbol{x}'$ we obtain

$$g(\boldsymbol{x}') - g(\boldsymbol{x}) \leq \left\langle \nabla g(\boldsymbol{x}), \boldsymbol{x}' - \boldsymbol{x} \right\rangle + \beta \left\| \boldsymbol{x}' - \boldsymbol{x} \right\|_2^2 . \tag{9.69}$$

We let $S$ be the support of $\boldsymbol{x}$ and $S'$ the support of $\boldsymbol{x}'$, i.e. the set of $s'$ indices of the largest magnitude entries of the vector. Since $\nabla g(\boldsymbol{x}) = \nabla f(\boldsymbol{x}) + \beta \boldsymbol{w}\boldsymbol{x}$, we have

$$\bar{\boldsymbol{x}} = \boldsymbol{x} - \eta \nabla g(\boldsymbol{x}) = (\mathbf{1} - 0.5\boldsymbol{w})\boldsymbol{x} - \eta \nabla f(\boldsymbol{x}) ,$$

where $\eta = (2\beta)^{-1}$. We let $A = S' \backslash S$ be the newly inserted entries and $B = S \backslash S'$ be the entries that were just removed from the support. Note that

$$\begin{aligned} \boldsymbol{x}' &= [\boldsymbol{x} - \eta \nabla g(\boldsymbol{x})]_{S'} \\ &= \boldsymbol{x} - \eta \nabla_{S'} g(\boldsymbol{x}) - \boldsymbol{x}_B \\ &= \boldsymbol{x} - \eta \nabla_{S' \cup B} g(\boldsymbol{x}) - \bar{\boldsymbol{x}}_B . \end{aligned}$$

Using (9.69), we have

$$\begin{aligned} & g(\boldsymbol{x}') - g(\boldsymbol{x}) \\ & \leq \langle \nabla g(\boldsymbol{x}), -\eta \nabla_{S' \cup B} g(\boldsymbol{x}) - \bar{\boldsymbol{x}}_B \rangle + \beta \left\| -\eta \nabla_{S' \cup B} g(\boldsymbol{x}) - \bar{\boldsymbol{x}}_B \right\|_2^2 \\ & = -(4\beta)^{-1} \left\| \nabla_{S' \cup B} g(\boldsymbol{x}) \right\|_2^2 + \beta \left\| \bar{\boldsymbol{x}}_B \right\|_2^2 \\ & = -\beta \left\| \eta \nabla_{S \cup A} g(\boldsymbol{x}) \right\|_2^2 + \beta \left\| \bar{\boldsymbol{x}}_B \right\|_2^2 \\ & \leq -\beta \left\| \eta \nabla_{S \cup A'} g(\boldsymbol{x}) \right\|_2^2 + \beta \left\| \bar{\boldsymbol{x}}_{B'} \right\|_2^2 , \end{aligned} \tag{9.70}$$

for any two sets $A' \in [n] \backslash S$ and $B' \subseteq S$ with $|A'| = |B'|$. The latter inequality follows because of the following lemma about IHT:

**Lemma 9.5.1.** *Suppose that we run one step of IHT on vector $\boldsymbol{x}$ supported on $S$ for some function $g$, and let the updated solution vector be $\boldsymbol{x}' = \bar{\boldsymbol{x}}_{(S \cup A) \backslash B}$, where $\bar{\boldsymbol{x}} = \boldsymbol{x} - \eta \nabla g(\boldsymbol{x})$. Then, for any $A' \subseteq [n] \backslash S$ and $B' \subseteq S$ with $|A'| = |B'|$, we have*

$$- \left\| \eta \nabla_A g(\boldsymbol{x}) \right\|_2^2 + \left\| \bar{\boldsymbol{x}}_B \right\|_2^2 \leq - \left\| \eta \nabla_{A'} g(\boldsymbol{x}) \right\|_2^2 + \left\| \bar{\boldsymbol{x}}_{B'} \right\|_2^2 . \tag{9.71}$$

*Proof.* If we denote $|A| = |B| = t$ and $|A'| = |B'| = t'$, then note that by definition of IHT, $A$ are the $t$ largest entries in

$$\left| \bar{\boldsymbol{x}}_{[n] \backslash S} \right| = \eta \left| \nabla_{[n] \backslash S} g(\boldsymbol{x}) \right| ,$$

and $B$ are the $t$ smallest entries in $|\bar{\boldsymbol{x}}_S|$. Similarly, we can assume that $A'$ are the $t'$ largest entries in $\eta \left| \nabla_{[n] \backslash S} g(\boldsymbol{x}) \right|$ and $B'$ are the $t'$ smallest entries in $|\bar{\boldsymbol{x}}_S|$, since this way the right hand side of (9.71) takes its minimum value. If $t' = t$, we are done. We consider two cases:

1. $t' > t$: In this case we have $A' \supseteq A$, $B' \supseteq B$, so

$$\begin{aligned}
&- \|\eta \nabla_{A'} g(\boldsymbol{x})\|_2^2 + \|\bar{\boldsymbol{x}}_{B'}\|_2^2 + \|\eta \nabla_A g(\boldsymbol{x})\|_2^2 - \|\bar{\boldsymbol{x}}_B\|_2^2 \\
&= - \left\|\eta \nabla_{A' \backslash A} g(\boldsymbol{x})\right\|_2^2 + \left\|\bar{\boldsymbol{x}}_{B' \backslash B}\right\|_2^2 \\
&= - \left\|\bar{\boldsymbol{x}}_{A' \backslash A}\right\|_2^2 + \left\|\bar{\boldsymbol{x}}_{B' \backslash B}\right\|_2^2 \\
&\geq (t' - t) \left( - \max_{i \in A' \backslash A} (\bar{x}_i)^2 + \min_{j \in B' \backslash B} (\bar{x}_j)^2 \right) \\
&\geq 0 \,,
\end{aligned}$$

   where the last inequality follows since, by definition of the IHT step, $|\bar{x}_i| \leq |\bar{x}_j|$ for any $i \in A' \backslash A$ and $j \in B' \backslash B$. Otherwise, $i$ would have taken $j$'s place in $S'$.

2. $t' < t$: In this case we have $A' \subseteq A$, $B' \subseteq B$. Similarly to the previous case,

$$\begin{aligned}
&- \|\eta \nabla_{A'} g(\boldsymbol{x})\|_2^2 + \|\bar{\boldsymbol{x}}_{B'}\|_2^2 + \|\eta \nabla_A g(\boldsymbol{x})\|_2^2 - \|\bar{\boldsymbol{x}}_B\|_2^2 \\
&= \left\|\eta \nabla_{A \backslash A'} g(\boldsymbol{x})\right\|_2^2 - \left\|\bar{\boldsymbol{x}}_{B \backslash B'}\right\|_2^2 \\
&= \left\|\bar{\boldsymbol{x}}_{A \backslash A'}\right\|_2^2 - \left\|\bar{\boldsymbol{x}}_{B \backslash B'}\right\|_2^2 \\
&\geq (t - t') \left( \min_{i \in A \backslash A'} (\bar{x}_i)^2 - \max_{j \in B \backslash B'} (\bar{x}_j)^2 \right) \\
&\geq 0 \,,
\end{aligned}$$

   where the last inequality follows since, by definition of the IHT step, $|\bar{x}_i| \geq |\bar{x}_j|$ for any $i \in A \backslash A'$ and $j \in B \backslash B'$. Otherwise $i$ wouldn't have taken $j$'s place in $S'$.

□

Now, let us assume that the first bullet in the lemma statement is false, i.e.

$$g(\boldsymbol{x}') - g(\boldsymbol{x}) > -(16\kappa)^{-1} (g(\boldsymbol{x}) - f(\boldsymbol{x}^*)) \,.$$

Setting $A' = B' = \emptyset$ in (9.70), we get that

$$g(\boldsymbol{x}') - g(\boldsymbol{x}) \leq -\beta \|\eta \nabla_S g(\boldsymbol{x})\|_2^2 \,,$$

so we conclude that

$$\|\nabla_S g(\boldsymbol{x})\|_2^2 < \frac{1}{16\kappa\beta\eta^2} (g(\boldsymbol{x}) - f(\boldsymbol{x}^*)) = \frac{\alpha}{4} (g(\boldsymbol{x}) - f(\boldsymbol{x}^*)) \,. \tag{9.72}$$

Now, we again use (9.70) but we set $A'$ to be the $s$ entries from $[n] \backslash S$ on which $\nabla g(\boldsymbol{x})$ has the largest magnitude, and $B'$ to be the $s$ entries from $S$ on which $\bar{\boldsymbol{x}}$ has the smallest magnitude. Also,

let $R$ be an arbitrary subset of $S \backslash S^*$ with size $r \geq 2s$. We then have

$$\begin{aligned}
&g(\boldsymbol{x}') - g(\boldsymbol{x}) \\
&\leq -(4\beta)^{-1} \left\| \nabla_{S \cup A'} g(\boldsymbol{x}) \right\|_2^2 + \beta \left\| \bar{\boldsymbol{x}}_{B'} \right\|_2^2 \\
&\leq -(4\beta)^{-1} \left\| \nabla_{S \cup S^*} g(\boldsymbol{x}) \right\|_2^2 + \beta \left\| \bar{\boldsymbol{x}}_{B'} \right\|_2^2 \\
&\leq -(4\beta)^{-1} \left\| \nabla_{S \cup S^*} g(\boldsymbol{x}) \right\|_2^2 + \frac{\beta s}{|R \backslash S^*|} \left\| \bar{\boldsymbol{x}}_{R \backslash S^*} \right\|_2^2 \\
&\leq -(4\beta)^{-1} \left\| \nabla_{S \cup S^*} g(\boldsymbol{x}) \right\|_2^2 + \frac{\beta s}{r - s} \left\| \bar{\boldsymbol{x}}_{R \backslash S^*} \right\|_2^2 ,
\end{aligned} \tag{9.73}$$

where we used the fact that

$$\left\| \nabla_{A'} g(\boldsymbol{x}) \right\|_2^2 \geq \left\| \nabla_{S^* \backslash S} g(\boldsymbol{x}) \right\|_2^2$$

by definition of $A'$ (and since $|S^* \backslash S| \leq s$), and the fact that, by definition of $B'$ (and since $|R \backslash S^*| \geq 2s - s = |B'|$),

$$\frac{1}{|B'|} \left\| \bar{\boldsymbol{x}}_{B'} \right\|_2^2 \leq \frac{1}{|R \backslash S^*|} \left\| \bar{\boldsymbol{x}}_{R \backslash S^*} \right\|_2^2 .$$

In fact, we will let $R = \{i \in S \mid w_i > 0\}$ be the set of elements that are being regularized. To lower bound the size $r$ of this set, note that by the guarantee of the lemma statement,

$$\begin{aligned}
n - s'/2 &\leq \left\| \boldsymbol{w} \right\|_1 \\
&\leq n - |\{i \in S \mid w_i = 0\}| \\
&= n - (s' - r) ,
\end{aligned}$$

so $r \geq s'/2$. We conclude that $r \geq 2s$ since $s' \geq 4s$.

Now, because of the fact that $f$ is $\alpha$-strongly convex, we have

$$\begin{aligned}
&f(\boldsymbol{x}^*) - f(\boldsymbol{x}) \\
&\geq \langle \nabla f(\boldsymbol{x}), \boldsymbol{x}^* - \boldsymbol{x} \rangle + (\alpha/2) \left\| \boldsymbol{x}^* - \boldsymbol{x} \right\|_2^2 \\
&= \langle \nabla g(\boldsymbol{x}), \boldsymbol{x}^* - \boldsymbol{x} \rangle - \beta \langle \boldsymbol{w}\boldsymbol{x}, \boldsymbol{x}^* - \boldsymbol{x} \rangle + (\alpha/2) \left\| \boldsymbol{x}^* - \boldsymbol{x} \right\|_2^2 \\
&\geq -\alpha^{-1} \left\| \nabla_{S \cup S^*} g(\boldsymbol{x}) \right\|_2^2 - \beta \langle \boldsymbol{w}\boldsymbol{x}, \boldsymbol{x}^* - \boldsymbol{x} \rangle + (\alpha/4) \left\| \boldsymbol{x}^* - \boldsymbol{x} \right\|_2^2 ,
\end{aligned} \tag{9.74}$$

where we used the inequality

$$\langle \boldsymbol{a}, \boldsymbol{b} \rangle + (\alpha/4) \left\| \boldsymbol{b} \right\|_2^2 \geq -\alpha^{-1} \left\| \boldsymbol{a} \right\|_2^2 .$$

By re-arranging and plugging (9.74) into (9.73), we get

$$\begin{aligned}
&g(\boldsymbol{x}') - g(\boldsymbol{x}) \\
&\leq -(4\kappa)^{-1} \Big( f(\boldsymbol{x}) - f(\boldsymbol{x}^*) - \beta \langle \boldsymbol{w}\boldsymbol{x}, \boldsymbol{x}^* - \boldsymbol{x} \rangle + (\alpha/4) \left\| \boldsymbol{x}^* - \boldsymbol{x} \right\|_2^2 \Big) + \frac{\beta s}{r - s} \left\| \bar{\boldsymbol{x}}_{R \backslash S^*} \right\|_2^2 \\
&= -(4\kappa)^{-1} \Big( g(\boldsymbol{x}) - f(\boldsymbol{x}^*) - \beta \langle \boldsymbol{w}\boldsymbol{x}, \boldsymbol{x}^* - \boldsymbol{x} \rangle - (\beta/2) \left\| \boldsymbol{x} \right\|_{\boldsymbol{w},2}^2 + (\alpha/4) \left\| \boldsymbol{x}^* - \boldsymbol{x} \right\|_2^2 - \frac{4\kappa\beta s}{r - s} \left\| \bar{\boldsymbol{x}}_{R \backslash S^*} \right\|_2^2 \Big) .
\end{aligned} \tag{9.75}$$

Now, note that by definition of $\bar{\boldsymbol{x}}$ we have

$$\begin{aligned}&\left\|\bar{\boldsymbol{x}}_{R\setminus S^*}\right\|_2^2\\&\leq 2\left\|\boldsymbol{x}_{R\setminus S^*}\right\|_2^2+2(2\beta)^{-2}\left\|\nabla_{R\setminus S^*}g(\boldsymbol{x})\right\|_2^2\end{aligned}$$

and, since $w_i \geq 1/2$ for each $i \in R$,

$$\left\|\boldsymbol{x}_{R\setminus S^*}\right\|_2^2 \leq 2\left\|\boldsymbol{x}_{R\setminus S^*}\right\|_{\boldsymbol{w},2}^2 \leq 2\left\|\boldsymbol{x}_{S\setminus S^*}\right\|_{\boldsymbol{w},2}^2 .$$

Therefore,

$$\begin{aligned}&\frac{4\kappa\beta s}{r-s}\left\|\bar{\boldsymbol{x}}_{R\setminus S^*}\right\|_2^2\\&\leq \frac{16\kappa\beta s}{r-s}\left\|\boldsymbol{x}_{S\setminus S^*}\right\|_{\boldsymbol{w},2}^2+\frac{2\kappa s}{\beta(r-s)}\left\|\nabla_{R\setminus S^*}g(\boldsymbol{x})\right\|_2^2\\&\leq \frac{16\kappa\beta s}{r-s}\left\|\boldsymbol{x}_{S\setminus S^*}\right\|_{\boldsymbol{w},2}^2+\frac{s}{2(r-s)}(g(\boldsymbol{x})-f(\boldsymbol{x}^*)),\end{aligned}$$

where the last inequality follows from (9.72) since $R\setminus S^* \subseteq S$. Plugging this back into (9.75), we get

$$\begin{aligned}&g(\boldsymbol{x}')-g(\boldsymbol{x})\\&\leq -(4\kappa)^{-1}\Big(\left(1-\frac{s}{2(r-s)}\right)(g(\boldsymbol{x})-f(\boldsymbol{x}^*))\\&-\beta\langle \boldsymbol{w}\boldsymbol{x},\boldsymbol{x}^*-\boldsymbol{x}\rangle-(\beta/2)\left\|\boldsymbol{x}\right\|_{\boldsymbol{w},2}^2+(\alpha/4)\left\|\boldsymbol{x}^*-\boldsymbol{x}\right\|_2^2-\frac{16\kappa\beta s}{r-s}\left\|\boldsymbol{x}_{S\setminus S^*}\right\|_{\boldsymbol{w},2}^2\Big)\\&=-(4\kappa)^{-1}\Big(\left(1-\frac{s}{2(r-s)}\right)(g(\boldsymbol{x})-f(\boldsymbol{x}^*))\\&\underbrace{-\beta\langle \boldsymbol{w}\boldsymbol{x}_{S\cap S^*},\boldsymbol{x}^*-\boldsymbol{x}\rangle+(\alpha/4)\left\|\boldsymbol{x}^*-\boldsymbol{x}\right\|_2^2}_{\geq -(\kappa\beta)\left\|\boldsymbol{x}_{S\cap S^*}\right\|_{\boldsymbol{w}^2,2}^2}+\beta\left\|\boldsymbol{x}_{S\setminus S^*}\right\|_{\boldsymbol{w},2}^2-(\beta/2)\left\|\boldsymbol{x}\right\|_{\boldsymbol{w},2}^2\underbrace{-\frac{16\kappa\beta s}{r-s}\left\|\boldsymbol{x}_{S\setminus S^*}\right\|_{\boldsymbol{w},2}^2}_{\geq -(\beta/4)\left\|\boldsymbol{x}_{S\setminus S^*}\right\|_{\boldsymbol{w},2}^2}\Big)\\&\leq -(4\kappa)^{-1}\Big(\left(1-\frac{s}{2(r-s)}\right)(g(\boldsymbol{x})-f(\boldsymbol{x}^*))-(\kappa\beta)\left\|\boldsymbol{x}_{S\cap S^*}\right\|_{\boldsymbol{w}^2,2}^2-(\beta/2)\left\|\boldsymbol{x}_{S\cap S^*}\right\|_{\boldsymbol{w},2}^2+(\beta/4)\left\|\boldsymbol{x}_{S\setminus S^*}\right\|_{\boldsymbol{w},2}^2\Big)\\&\leq -(4\kappa)^{-1}\Big(0.5\,(g(\boldsymbol{x})-f(\boldsymbol{x}^*))-(\kappa+1)\beta\left\|\boldsymbol{x}_{S\cap S^*}\right\|_{\boldsymbol{w}^2,2}^2+(\beta/4)\left\|\boldsymbol{x}_{S\setminus S^*}\right\|_{\boldsymbol{w},2}^2\Big),\end{aligned} \tag{9.76}$$

where we used the fact that

$$\frac{s}{2(r-s)} \leq 1/2 ,$$

which holds as long as $s' \geq 4s$, and

$$\frac{16\kappa\beta s}{r-s} \leq \beta/4 ,$$

which holds as long as $r \geq s'/2$ and $s' \geq (128\kappa+2)s$. In the last inequality we also used the property $\boldsymbol{w}/2 \leq \boldsymbol{w}^2$, which is by definition of $\boldsymbol{w}$.

Now, note that, because we have assumed that the first bullet of the statement doesn't hold, it

has to be the case that

$$(1/4)\left(g(\boldsymbol{x}) - f(\boldsymbol{x}^*)\right) - (\kappa+1)\beta \left\|\boldsymbol{x}_{S\cap S^*}\right\|_{\boldsymbol{w}^2,2}^2 + (\beta/4)\left\|\boldsymbol{x}_{S\setminus S^*}\right\|_{\boldsymbol{w},2}^2 \leq 0\,.$$

This immediately implies that

$$\begin{aligned}
&(\kappa+1)\beta \left\|\boldsymbol{x}_{S\cap S^*}\right\|_{\boldsymbol{w}^2,2}^2 \geq (\beta/4)\left\|\boldsymbol{x}_{S\setminus S^*}\right\|_{\boldsymbol{w},2}^2 \\
&\Rightarrow (4\kappa+4)\left\|\boldsymbol{x}_{S\cap S^*}\right\|_{\boldsymbol{w}^2,2}^2 \geq \left\|\boldsymbol{x}_{S\setminus S^*}\right\|_{\boldsymbol{w},2}^2 \\
&\Rightarrow (4\kappa+6)\left\|\boldsymbol{x}_{S\cap S^*}\right\|_{\boldsymbol{w}^2,2}^2 \geq \left\|\boldsymbol{x}\right\|_{\boldsymbol{w},2}^2\,,
\end{aligned}$$

so

$$\left\|\boldsymbol{x}_{S\cap S^*}\right\|_{\boldsymbol{w}^2,2}^2 \geq (4\kappa+6)^{-1}\left\|\boldsymbol{x}\right\|_{\boldsymbol{w},2}^2\,.$$

Similarly we also have

$$\begin{aligned}
&(\kappa+1)\beta \left\|\boldsymbol{x}_{S\cap S^*}\right\|_{\boldsymbol{w}^2,2}^2 \geq (1/4)\left(g(\boldsymbol{x}) - f(\boldsymbol{x}^*)\right) \\
&\Rightarrow (\beta/2)\left\|\boldsymbol{x}_{S\cap S^*}\right\|_{\boldsymbol{w}^2,2}^2 \geq (8\kappa+8)^{-1}(g(\boldsymbol{x}) - f(\boldsymbol{x}^*))\,.
\end{aligned}$$

Therefore the second bullet of the statement is true, and we are done.

□

### 9.5.4 Low Rank Minimization

#### Preliminaries

We will use the following simple lemma about Frobenius products between low rank projections and symmetric PSD matrices. We remind the reader that $H_r\left(\boldsymbol{A}\right)$ is the matrix consisting of the top $r$ components from the singular value decomposition of $\boldsymbol{A}$.

**Lemma 9.5.2.** *For any two symmetric PSD matrices $\boldsymbol{\Pi}, \boldsymbol{A} \in \mathbb{R}^{n\times n}$, where $\operatorname{rank}(\boldsymbol{\Pi}) \leq r$ and $\|\boldsymbol{\Pi}\|_2 \leq 1$, we have that*

$$\left|\langle \boldsymbol{\Pi}, \boldsymbol{A}\rangle\right| \leq \operatorname{Tr}\left[H_r(\boldsymbol{A})\right]\,.$$

*Proof.* We will use the following inequality for singular values

$$\sum_{i=1}^{k}\sigma_i(AB) \leq \sum_{i=1}^{k}\sigma_i(A)\sigma_i(B)$$

for $k = 1,\ldots,n$, $A, B \in \mathbb{R}^{n\times n}$ and $\sigma_1(A) \geq \cdots \geq \sigma_n(A)$ are singular values of matrix $A$ (see page

**Algorithm 25** Regularized Local Search

$\boldsymbol{A}^0$: initial rank-$r'$ solution
$\boldsymbol{W}^0 = \boldsymbol{Y}^0 = \boldsymbol{I}$: initial regularization weights
$\eta$: step size, $T$: #iterations
$c$: weight step size
**for** $t = 0, \ldots, T-1$ **do**
  $\Phi(\boldsymbol{A}) := (\beta/4)\left(\left\langle \boldsymbol{W}^t, \boldsymbol{A}\boldsymbol{A}^\top \right\rangle + \left\langle \boldsymbol{Y}^t, \boldsymbol{A}^\top \boldsymbol{A} \right\rangle\right)$
  $g(\boldsymbol{A}) := f(\boldsymbol{A}) + \Phi(\boldsymbol{A})$
  $\bar{A} = H_{s'-1}\left(\boldsymbol{A}^t\right) - 0.5 H_1\left(\eta \nabla g(\boldsymbol{A}^t)\right)$
  $\boldsymbol{P} = (\boldsymbol{W}^t)^{1/2} \boldsymbol{A}^t (\boldsymbol{A}^t)^\top (\boldsymbol{W}^t)^{1/2}$
  $\boldsymbol{Q} = (\boldsymbol{Y}^t)^{1/2} (\boldsymbol{A}^t)^\top \boldsymbol{A}^t (\boldsymbol{Y}^t)^{1/2}$
  $\Delta = g(\boldsymbol{A}^t) - f(\boldsymbol{A}^*)$

  **if** $g(\boldsymbol{A}^t) - g(\bar{A}) \geq (r')^{-1}\Delta$ **then**
    Let $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ be the SVD of $\bar{A}$
    $\boldsymbol{A}^{t+1} = \underset{\boldsymbol{A}=\boldsymbol{U}\boldsymbol{X}\boldsymbol{V}^\top}{\operatorname{argmin}} g(\boldsymbol{A})$
    $\boldsymbol{W}^{t+1}, \boldsymbol{Y}^{t+1} = \boldsymbol{W}^t, \boldsymbol{Y}^t$

  **else if** $\max\left\{\operatorname{Tr}\left[H_r\left(\boldsymbol{P}\right)\right], \operatorname{Tr}\left[H_r\left(\boldsymbol{Q}\right)\right]\right\} \geq (0.4/\beta)\Delta$ **then**
    $\boldsymbol{A}^{t+1} = \boldsymbol{A}^t$
    $\boldsymbol{W}^{t+1} = (\boldsymbol{W}^t)^{1/2}\left(\boldsymbol{I} - r^{-1}\boldsymbol{\Pi}_{\operatorname{im}(\boldsymbol{P})}\right)(\boldsymbol{W}^t)^{1/2}$
    $\boldsymbol{Y}^{t+1} = (\boldsymbol{Y}^t)^{1/2}\left(\boldsymbol{I} - r^{-1}\boldsymbol{\Pi}_{\operatorname{im}(\boldsymbol{Q})}\right)(\boldsymbol{Y}^t)^{1/2}$

  **else**
    $\boldsymbol{A}^{t+1} = \boldsymbol{A}^t$
    $\boldsymbol{W}^{t+1} = \boldsymbol{W}^t - \frac{\boldsymbol{W}^t \boldsymbol{A}^t (\boldsymbol{A}^t)^\top \boldsymbol{W}^t}{\langle \boldsymbol{W}^t, \boldsymbol{A}^t (\boldsymbol{A}^t)^\top \rangle}$
    $\boldsymbol{Y}^{t+1} = \boldsymbol{Y}^t - \frac{\boldsymbol{Y}^t (\boldsymbol{A}^t)^\top \boldsymbol{A}^t \boldsymbol{Y}^t}{\langle \boldsymbol{Y}^t, (\boldsymbol{A}^t)^\top \boldsymbol{A}^t \rangle}$

177 in [82]). Then

$$\begin{aligned}
|\langle \boldsymbol{\Pi}, \boldsymbol{A} \rangle| &= \operatorname{Tr}\left[\boldsymbol{\Pi}\, \boldsymbol{A}^\top\right] \\
&= \operatorname{Tr}\left[\boldsymbol{\Pi}\, \boldsymbol{A}\right] \\
&= \sum_{i=1}^{n} \sigma_i(\boldsymbol{\Pi}\, \boldsymbol{A}) \\
&\leq \sum_{i=1}^{n} \sigma_i(\boldsymbol{\Pi}) \sigma_i(\boldsymbol{A}) \\
&= \sum_{i=1}^{r} \sigma_i(\boldsymbol{\Pi}) \sigma_i(\boldsymbol{A}) \\
&\leq \sum_{i=1}^{r} \sigma_i(\boldsymbol{A}) \\
&= \operatorname{Tr}\left[H_r(\boldsymbol{A})\right] .
\end{aligned}$$

□

**Analysis**

This section is devoted to proving Theorem 7.1.2, which analyzes an algorithm for low rank optimization that uses adaptive regularization.

**Theorem 7.1.2** (Adaptive Regularization for Low Rank Optimization)**.** *Let $f \in \mathbb{R}^{m \times n} \to \mathbb{R}$ be a convex function with condition number $\kappa$ and consider the low rank minimization problem*

$$\min_{\operatorname{rank}(\boldsymbol{A}) \leq r} f(\boldsymbol{A}) . \tag{9.77}$$

*For any error parameter $\varepsilon > 0$, there exists a polynomial time algorithm that returns a matrix $\boldsymbol{A}$ with $\operatorname{rank}(\boldsymbol{A}) \leq O\left(r\left(\kappa + \log \frac{f(\boldsymbol{O}) - f(\boldsymbol{A}^*)}{\varepsilon}\right)\right)$ and $f(\boldsymbol{A}) \leq f(\boldsymbol{A}^*) + \varepsilon$, where $\boldsymbol{A}^*$ is any rank-$r$ matrix.*

*Proof of Theorem 7.1.2.* Let the smoothness and strong convexity parameters of $f$ be $\beta, \alpha$. We repeatedly apply Lemma 9.5.3 $T \geq O\left(r\kappa \log \frac{f(\boldsymbol{A}^0) + (\beta/2)\left\|\boldsymbol{A}^0\right\|_F^2 - f(\boldsymbol{A}^*)}{\varepsilon}\right)$ times starting from solution $\boldsymbol{A}^0 = \boldsymbol{O}$ and weight matrices $\boldsymbol{W}^0 = \boldsymbol{I}$, $\boldsymbol{Y}^0 = \boldsymbol{I}$. Thus, we obtain solutions $\boldsymbol{A}^0, \ldots, \boldsymbol{A}^T$, and weights $\boldsymbol{W}^0, \boldsymbol{W}^1, \ldots, \boldsymbol{W}^T$ and $\boldsymbol{Y}^0, \boldsymbol{Y}^1, \ldots, \boldsymbol{Y}^T$. We let

$$\begin{aligned}
&g^t(\boldsymbol{A}) \\
&= f(\boldsymbol{A}) + (\beta/4)\left(\langle\, \boldsymbol{W}^t, \boldsymbol{A}^t (\boldsymbol{A}^t)^\top \rangle + \langle\, \boldsymbol{Y}^t, (\boldsymbol{A}^t)^\top \boldsymbol{A}^\top \rangle\right)
\end{aligned}$$

be the regularized function at iteration $t$.

We denote by $T_i$ the total number of iterations for which item $i \in \{1, 2, 3\}$ from the statement of Lemma 9.5.3 holds.

Consider the $T_2$ iterations for which item 2 from the statement of Lemma 9.5.3 holds. Without loss of generality, $\boldsymbol{W}$ is updated at least $T_2/2$ times. Letting $\boldsymbol{A}^* = \boldsymbol{U}^* \boldsymbol{\Sigma}^* \boldsymbol{V}^{*\top}$ be the singular value

decomposition of $\boldsymbol{A}^*$, for each such iteration we have

$$\begin{aligned}&\operatorname{Tr}\left[\boldsymbol{\Pi}_{\operatorname{im}(\boldsymbol{U}^*)}\boldsymbol{W}^{t+1}\boldsymbol{\Pi}_{\operatorname{im}(\boldsymbol{U}^*)}\right]\\&\leq \operatorname{Tr}\left[\boldsymbol{\Pi}_{\operatorname{im}(\boldsymbol{U}^*)}\boldsymbol{W}^{t}\boldsymbol{\Pi}_{\operatorname{im}(\boldsymbol{U}^*)}\right]-(10\kappa)^{-1}\,,\end{aligned}$$

and for all other types of iterations we have $\boldsymbol{W}^{t+1}\preceq\boldsymbol{W}^t$. Therefore,

$$\begin{aligned}&\operatorname{Tr}\left[\boldsymbol{\Pi}_{\operatorname{im}(\boldsymbol{U}^*)}\boldsymbol{W}^{T}\boldsymbol{\Pi}_{\operatorname{im}(\boldsymbol{U}^*)}\right]\\&\leq \operatorname{Tr}\left[\boldsymbol{\Pi}_{\operatorname{im}(\boldsymbol{U}^*)}\boldsymbol{W}^{0}\boldsymbol{\Pi}_{\operatorname{im}(\boldsymbol{U}^*)}\right]-\frac{T_2}{2}(10\kappa)^{-1}\,.\end{aligned}$$

However, note that by the guarantee of Lemma 9.5.3 that $\boldsymbol{W}^T\succeq\boldsymbol{O}$, we have

$$\operatorname{Tr}\left[\boldsymbol{\Pi}_{\operatorname{im}(\boldsymbol{U}^*)}\boldsymbol{W}^{T}\boldsymbol{\Pi}_{\operatorname{im}(\boldsymbol{U}^*)}\right]\geq 0\,,$$

and because $\boldsymbol{W}^0=\boldsymbol{I}$ we also know that

$$\operatorname{Tr}\left[\boldsymbol{\Pi}_{\operatorname{im}(\boldsymbol{U}^*)}\boldsymbol{W}^{0}\boldsymbol{\Pi}_{\operatorname{im}(\boldsymbol{U}^*)}\right]=\operatorname{Tr}\left[\boldsymbol{\Pi}_{\operatorname{im}(\boldsymbol{U}^*)}\right]\leq r\,.$$

This implies that $T_2\leq 20\kappa r$.

Now, if $T_1\geq 16r\kappa\log\frac{g^0(\boldsymbol{A}^0)-f(\boldsymbol{A}^*)}{\varepsilon}$, and since $g^t(\boldsymbol{A}^t)$ is non-increasing for all $t$, we have

$$\begin{aligned}&g^T(\boldsymbol{A}^T)-f(\boldsymbol{A}^*)\\&\leq\left(1-(16r\kappa)^{-1}\right)^{T_1}\left(g^0(\boldsymbol{A}^0)-f(\boldsymbol{A}^*)\right)\\&\leq\varepsilon\,,\end{aligned}$$

so $T_1\leq 16r\kappa\log\frac{g^0(\boldsymbol{A}^0)-f(\boldsymbol{A}^*)}{\varepsilon}$.

Similarly, if $T_3\geq 10r\log\frac{g^0(\boldsymbol{A}^0)-f(\boldsymbol{A}^*)}{\varepsilon}$ we have

$$\begin{aligned}&g^T(\boldsymbol{A}^T)-f(\boldsymbol{A}^*)\\&\leq\left(1-(10r)^{-1}\right)^{T_4}\left(g^0(\boldsymbol{A}^0)-f(\boldsymbol{A}^*)\right)\\&\leq\varepsilon\,,\end{aligned}$$

so $T_3\geq 10r\log\frac{g^0(\boldsymbol{A}^0)-f(\boldsymbol{A}^*)}{\varepsilon}$.

Overall, we have that the total number of iterations is

$$T=\sum T_i\leq 36r(\kappa+1)\log\frac{g^0(\boldsymbol{A}^0)-f(\boldsymbol{A}^*)}{\varepsilon}\,.$$

The only thing left is to ensure that the conditions

$$\begin{aligned}\operatorname{Tr}\left[\boldsymbol{I}-\boldsymbol{W}^t\right]&\leq r'/2\\\operatorname{Tr}\left[\boldsymbol{I}-\boldsymbol{Y}^t\right]&\leq r'/2\end{aligned}$$

of Lemma 9.5.3 are satisfied for all $t$. By the guarantees of Lemma 9.5.3, if one of items 2, 3 holds, then

$$\operatorname{Tr}\left[\boldsymbol{I}-\boldsymbol{W}^{t+1}\right]\leq\operatorname{Tr}\left[\boldsymbol{I}-\boldsymbol{W}^{t}\right]+1\,,$$

and if item 1 holds, then

$$\operatorname{Tr}\left[\boldsymbol{I} - \boldsymbol{W}^{t+1}\right] = \operatorname{Tr}\left[\boldsymbol{I} - \boldsymbol{W}^{t}\right] .$$

As $\operatorname{Tr}\left[\boldsymbol{I} - \boldsymbol{W}^{0}\right] = 0$, we have

$$\begin{aligned}
&\operatorname{Tr}\left[\boldsymbol{I} - \boldsymbol{W}^{T}\right] \\
&\leq T_2 + T_3 \\
&\leq 20\kappa r + 10 r \log \frac{g^0(\boldsymbol{A}^0) - f(\boldsymbol{A}^*)}{\varepsilon} \\
&\leq r'/2 \,,
\end{aligned}$$

where the last inequality holds as long as

$$r' \geq 20 r \left(2\kappa + \log \frac{g^0(\boldsymbol{A}^0) - f(\boldsymbol{A}^*)}{\varepsilon}\right) .$$

□

**Lemma 9.5.3** (Low rank minimization step analysis)**.** *Let $f : \mathbb{R}^{m\times n} \to \mathbb{R}$ be a $\beta$-smooth and $\alpha$-strongly convex function with condition number $\kappa = \beta/\alpha$, and $\boldsymbol{W} \in \mathbb{R}^{m\times m}, \boldsymbol{Y} \in \mathbb{R}^{n\times n}$ be symmetric positive semi-definite weight matrices with spectral norm bounded by 1 and such that $\operatorname{Tr}[\boldsymbol{I} - \boldsymbol{W}] \leq r'/2$ and $\operatorname{Tr}[\boldsymbol{I} - \boldsymbol{Y}] \leq r'/2$ for fixed parameter $r' \geq 256r$. We define the regularized function*

$$g(\boldsymbol{A}) := f(\boldsymbol{A}) + \underbrace{(\beta/4)\left(\langle \boldsymbol{W}, \boldsymbol{A}\boldsymbol{A}^\top\rangle + \langle \boldsymbol{Y}, \boldsymbol{A}^\top\boldsymbol{A}\rangle\right)}_{\Phi(\boldsymbol{A})} .$$

*Now, consider a rank-$r'$ matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ with singular value decomposition*

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{V}^\top = \sum_{j\in S} \lambda_j \boldsymbol{u}_j \boldsymbol{v}_j^\top$$

*and with the property that*

$$\boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{U})} \cdot \nabla g(\boldsymbol{A}) \cdot \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V})} = \boldsymbol{O} .$$

*For any rank-$r$ solution $\boldsymbol{A}^*$ where $r' \geq 256r$, there is a procedure that updates $\boldsymbol{A}, \boldsymbol{W}, \boldsymbol{Y}$, and for which exactly one of the following scenarios holds:*

1. *$\boldsymbol{A}$ is updated to a rank-$r'$ matrix $\boldsymbol{A}'$, and $\boldsymbol{W}, \boldsymbol{Y}$ are not updated. We have sufficient progress in the regularized function:*

$$g(\boldsymbol{A}') \leq g(\boldsymbol{A}) - (16\kappa r)^{-1} \left(g(\boldsymbol{A}) - f(\boldsymbol{A}^*)\right) .$$

2. *Exactly one of $\boldsymbol{W}$ or $\boldsymbol{Y}$ is updated (wlog $\boldsymbol{W}$) to a symmetric PSD $\boldsymbol{W}' \preceq \boldsymbol{W}$, and $\boldsymbol{A}$ is not updated. We have*

$$\operatorname{Tr}\left[\boldsymbol{I} - \boldsymbol{W}'\right] \leq \operatorname{Tr}[\boldsymbol{I} - \boldsymbol{W}] + 1$$

*and*

$$\begin{aligned}&\operatorname{Tr}\left[\boldsymbol{\Pi}_{\operatorname{im}(\boldsymbol{U}^*)}\boldsymbol{W}'\boldsymbol{\Pi}_{\operatorname{im}(\boldsymbol{U}^*)}\right]\\&\leq \operatorname{Tr}\left[\boldsymbol{\Pi}_{\operatorname{im}(\boldsymbol{U}^*)}\boldsymbol{W}\boldsymbol{\Pi}_{\operatorname{im}(\boldsymbol{U}^*)}\right]-(10\kappa)^{-1}\,.\end{aligned}$$

*Respectively, for* $\boldsymbol{Y}$*:*

$$\begin{aligned}&\operatorname{Tr}\left[\boldsymbol{\Pi}_{\operatorname{im}(\boldsymbol{V}^*)}\boldsymbol{Y}'\boldsymbol{\Pi}_{\operatorname{im}(\boldsymbol{V}^*)}\right]\\&\leq \operatorname{Tr}\left[\boldsymbol{\Pi}_{\operatorname{im}(\boldsymbol{V}^*)}\boldsymbol{Y}\boldsymbol{\Pi}_{\operatorname{im}(\boldsymbol{V}^*)}\right]-(10\kappa)^{-1}\,.\end{aligned}$$

3. *Exactly one of* $\boldsymbol{W}$ *or* $\boldsymbol{Y}$ *is updated (wlog* $\boldsymbol{W}$*) to a symmetric PSD* $\boldsymbol{W}' \preceq \boldsymbol{W}$*, and* $\boldsymbol{A}$ *is not updated. We have sufficient progress in the regularized function, where* $g'$ *is the regularized function with the new weights:*

$$g'(\boldsymbol{A}) \leq g(\boldsymbol{A}) - (10r)^{-1}\left(g(\boldsymbol{A}) - f(\boldsymbol{A}^*)\right)\,.$$

*Additionally,*

$$\operatorname{Tr}\left[\boldsymbol{I} - \boldsymbol{W}'\right] \leq \operatorname{Tr}[\boldsymbol{I} - \boldsymbol{W}] + 1$$

*Proof.* We attempt to make the update $\boldsymbol{A} \to \boldsymbol{A}'$ as defined in Lemma 9.5.4. If it makes enough progress, i.e.

$$g(\boldsymbol{A}') \leq g(\boldsymbol{A}) - (16\kappa r)^{-1}\left(g(\boldsymbol{A}) - f(\boldsymbol{A}^*)\right)\,,$$

we are done. Otherwise, one of the items 2-5 in the statement of Lemma 9.5.4 must hold. Let us take them one by one.

**Item 2:**

$$\langle \boldsymbol{\Pi}_{\operatorname{im}(\boldsymbol{U}^*)}, \boldsymbol{W}\boldsymbol{A}\boldsymbol{A}^\top\boldsymbol{W}\rangle \geq (10\kappa)^{-1}\langle \boldsymbol{W}, \boldsymbol{A}\boldsymbol{A}^\top\rangle\,.$$

We update $\boldsymbol{W}$ as

$$\boldsymbol{W}' = \boldsymbol{W} - c\cdot \boldsymbol{W}\boldsymbol{A}\boldsymbol{A}^\top\boldsymbol{W}\,,$$

where $c = \langle \boldsymbol{W}, \boldsymbol{A}\boldsymbol{A}^\top\rangle^{-1}$. Note that this update preserves symmetry, and

$$\boldsymbol{O} \preceq \boldsymbol{W}' \preceq \boldsymbol{W}\,.$$

This is because

$$c\boldsymbol{W}^{1/2}\boldsymbol{A}\boldsymbol{A}^\top\boldsymbol{W}^{1/2} \preceq c\langle \boldsymbol{W}, \boldsymbol{A}\boldsymbol{A}^\top\rangle\cdot\boldsymbol{I} \preceq \boldsymbol{I}\,,$$

so

$$\boldsymbol{W}' = \boldsymbol{W}^{1/2}\left(\boldsymbol{I} - c\boldsymbol{W}^{1/2}\boldsymbol{A}\boldsymbol{A}^\top\boldsymbol{W}^{1/2}\right)\boldsymbol{W}^{1/2} \succeq \boldsymbol{O}$$

and

$$\boldsymbol{W}' = \boldsymbol{W} - c\boldsymbol{W}\boldsymbol{A}\boldsymbol{A}^\top\boldsymbol{W} \preceq \boldsymbol{W}\,.$$

Now, note that

$$\begin{aligned}\operatorname{Tr}\left[\boldsymbol{I}-\boldsymbol{W}'\right] &= \operatorname{Tr}\left[\boldsymbol{I}-\boldsymbol{W}\right] + c\langle \boldsymbol{W}^2, \boldsymbol{A}\boldsymbol{A}^\top\rangle \\ &\leq \operatorname{Tr}\left[\boldsymbol{I}-\boldsymbol{W}\right] + c\langle \boldsymbol{W}, \boldsymbol{A}\boldsymbol{A}^\top\rangle \\ &= \operatorname{Tr}\left[\boldsymbol{I}-\boldsymbol{W}\right] + 1\,,\end{aligned}$$

where we used the fact that $\boldsymbol{W}^2 \preceq \boldsymbol{W}$, and (letting $\boldsymbol{\Pi}^* = \boldsymbol{\Pi}_{\operatorname{im}(\boldsymbol{U}^*)}$ for convenience),

$$\begin{aligned}\operatorname{Tr}\left[\boldsymbol{\Pi}^*\,\boldsymbol{W}'\boldsymbol{\Pi}^*\right] &= \operatorname{Tr}\left[\boldsymbol{\Pi}^*\,\boldsymbol{W}\,\boldsymbol{\Pi}^*\right] - c\langle \boldsymbol{\Pi}^*, \boldsymbol{W}\,\boldsymbol{A}\boldsymbol{A}^\top\,\boldsymbol{W}\rangle \\ &\leq \operatorname{Tr}\left[\boldsymbol{\Pi}^*\,\boldsymbol{W}\,\boldsymbol{\Pi}^*\right] - c/(10\kappa)\langle \boldsymbol{W}, \boldsymbol{A}\boldsymbol{A}^\top\rangle \\ &= \operatorname{Tr}\left[\boldsymbol{\Pi}^*\,\boldsymbol{W}\,\boldsymbol{\Pi}^*\right] - (10\kappa)^{-1}\,,\end{aligned} \tag{9.78}$$

**Item 3:**

$$\langle \boldsymbol{\Pi}_{\operatorname{im}(\boldsymbol{V}^*)}, \boldsymbol{Y}\,\boldsymbol{A}^\top\,\boldsymbol{A}\,\boldsymbol{Y}\rangle \geq (10\kappa)^{-1}\,\langle \boldsymbol{Y}, \boldsymbol{A}^\top\,\boldsymbol{A}\rangle\,.$$

This is entirely analogous to the previous case.

**Item 4:**

$$(\beta/4)\operatorname{Tr}\left[H_r\left(\boldsymbol{A}^\top\,\boldsymbol{W}\,\boldsymbol{A}\right)\right] \geq 10^{-1}\,(g(\boldsymbol{A}) - f(\boldsymbol{A}^*))\;. \tag{9.79}$$

After considering the eigendecomposition

$$\boldsymbol{W}^{1/2}\boldsymbol{A}\boldsymbol{A}^\top\,\boldsymbol{W}^{1/2} = \sum_{i\in[r']} \bar{\lambda}_i \bar{\boldsymbol{v}}_i \bar{\boldsymbol{v}}_i^\top$$

with $\bar{\lambda}_1 \geq \bar{\lambda}_2 \geq \cdots \geq \bar{\lambda}_{r'} \geq 0$, (9.79) can be re-phrased as

$$(\beta/4)\sum_{i\in[r]} \bar{\lambda}_i > (1/10)\,(g(\boldsymbol{A}) - f(\boldsymbol{A}^*))\;.$$

We update $\boldsymbol{W}$ as

$$\boldsymbol{W}' = \boldsymbol{W}^{1/2}\left(\boldsymbol{I} - r^{-1}\sum_{i\in[r]} \bar{\boldsymbol{v}}_i \bar{\boldsymbol{v}}_i^\top\right)\boldsymbol{W}^{1/2}$$

and let $g'$ be the new regularized objective. First of all, note that this operation preserves symmetry, and that $\boldsymbol{O} \preceq \boldsymbol{W}' \preceq \boldsymbol{I}$, since $\sum\limits_{i\in[r]} \bar{\boldsymbol{v}}_i \bar{\boldsymbol{v}}_i^\top \preceq \boldsymbol{I}$. Additionally,

$$\begin{aligned}\operatorname{Tr}\left[\boldsymbol{I}-\boldsymbol{W}'\right] &= \operatorname{Tr}\left[\boldsymbol{I}-\boldsymbol{W}\right] + r^{-1}\sum_{i\in[r]} \bar{\boldsymbol{v}}_i^\top\,\boldsymbol{W}\,\bar{\boldsymbol{v}}_i \\ &\leq \operatorname{Tr}\left[\boldsymbol{W}\right] + 1\end{aligned}$$

and

$$\begin{aligned}
&g'(\boldsymbol{A}) - g(\boldsymbol{A}) \\
&= (\beta/4)\langle \boldsymbol{W}', \boldsymbol{A}\boldsymbol{A}^\top\rangle - (\beta/4)\langle \boldsymbol{W}, \boldsymbol{A}\boldsymbol{A}^\top\rangle \\
&= -(\beta/(4r))\left\langle \boldsymbol{W}^{1/2}\left(\sum_{i\in[r]} \bar{\boldsymbol{v}}_i\bar{\boldsymbol{v}}_i^\top\right)\boldsymbol{W}^{1/2}, \boldsymbol{A}\boldsymbol{A}^\top\right\rangle \\
&= -(\beta/(4r))\sum_{i\in[r]} \bar{\lambda}_i \\
&\leq -(10r)^{-1}(g(\boldsymbol{A}) - f(\boldsymbol{A}^*))\,,
\end{aligned}$$

**Item 5:**

$$(\beta/4)\operatorname{Tr}\left[H_r\left(\boldsymbol{A}\boldsymbol{Y}\boldsymbol{A}^\top\right)\right] \geq 10^{-1}\left(g(\boldsymbol{A}) - f(\boldsymbol{A}^*)\right)\,.$$

This is entirely analogous to the previous case. □

**Lemma 9.5.4.** *Let $f : \mathbb{R}^{m\times n} \to \mathbb{R}$ be a $\beta$-smooth and $\alpha$-strongly convex function with condition number $\kappa = \beta/\alpha$, and $\boldsymbol{W} \in \mathbb{R}^{m\times m}$, $\boldsymbol{Y} \in \mathbb{R}^{n\times n}$ be symmetric positive semi-definite weight matrices with spectral norm bounded by 1 and such that $\operatorname{Tr}[\boldsymbol{I} - \boldsymbol{W}], \operatorname{Tr}[\boldsymbol{I} - \boldsymbol{Y}] \leq r'/2$ for some parameter $r' \geq 0$. We define the regularized function*

$$g(\boldsymbol{A}) := f(\boldsymbol{A}) + \underbrace{(\beta/4)\left(\langle \boldsymbol{W}, \boldsymbol{A}\boldsymbol{A}^\top\rangle + \langle \boldsymbol{Y}, \boldsymbol{A}^\top\boldsymbol{A}\rangle\right)}_{\Phi(\boldsymbol{A})}\,.$$

*Now, consider a rank-$r'$ matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ with singular value decomposition*

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{V}^\top = \sum_{j\in S} \lambda_j \boldsymbol{u}_j\boldsymbol{v}_j^\top$$

*and with the property that*

$$\boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{U})} \cdot \nabla g(\boldsymbol{A}) \cdot \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V})} = \boldsymbol{O}\,.$$

*We define an updated solution*

$$\boldsymbol{A}' = \boldsymbol{A} - \eta \cdot H_1(\nabla g(\boldsymbol{A})) - \lambda_j \boldsymbol{u}_j\boldsymbol{v}_j^\top\,,$$

*where $\eta = (2\beta)^{-1}$, $H_1(\cdot)$ returns the top singular component, and $j \in S$ is picked to minimize $\lambda_j$.*

*Then, for any rank-$r$ solution $\boldsymbol{A}^*$, where $r' \geq 256r$, and its singular value decomposition $\boldsymbol{A}^* = \boldsymbol{U}^*\boldsymbol{\Lambda}^*\boldsymbol{V}^{*\top}$, at least one of the following conditions holds:*

1. *We have sufficient progress in the regularized function:*

$$g(\boldsymbol{A}') \leq g(\boldsymbol{A}) - (16\kappa r)^{-1}\left(g(\boldsymbol{A}) - f(\boldsymbol{A}^*)\right)\,.$$

2. *$\boldsymbol{W}\boldsymbol{A}\boldsymbol{A}^\top\boldsymbol{W}$ is significantly correlated to $\boldsymbol{U}^*$:*

$$\langle \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{U}^*)}, \boldsymbol{W}\boldsymbol{A}\boldsymbol{A}^\top\boldsymbol{W}\rangle \geq (10\kappa)^{-1}\langle \boldsymbol{W}, \boldsymbol{A}\boldsymbol{A}^\top\rangle\,.$$

3. $\boldsymbol{Y}\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{Y}$ *is significantly correlated to* $\boldsymbol{V}^*$*:*

$$\langle \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V}^*)}, \boldsymbol{Y}\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{Y}\rangle \geq (10\kappa)^{-1}\langle \boldsymbol{Y}, \boldsymbol{A}^\top\boldsymbol{A}\rangle .$$

4. *The spectrum of* $\boldsymbol{A}^\top\boldsymbol{W}\boldsymbol{A}$ *is highly concentrated and responsible for a constant fraction of the error:*

$$(\beta/4)\,\mathrm{Tr}\left[H_r\left(\boldsymbol{A}^\top\boldsymbol{W}\boldsymbol{A}\right)\right] \geq 10^{-1}\left(g(\boldsymbol{A}) - f(\boldsymbol{A}^*)\right) .$$

*and*

5. *The spectrum of* $\boldsymbol{A}\boldsymbol{Y}\boldsymbol{A}^\top$ *is highly concentrated and responsible for a constant fraction of the error:*

$$(\beta/4)\,\mathrm{Tr}\left[H_r\left(\boldsymbol{A}\boldsymbol{Y}\boldsymbol{A}^\top\right)\right] \geq 10^{-1}\left(g(\boldsymbol{A}) - f(\boldsymbol{A}^*)\right) .$$

*Proof.* Note that $g$ is a $2\beta$-smooth function. This follows because

$$\nabla g(\boldsymbol{A}) = \nabla f(\boldsymbol{A}) + (\beta/2)\left(\boldsymbol{W}\boldsymbol{A} + \boldsymbol{A}\boldsymbol{Y}\right) ,$$

and so for any two matrices $\boldsymbol{A}, \boldsymbol{A}'$,

$$\begin{aligned}
&\left\|\nabla g(\boldsymbol{A}') - \nabla g(\boldsymbol{A})\right\|_F \\
&\leq \left\|\nabla f(\boldsymbol{A}') - \nabla f(\boldsymbol{A})\right\|_F + (\beta/2)\left\|\boldsymbol{W}(\boldsymbol{A}' - \boldsymbol{A})\right\|_F + (\beta/2)\left\|(\boldsymbol{A}' - \boldsymbol{A})\boldsymbol{Y}\right\|_F \\
&\leq 2\beta\left\|\boldsymbol{A}' - \boldsymbol{A}\right\|_F ,
\end{aligned}$$

which is known to imply $2\beta$-smoothness of $g$. Here we used the triangle inequality and the fact that $\boldsymbol{W}, \boldsymbol{Y} \preceq \boldsymbol{I}$. Therefore, we have

$$\begin{aligned}
&g(\boldsymbol{A}') - g(\boldsymbol{A}) \\
&\leq \langle \nabla g(\boldsymbol{A}), \boldsymbol{A}' - \boldsymbol{A}\rangle + \left\|\nabla g(\boldsymbol{A}') - \nabla g(\boldsymbol{A})\right\|_F \left\|\boldsymbol{A}' - \boldsymbol{A}\right\|_F \\
&\leq \langle \nabla g(\boldsymbol{A}), \boldsymbol{A}' - \boldsymbol{A}\rangle + \beta\left\|\boldsymbol{A}' - \boldsymbol{A}\right\|_F^2 \\
&\leq -\eta\left\|\nabla g(\boldsymbol{A})\right\|_2^2 + 2\beta\eta^2\left\|\nabla g(\boldsymbol{A})\right\|_2^2 + 2\beta\lambda_j^2 \\
&= -(8\beta)^{-1}\left\|\nabla g(\boldsymbol{A})\right\|_2^2 + 2\beta\lambda_j^2 ,
\end{aligned} \tag{9.80}$$

where in the second inequality we used the facts that

$$\begin{aligned}
&\langle \nabla g(\boldsymbol{A}), -\lambda_j \boldsymbol{u}_j \boldsymbol{v}_j^\top\rangle \\
&= \langle \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{U})}\nabla g(\boldsymbol{A})\boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V})}, -\lambda_j \boldsymbol{u}_j \boldsymbol{v}_j^\top\rangle \\
&= 0
\end{aligned}$$

and that, for any two matrices $\boldsymbol{B}, \boldsymbol{C}$,

$$\left\|\boldsymbol{B} + \boldsymbol{C}\right\|_F^2 \leq 2\left\|\boldsymbol{B}\right\| + 2\left\|\boldsymbol{C}\right\|_F^2 .$$

The last equality follows by our choice of $\eta$. In order to lower bound $\left\|\nabla g(\boldsymbol{A})\right\|_2^2$, we use the strong

convexity of $f$ as follows:

$$\begin{aligned}
&f(\boldsymbol{A}^*) - f(\boldsymbol{A}) \\
&\geq \langle \nabla f(\boldsymbol{A}), \boldsymbol{A}^* - \boldsymbol{A} \rangle + (\alpha/2) \left\| \boldsymbol{A}^* - \boldsymbol{A} \right\|_F^2 \\
&= \langle \nabla g(\boldsymbol{A}), \boldsymbol{A}^* - \boldsymbol{A} \rangle - \langle \nabla \Phi(\boldsymbol{A}), \boldsymbol{A}^* - \boldsymbol{A} \rangle + (\alpha/2) \left\| \boldsymbol{A}^* - \boldsymbol{A} \right\|_F^2 \\
&= \underbrace{\langle \nabla g(\boldsymbol{A}), \boldsymbol{A}^* - \boldsymbol{A} \rangle + (\alpha/4) \left\| \boldsymbol{A}^* - \boldsymbol{A} \right\|_F^2}_{P} - \langle \nabla \Phi(\boldsymbol{A}), \boldsymbol{A}^* - \boldsymbol{A} \rangle + (\alpha/4) \left\| \boldsymbol{A}^* - \boldsymbol{A} \right\|_F^2 .
\end{aligned} \tag{9.81}$$

**Bounding $P$.** We let $\boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{U})}$, $\boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V})}$ be the orthogonal projections onto the images of $\boldsymbol{U}$ and $\boldsymbol{V}$ respectively, so we can write

$$\begin{aligned}
&\boldsymbol{A}^* - \boldsymbol{A} \\
&= \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{U})} (\boldsymbol{A}^* - \boldsymbol{A}) \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V})} + \left(\boldsymbol{I} - \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{U})}\right) (\boldsymbol{A}^* - \boldsymbol{A}) \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V})} + (\boldsymbol{A}^* - \boldsymbol{A}) \left(\boldsymbol{I} - \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V})}\right) \\
&= \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{U})} (\boldsymbol{A}^* - \boldsymbol{A}) \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V})} + \left(\boldsymbol{I} - \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{U})}\right) \boldsymbol{A}^* \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V})} + \boldsymbol{A}^* \left(\boldsymbol{I} - \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V})}\right) .
\end{aligned}$$

Now, note that

$$\begin{aligned}
&\langle \nabla g(\boldsymbol{A}), \boldsymbol{A}^* - \boldsymbol{A} \rangle \\
&= \langle \nabla g(\boldsymbol{A}), \left(\boldsymbol{I} - \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{U})}\right) \boldsymbol{A}^* \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V})} \rangle + \langle \nabla g(\boldsymbol{A}), \boldsymbol{A}^* \left(\boldsymbol{I} - \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V})}\right) \rangle ,
\end{aligned}$$

where we used the fact that

$$\begin{aligned}
&\langle \nabla g(\boldsymbol{A}), \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{U})} (\boldsymbol{A}^* - \boldsymbol{A}) \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V})} \rangle \\
&= \langle \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{U})} \nabla g(\boldsymbol{A}) \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V})}, \boldsymbol{A}^* - \boldsymbol{A} \rangle \\
&= 0 ,
\end{aligned}$$

and

$$\begin{aligned}
&(\alpha/4) \left\| \boldsymbol{A}^* - \boldsymbol{A} \right\|_F^2 \\
&\geq (\alpha/4) \left\| \left(\boldsymbol{I} - \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{U})}\right) \boldsymbol{A}^* \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V})} \right\|_F^2 + (\alpha/4) \left\| \boldsymbol{A}^* \left(\boldsymbol{I} - \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V})}\right) \right\|_F^2 .
\end{aligned}$$

Additionally, note that for any rank-$r$ matrix $\boldsymbol{B}$, we have

$$\begin{aligned}
&\langle \nabla g(\boldsymbol{A}), \boldsymbol{B} \rangle + (\alpha/4) \left\| \boldsymbol{B} \right\|_F^2 \\
&\geq -\alpha^{-1} \left\| H_r \left( \nabla g(\boldsymbol{A}) \right) \right\|_F^2 \\
&\geq -\alpha^{-1} r \left\| \nabla g(\boldsymbol{A}) \right\|_2^2 ,
\end{aligned}$$

a proof of which can be found e.g. in Lemma A.6 of [13]. Applying this inequality with

$$\boldsymbol{B} = \left(\boldsymbol{I} - \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{U})}\right) \boldsymbol{A}^* \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V})}$$

and

$$\boldsymbol{B} = \boldsymbol{A}^* \left(\boldsymbol{I} - \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V})}\right)$$

and summing them up, we obtain

$$\begin{aligned} P &= \langle \nabla g(\boldsymbol{A}), \boldsymbol{A}^* - \boldsymbol{A} \rangle + (\alpha/4) \left\| \boldsymbol{A}^* - \boldsymbol{A} \right\|_F^2 \\ &\geq -2\alpha^{-1} r \left\| \nabla g(\boldsymbol{A}) \right\|_2^2 . \end{aligned}$$

Plugging this into (9.81) and re-arranging, we get

$$\begin{aligned} &\left\| \nabla g(\boldsymbol{A}) \right\|_2^2 \\ &\geq \alpha/(2r) \Big( f(\boldsymbol{A}) - f(\boldsymbol{A}^*) - \langle \nabla \Phi(\boldsymbol{A}), \boldsymbol{A}^* - \boldsymbol{A} \rangle + (\alpha/4) \left\| \boldsymbol{A}^* - \boldsymbol{A} \right\|_F^2 \Big) \\ &= \alpha/(2r) \Big( g(\boldsymbol{A}) - f(\boldsymbol{A}^*) - \Phi(\boldsymbol{A}) \underbrace{- \langle \nabla \Phi(\boldsymbol{A}), \boldsymbol{A}^* - \boldsymbol{A} \rangle + (\alpha/4) \left\| \boldsymbol{A}^* - \boldsymbol{A} \right\|_F^2}_{Q} \Big) . \end{aligned} \tag{9.82}$$

**Bounding $Q$.** We know that

$$- \langle \nabla \Phi(\boldsymbol{A}), \boldsymbol{A}^* - \boldsymbol{A} \rangle = -(\beta/2) \langle \boldsymbol{W}\boldsymbol{A} + \boldsymbol{A}\boldsymbol{Y}, \boldsymbol{A}^* - \boldsymbol{A} \rangle .$$

If we let

$$\boldsymbol{A}^* = \boldsymbol{U}^* \boldsymbol{\Lambda}^* \boldsymbol{V}^{*\top}$$

be the SVD of $\boldsymbol{A}^*$ and $\boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{U}^*)}$, $\boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V}^*)}$ be the orthogonal projections onto the images of $\boldsymbol{U}^*$ and $\boldsymbol{V}^*$ respectively, then we have

$$\begin{aligned} &- (\beta/2) \langle \boldsymbol{W}\boldsymbol{A}, \boldsymbol{A}^* - \boldsymbol{A} \rangle \\ &= -(\beta/2) \langle \boldsymbol{W}\boldsymbol{A}, \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{U}^*)} (\boldsymbol{A}^* - \boldsymbol{A}) \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V}^*)} \rangle + (\beta/2) \langle \boldsymbol{W}, \boldsymbol{A}\boldsymbol{A}^\top \rangle - (\beta/2) \langle \boldsymbol{W}\boldsymbol{A}, \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{U}^*)} \boldsymbol{A} \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V}^*)} \rangle . \end{aligned}$$

Looking at the first term of this, we have

$$\begin{aligned} &- (\beta/2) \langle \boldsymbol{W}\boldsymbol{A}, \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{U}^*)} (\boldsymbol{A}^* - \boldsymbol{A}) \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V}^*)} \rangle + (\alpha/8) \left\| \boldsymbol{A}^* - \boldsymbol{A} \right\|_F^2 \\ &= -(\beta/2) \langle \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{U}^*)} \boldsymbol{W}\boldsymbol{A} \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V}^*)}, \boldsymbol{A}^* - \boldsymbol{A} \rangle + (\alpha/8) \left\| \boldsymbol{A}^* - \boldsymbol{A} \right\|_F^2 \\ &\geq -\beta^2/(2\alpha) \left\| \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{U}^*)} \boldsymbol{W}\boldsymbol{A} \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V}^*)} \right\|_F^2 . \end{aligned}$$

Similarly for the terms containing $\boldsymbol{Y}$, we get

$$\begin{aligned} &- (\beta/2) \langle \boldsymbol{A}\boldsymbol{Y}, \boldsymbol{A}^* - \boldsymbol{A} \rangle \\ &= -(\beta/2) \langle \boldsymbol{A}\boldsymbol{Y}, \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{U}^*)} (\boldsymbol{A}^* - \boldsymbol{A}) \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V}^*)} \rangle + (\beta/2) \langle \boldsymbol{Y}, \boldsymbol{A}^\top \boldsymbol{A} \rangle - (\beta/2) \langle \boldsymbol{A}\boldsymbol{Y}, \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{U}^*)} \boldsymbol{A} \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V}^*)} \rangle . \end{aligned}$$

and

$$\begin{aligned} &- (\beta/2) \langle \boldsymbol{A}\boldsymbol{Y}, \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{U}^*)} (\boldsymbol{A}^* - \boldsymbol{A}) \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V}^*)} \rangle + (\alpha/8) \left\| \boldsymbol{A}^* - \boldsymbol{A} \right\|_F^2 \\ &\geq -\beta^2/(2\alpha) \left\| \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{U}^*)} \boldsymbol{A}\boldsymbol{Y} \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V}^*)} \right\|_F^2 . \end{aligned}$$

In summary, we have

$$\begin{aligned} Q &= -\langle \nabla\Phi(\boldsymbol{A}), \boldsymbol{A}^* - \boldsymbol{A}\rangle + (\alpha/4)\left\|\boldsymbol{A}^* - \boldsymbol{A}\right\|_F^2 \\ &\geq -\beta^2/(2\alpha)\left\|\boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{U}^*)}\boldsymbol{W}\boldsymbol{A}\boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V}^*)}\right\|_F^2 - \beta^2/(2\alpha)\left\|\boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{U}^*)}\boldsymbol{A}\boldsymbol{Y}\boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V}^*)}\right\|_F^2 \\ &+ (\beta/2)\langle \boldsymbol{W}, \boldsymbol{A}\boldsymbol{A}^\top\rangle + (\beta/2)\langle \boldsymbol{Y}, \boldsymbol{A}^\top\boldsymbol{A}\rangle \\ &- (\beta/2)\langle \boldsymbol{W}\boldsymbol{A}, \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{U}^*)}\boldsymbol{A}\boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V}^*)}\rangle - (\beta/2)\langle \boldsymbol{A}\boldsymbol{Y}, \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{U}^*)}\boldsymbol{A}\boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V}^*)}\rangle\,. \end{aligned} \tag{9.83}$$

Now, let us assume that all items 2-5 from the lemma statement are false. For the first term of (9.83), we have

$$\begin{aligned} &-\beta^2/(2\alpha)\left\|\boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{U}^*)}\boldsymbol{W}\boldsymbol{A}\boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V}^*)}\right\|_F^2 \\ &\geq -\beta^2/(2\alpha)\left\|\boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{U}^*)}\boldsymbol{W}\boldsymbol{A}\right\|_F^2 \\ &= -\beta^2/(2\alpha)\langle \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{U}^*)}, \boldsymbol{W}\boldsymbol{A}\boldsymbol{A}^\top\boldsymbol{W}\rangle \\ &\geq -(\beta/20)\langle \boldsymbol{W}, \boldsymbol{A}\boldsymbol{A}^\top\rangle\,, \end{aligned}$$

where we used item 2 from the lemma statement, and similarly for the second term of (9.83),

$$\begin{aligned} &-\beta^2/(2\alpha)\left\|\boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{U}^*)}\boldsymbol{A}\boldsymbol{Y}\boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V}^*)}\right\|_F^2 \\ &\geq -(\beta/20)\langle \boldsymbol{Y}, \boldsymbol{A}^\top\boldsymbol{A}\rangle\,. \end{aligned}$$

Now we look at the second to last term of (9.83), i.e.

$$\begin{aligned} &-(\beta/2)\langle \boldsymbol{W}\boldsymbol{A}, \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{U}^*)}\boldsymbol{A}\boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V}^*)}\rangle \\ &= -(\beta/2)\langle \boldsymbol{W}\boldsymbol{A}\boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V}^*)}\boldsymbol{A}^\top, \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{U}^*)}\rangle\,. \end{aligned}$$

Now, we use the matrix Holder inequality

$$\begin{aligned} &-(\beta/2)\langle \boldsymbol{W}\boldsymbol{A}\boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V}^*)}\boldsymbol{A}^\top, \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{U}^*)}\rangle \\ &\geq -(\beta/2)\left\|\boldsymbol{W}\boldsymbol{A}\boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V}^*)}\boldsymbol{A}^\top\right\|_*\left\|\boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{U}^*)}\right\|_2 \\ &\geq -(\beta/2)\left\|\boldsymbol{W}\boldsymbol{A}\boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V}^*)}\boldsymbol{A}^\top\right\|_*\,, \end{aligned}$$

which can be proved by applying von Neumann's trace inequality and then the classical Holder inequality. Now, note that the matrix $\boldsymbol{W}\boldsymbol{A}\boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V}^*)}\boldsymbol{A}^\top$ is similar to $\boldsymbol{W}^{1/2}\boldsymbol{A}\boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V}^*)}\boldsymbol{A}^\top\boldsymbol{W}^{1/2}$, and so they have the same eigenvalues. Furthermore, the latter is a symmetric PSD matrix, and so the former has real positive eigenvalues as well. This means that its singular values are the same as its eigenvalues, and as a result the nuclear norm is equal to the trace, i.e.

$$\begin{aligned} &-(\beta/2)\left\|\boldsymbol{W}\boldsymbol{A}\boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V}^*)}\boldsymbol{A}^\top\right\|_* \\ &= -(\beta/2)\,\mathrm{Tr}\left(\boldsymbol{W}\boldsymbol{A}\boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V}^*)}\boldsymbol{A}^\top\right) \\ &= -(\beta/2)\langle \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V}^*)}, \boldsymbol{A}^\top\boldsymbol{W}\boldsymbol{A}\rangle \\ &\geq -(\beta/2)\,\mathrm{Tr}\left[H_r\left(\boldsymbol{A}^\top\boldsymbol{W}\boldsymbol{A}\right)\right] \\ &\geq -(1/5)\left(g(\boldsymbol{A}) - f(\boldsymbol{A}^*)\right)\,. \end{aligned}$$

where we also used Lemma 9.5.2 and item 4 from the lemma statement. So we derived that

$$\begin{aligned}&-(\beta/2)\langle \boldsymbol{W}\boldsymbol{A}, \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{U}^*)}\boldsymbol{A}\boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V}^*)}\rangle \\ &\geq -(1/5)\left(g(\boldsymbol{A}) - f(\boldsymbol{A}^*)\right),\end{aligned}$$

and similarly for the last term of (9.83),

$$\begin{aligned}&-(\beta/2)\langle \boldsymbol{A}\boldsymbol{Y}, \boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{U}^*)}\boldsymbol{A}\boldsymbol{\Pi}_{\mathrm{im}(\boldsymbol{V}^*)}\rangle \\ &\geq -(1/5)\left(g(\boldsymbol{A}) - f(\boldsymbol{A}^*)\right).\end{aligned}$$

Plugging the four inequalities that we derived back into (9.83), we get

$$\begin{aligned}Q &\geq (\beta/2 - \beta/20)\langle \boldsymbol{W}, \boldsymbol{A}\boldsymbol{A}^\top\rangle + (\beta/2 - \beta/20)\langle \boldsymbol{Y}, \boldsymbol{A}^\top\boldsymbol{A}\rangle - (2/5)\left(g(\boldsymbol{A}) - f(\boldsymbol{A}^*)\right) \\ &= (9/5)\Phi(\boldsymbol{A}) - (2/5)\left(g(\boldsymbol{A}) - f(\boldsymbol{A}^*)\right) \\ &> (3/2)\Phi(\boldsymbol{A}) - (2/5)\left(g(\boldsymbol{A}) - f(\boldsymbol{A}^*)\right).\end{aligned}$$

Finally, combining this with the smoothness inequality (9.80) and the lower bound on $\|\nabla g(\boldsymbol{A})\|_2^2$ (9.82), we derive

$$\begin{aligned}&g(\boldsymbol{A}') - g(\boldsymbol{A}) \\ &\leq -(16\kappa r)^{-1}\Big(g(\boldsymbol{A}) - f(\boldsymbol{A}^*) + (1/2)\Phi(\boldsymbol{A})\Big) + 2\beta\lambda_j^2 \\ &= -(16\kappa r)^{-1}\Big(g(\boldsymbol{A}) - f(\boldsymbol{A}^*)\Big) - (32\kappa r)^{-1}\Phi(\boldsymbol{A}) + 2\beta\lambda_j^2.\end{aligned}$$

What remains is the bound the sum of the last two terms. We remind the reader that $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{V}^\top$. Now, letting $\boldsymbol{z}$ equal to the vectorized diagonal of $\boldsymbol{U}^\top\boldsymbol{W}\boldsymbol{U}$ and $\boldsymbol{\lambda}$ to the vectorized diagonal of $\boldsymbol{\Lambda}$, note that

$$\|\boldsymbol{\lambda}\|_{\boldsymbol{z}}^2 = \langle \boldsymbol{\Lambda}^2, \boldsymbol{U}^\top\boldsymbol{W}\boldsymbol{U}\rangle = \langle \boldsymbol{W}, \boldsymbol{A}\boldsymbol{A}^\top\rangle,$$

using which we derive

$$\begin{aligned}\lambda_j^2 = \min_{j\in S}\lambda_j^2 &\leq \frac{\|\boldsymbol{\lambda}\|_{\boldsymbol{z}}^2}{\|\boldsymbol{z}\|_1} \\ &= \frac{\langle \boldsymbol{W}, \boldsymbol{A}\boldsymbol{A}^\top\rangle}{\mathrm{Tr}[\boldsymbol{U}^\top\boldsymbol{W}\boldsymbol{U}]} \\ &= \frac{\langle \boldsymbol{W}, \boldsymbol{A}\boldsymbol{A}^\top\rangle}{\mathrm{Tr}[\boldsymbol{U}^\top\boldsymbol{U}] - \mathrm{Tr}[\boldsymbol{U}^\top(\boldsymbol{I} - \boldsymbol{W})\boldsymbol{U}]} \\ &\leq \frac{\langle \boldsymbol{W}, \boldsymbol{A}\boldsymbol{A}^\top\rangle}{r' - \mathrm{Tr}[\boldsymbol{I} - \boldsymbol{W}]} \\ &\leq \frac{\langle \boldsymbol{W}, \boldsymbol{A}\boldsymbol{A}^\top\rangle}{r'/2} \\ &\leq \frac{\langle \boldsymbol{W}, \boldsymbol{A}\boldsymbol{A}^\top\rangle}{128r\kappa},\end{aligned}$$

where we used the fact that

$$\begin{aligned}
&\mathrm{Tr}[\boldsymbol{U}^\top(\boldsymbol{I}-\boldsymbol{W})\,\boldsymbol{U}] \\
&= \mathrm{Tr}\left[(\boldsymbol{I}-\boldsymbol{W})^{1/2}\,\boldsymbol{U}\,\boldsymbol{U}^\top(\boldsymbol{I}-\boldsymbol{W})^{1/2}\right] \\
&\le \mathrm{Tr}[\boldsymbol{I}-\boldsymbol{W}]\,,
\end{aligned}$$

because the columns of $\boldsymbol{U}$ are orthonormal. We also used the property that $\mathrm{Tr}[\boldsymbol{I}-\boldsymbol{W}] \le r'/2$ and the fact that $r' \ge 256r\kappa$ by the lemma statement.

Similarly, we derive that

$$\lambda_j^2 \le \frac{\langle \boldsymbol{Y}, \boldsymbol{A}^\top\boldsymbol{A}\rangle}{128r\kappa}\,,$$

and, adding these two inequalities, we have

$$2\beta\lambda_j^2 \le (32r\kappa)^{-1}\Phi(\boldsymbol{A})\,,$$

finally concluding that

$$g(\boldsymbol{A}') - g(\boldsymbol{A}) \le -(16\kappa r)^{-1}\Big(g(\boldsymbol{A}) - f(\boldsymbol{A}^*)\Big)\,.$$

□

### 9.5.5 Lower Bounds

**Lemma 9.5.5** (IHT lower bound)**.** *Let $f(\boldsymbol{x}) := (1/2)\,\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2^2$. For any $\kappa, s \ge 1$, $s' \le 0.6s\kappa^2$, there exists a (diagonal) matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and a vector $\boldsymbol{b} \in \mathbb{R}^n$ where $n = s(\kappa^2+\kappa+1)$, $f$ is 1-strongly convex and $\kappa$-smooth, as well as an $s$-sparse solution $\boldsymbol{x}^*$ and an $s'$-sparse solution $\boldsymbol{x}$, such that*

$$f(\boldsymbol{x}) \ge f(\boldsymbol{x}^*) + 0.1s\kappa^2$$

*but*

$$\boldsymbol{x} = H_{s'}\left(\boldsymbol{x} - \beta^{-1}\nabla f(\boldsymbol{x})\right)\,,$$

*i.e. $\boldsymbol{x}$ is a fixpoint for IHT.*

*Proof.* We use the same example as in [14], Section 5.2: $\boldsymbol{A}$ is diagonal with

$$\boldsymbol{A}_{ii} = \begin{cases} 1 & \text{if } i \in I_1 \\ \sqrt{\kappa} & \text{if } i \in I_2 \\ 1 & \text{if } i \in I_3\,, \end{cases}$$

where $I_1 = [s], I_2 = [s+1, s(\kappa+1)], I_3 = [s(\kappa+1)+1, s(\kappa^2+\kappa+1)]$, and $\boldsymbol{b}$ is defined as

$$b_i = \begin{cases} \kappa\sqrt{1-4\delta} & \text{if } i \in I_1 \\ \sqrt{\kappa}\sqrt{1-2\delta} & \text{if } i \in I_2 \\ 1 & \text{if } i \in I_3\,, \end{cases}$$

for some sufficiently small $\delta > 0$ used for tie-breaking. We define

$$x_i^* = \begin{cases} \kappa\sqrt{1-4\delta} & \text{if } i \in I_1 \\ 0 & \text{otherwise} \end{cases}$$

and, for some arbitrary $s'$-sized $S \subseteq I_3$

$$x_i = \begin{cases} 0 & \text{if } i \in I_1 \cup I_2 \cup I_3 \backslash S \\ 1 & \text{otherwise}\,. \end{cases}$$

Note that $f(\boldsymbol{x}) - f(\boldsymbol{x}^*) = 0.5s\kappa^2(1-4\delta) - 0.5s' \geq 0.1s\kappa^2$. Furthermore, the gradient is equal to

$$\begin{aligned} \nabla f(\boldsymbol{x}) &= \boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) \\ &= \begin{cases} -\kappa\sqrt{1-4\delta} & \text{if } i \in I_1 \\ -\kappa\sqrt{1-2\delta} & \text{if } i \in I_2 \\ -1 & \text{if } i \in I_3 \backslash S \\ 0 & \text{if } i \in S\,, \end{cases} \end{aligned}$$

and since we have $\beta = \kappa$,

$$\boldsymbol{x} - \beta^{-1}\nabla f(\boldsymbol{x}) = \begin{cases} \sqrt{1-4\delta} & \text{if } i \in I_1 \\ \sqrt{1-2\delta} & \text{if } i \in I_2 \\ 1/\kappa & \text{if } i \in I_3 \backslash S \\ 1 & \text{if } i \in S\,, \end{cases}$$

implying that $H_{s'}\left(\boldsymbol{x} - \beta^{-1}\nabla f(\boldsymbol{x})\right) = \boldsymbol{x}$. □

## 9.6 Appendix for Chapter 8

### Preliminaries and Notation

Given an positive integer $k$, we denote $[k] = \{1, 2, \ldots, k\}$. Given a matrix $A$, we denote by $\|A\|_F$ its Frobenius norm, i.e. the $\ell_2$ norm of the entries of $A$ (or equivalently of the singular values of $A$). The following lemma is a simple corollary of the definition of the Frobenius norm:

**Lemma 9.6.1.** *Given two matrices $A \in \mathbb{R}^{m\times n}$,$B \in \mathbb{R}^{m\times n}$, we have $\|A+B\|_F^2 \leq 2\left(\|A\|_F^2 + \|B\|_F^2\right)$.*

*Proof.*

$$\|A+B\|_F^2 = \sum_{ij}(A+B)_{ij}^2 \leq 2\sum_{ij}(A_{ij}^2 + B_{ij}^2) = 2(\|A\|_F^2 + \|B\|_F^2)$$

□

**Definition 9.6.2** (Rank-restricted smoothness, strong convexity, condition number)**.** *Given a convex function $R \in \mathbb{R}^{m\times n} \to \mathbb{R}$ and an integer parameter $r$, the* rank-restricted smoothness of $R$ at rank $r$ *is the minimum constant $\rho_r^+ \geq 0$ such that for any two matrices $A \in \mathbb{R}^{m\times n}, B \in \mathbb{R}^{m\times n}$ such that*

$\operatorname{rank}(A-B) \leq r$, *we have*

$$R(B) \leq R(A) + \langle \nabla R(A), B - A \rangle + \frac{\rho_r^+}{2}\|B - A\|_F^2 .$$

*Similarly, the* rank-restricted strong convexity of $R$ at rank $r$ *is the maximum constant* $\rho_r^- \geq 0$ *such that for any two matrices* $A \in \mathbb{R}^{m\times n}, B \in \mathbb{R}^{m\times n}$ *such that* $\operatorname{rank}(A-B) \leq r$, *we have*

$$R(B) \geq R(A) + \langle \nabla R(A), B - A \rangle + \frac{\rho_r^-}{2}\|B - A\|_F^2 .$$

*Given that* $\rho_r^+, \rho_r^-$ *exist and are nonzero, the* rank-restricted condition number of $R$ at rank $r$ *is then defined as*

$$\kappa_r = \frac{\rho_r^+}{\rho_r^-}$$

Note that $\rho_r^+$ is increasing and $\rho_r^-$ is decreasing in $r$. Therefore, even though our bounds are proven in terms of the constants $\frac{\rho_1^+}{\rho_r^-}$ and $\frac{\rho_2^+}{\rho_r^-}$, these quantities are always at most $\frac{\rho_r^+}{\rho_r^-} = \kappa_r$ as long as $r \geq 2$, and so they directly imply the same bounds in terms of the constant $\kappa_r$.

**Definition 9.6.3** (Spectral norm)**.** *Given a matrix* $A \in \mathbb{R}^{m\times n}$, *we denote its spectral norm by* $\|A\|_2$. *The spectral norm is defined as*

$$\|A\|_2 = \max_{x\in\mathbb{R}^n} \frac{\|Ax\|_2}{\|x\|_2} ,$$

**Definition 9.6.4** (Singular value thresholding operator)**.** *Given a matrix* $A \in \mathbb{R}^{m\times n}$ *of rank* $k$, *a singular value decomposition* $A = U\Sigma V^\top$ *such that* $\Sigma_{11} \geq \Sigma_{22} \geq \cdots \geq \Sigma_{kk}$, *and an integer* $1 \leq r \leq k$, *we define* $H_r(A) = U\Sigma' V^\top$, *here* $\Sigma'$ *is a diagonal matrix with*

$$\Sigma'_{ii} = \begin{cases} \Sigma_{ii} & \text{if } i \leq r \\ 0 & \text{otherwise} \end{cases}$$

*In other words,* $H_r(\cdot)$ *is an operator that eliminates all but the top* $r$ *highest singular values of a matrix.*

**Lemma 9.6.5** (Weyl's inequality)**.** *For any matrix* $A$ *and integer* $i \geq 1$, *let* $\sigma_i(A)$ *be the* $i$*-th largest singular value of* $A$ *or* $0$ *if* $i > \operatorname{rank}(A)$. *Then, for any two matrices* $A, B$ *and integers* $i \geq 1, j \geq 1$*:*

$$\sigma_{i+j-1}(A+B) \leq \sigma_i(A) + \sigma_j(B)$$

A proof of the previous fact can be found e.g. in [59].

**Lemma 9.6.6** ($H_r(\cdot)$ optimization problem)**.** *Let* $A \in \mathbb{R}^{m\times n}$ *be a rank-*$k$ *matrix and* $r \in [k]$ *be an integer parameter. Then* $M = \frac{1}{\lambda}H_r(A)$ *is an optimal solution to the following optimization problem:*

$$\max_{\operatorname{rank}(M)\leq r} \{\langle A, M\rangle - \frac{\lambda}{2}\|M\|_F^2\} \tag{9.84}$$

*Proof.* Let $U\Sigma V^\top = \sum_i \Sigma_{ii} U_i V_i^\top$ be a singular value decomposition of $A$. We note that (9.84) is equivalent to

$$\min_{\text{rank}(M)\leq r} \|A - \lambda M\|_F^2 := f(M) \tag{9.85}$$

Now, note that $f(\frac{1}{\lambda}H_r(A)) = \|A - H_r(A)\|_F^2 = \sum_{i=r+1}^{k} \Sigma_{ii}^2$. On the other hand, by applying Weyl's inequality (Lemma 9.6.5) for $j = r+1$,

$$f(M) = \|A - \lambda M\|_F^2 = \sum_{i=1}^{k+r} \sigma_i^2(A - \lambda M) \geq \sum_{i=1}^{k+r} (\sigma_{i+r}(A) - \sigma_{r+1}(\lambda M))^2 = \sum_{i=r+1}^{k} \Sigma_{ii}^2\,,$$

where the last equality follows from the fact that $\text{rank}(A) = k$ and $\text{rank}(M) \leq r$. Therefore, $M = \frac{1}{\lambda}H_r(A)$ minimizes (9.85) and thus maximizes (9.84). □

**Proof of Theorem 8.2.1 (greedy)**

We will start with the following simple lemma about the Frobenius norm of a sum of matrices with orthogonal columns or rows:

**Lemma 9.6.7.** *Let $U \in \mathbb{R}^{m\times r}, V \in \mathbb{R}^{n\times r}, X \in \mathbb{R}^{m\times r}, Y \in \mathbb{R}^{n\times r}$ be such that the columns of $U$ are orthogonal to the columns of $X$* or *the columns of $V$ are orthogonal to the columns of $Y$. Then $\|UV^\top + XY^\top\|_F^2 = \|UV^\top\|_F^2 + \|XY^\top\|_F^2$.*

*Proof.* If the columns of $U$ are orthogonal to those of $X$, then $U^\top X = \mathbf{0}$ and if the columns of $V$ are orthogonal to those of $Y$, then $Y^\top V = \mathbf{0}$. Therefore in any case $\langle UV^\top, XY^\top\rangle = \text{Tr}(VU^\top XY^\top) = \text{Tr}(U^\top XY^\top V) = \mathbf{0}$, implying

$$\|UV^\top + XY^\top\|_F^2 = \|UV^\top\|_F^2 + \|XY^\top\|_F^2 + 2\langle UV^\top, XY^\top\rangle = \|UV^\top\|_F^2 + \|XY^\top\|_F^2$$

□

Additionally, we have the following lemma regarding the optimality conditions of (8.2):

**Lemma 9.6.8.** *Let $A = UXV^\top$ where $U \in \mathbb{R}^{m\times r}$, $X \in \mathbb{R}^{r\times r}$, and $V \in \mathbb{R}^{n\times r}$, such that $X$ is the optimal solution to (8.2). Then for any $u \in \text{im}(U)$ and $v \in \text{im}(V)$ we have that $\langle \nabla R(A), uv^\top\rangle = 0$.*

*Proof.* By the optimality condition of 8.2, we have that

$$U^\top \nabla R(A) V = \mathbf{0}$$

Now, for any $u = Ux$ and $v = Vy$ we have

$$\langle \nabla R(A), uv^\top\rangle = u^\top \nabla R(A) v = x^\top U^\top \nabla R(A) V y = 0$$

□

We are now ready for the proof of Theorem 8.2.1.

*Proof.* Let $A_{t-1}$ be the current solution $UV^\top$ before iteration $t-1 \geq 0$. Let $u \in \mathbb{R}^m$ and $v \in \mathbb{R}^m$ be left and right singular vectors of matrix $\nabla R(A)$, i.e. unit vectors maximizing $|\langle \nabla R(A), uv^\top \rangle|$. Let

$$\mathcal{B}_t = \{B | B = A_{t-1} + \eta uv^T, \eta \in \mathbb{R}\}.$$

By smoothness we have

$$\begin{aligned}
R(A_{t-1}) - R(A_t) &\geq \max_{B \in \mathcal{B}_t} \{R(A_{t-1}) - R(B)\} \\
&\geq \max_{B \in \mathcal{B}_t} \left\{ -\langle \nabla R(A_{t-1}), B - A_{t-1} \rangle - \frac{\rho_1^+}{2} \|B - A_{t-1}\|_F^2 \right\} \\
&\geq \max_{\eta} \left\{ \eta \langle \nabla R(A_{t-1}), uv^\top \rangle - \eta^2 \frac{\rho_1^+}{2} \right\} \\
&= \max_{\eta} \left\{ \eta \|\nabla R(A_{t-1})\|_2 - \eta^2 \frac{\rho_1^+}{2} \right\} \\
&= \frac{\|\nabla R(A_{t-1})\|_2^2}{2\rho_1^+}
\end{aligned}$$

where $\|\cdot\|_2$ is the spectral norm (i.e. maximum magnitude of a singular value).

On the other hand, by strong convexity and noting that

$$\begin{aligned}
&\operatorname{rank}(A^* - A_{t-1}) \leq \operatorname{rank}(A^*) + \operatorname{rank}(A_{t-1}) \leq r^* + r\,, \\
&R(A^*) - R(A_{t-1}) \geq \langle \nabla R(A_{t-1}), A^* - A_{t-1} \rangle + \frac{\rho_{r+r^*}^-}{2} \|A^* - A_{t-1}\|_F^2\,.
\end{aligned} \tag{9.86}$$

Let $A_{t-1} = UV^\top$ and $A^* = U^*V^{*\top}$. We let $\Pi_{\mathrm{im}(U)} = U(U^\top U)^+ U^\top$ and $\Pi_{\mathrm{im}(V)} = V(V^\top V)^+ V^\top$ denote the orthogonal projections onto the images of $U$ and $V$ respectively. We now write

$$A^* = U^*V^{*\top} = (U^1 + U^2)(V^1 + V^2)^\top = U^1V^{1\top} + U^1V^{2\top} + U^2V^{*\top}$$

where $U^1 = \Pi_{\mathrm{im}(U)} U^*$ is a matrix where every column of $U^*$ is replaced by its projection on $\mathrm{im}(U)$ and $U^2 = U^* - U^1$ and similarly $V^1 = \Pi_{\mathrm{im}(V)} V^*$ is a matrix where every column of $V^*$ is replaced by its projection on $\mathrm{im}(V)$ and $V^2 = V^* - V^1$. By setting $U' = (-U \mid U^1)$ and $V' = (V \mid V^1)$ we can write

$$A^* - A_{t-1} = U'V'^\top + U^1V^{2\top} + U^2V^{*\top}$$

where $\mathrm{im}(U') = \mathrm{im}(U)$ and $\mathrm{im}(V') = \mathrm{im}(V)$. Also, note that

$$\operatorname{rank}(U^1V^{2\top}) \leq \operatorname{rank}(V^2) \leq \operatorname{rank}(V^*) = \operatorname{rank}(A^*) \leq r^*$$

and similarly $\operatorname{rank}(U^2V^{*\top}) \leq r^*$. So now the right hand side of (9.86) can be reshaped as

$$\begin{aligned}
&\langle \nabla R(A_{t-1}), A^* - A_{t-1} \rangle + \frac{\rho_{r+r^*}^-}{2} \|A^* - A_{t-1}\|_F^2 \\
&= \langle \nabla R(A_{t-1}), U'V'^\top + U^1V^{2\top} + U^2V^{*\top} \rangle + \frac{\rho_{r+r^*}^-}{2} \|U'V'^\top + U^1V^{2\top} + U^2V^{*\top}\|_F^2
\end{aligned}$$

Now, note that since by definition the columns of $U'$ are in $\mathrm{im}(U)$ and the columns of $V'$ are in

$\mathrm{im}(V)$, Lemma 9.6.8 implies that $\langle \nabla R(A_{t-1}), U'V'^{\top} \rangle = 0$. Therefore the above is equal to

$$\begin{aligned}
&\langle \nabla R(A_{t-1}), U^1V^{2\top} + U^2V^{*\top} \rangle + \frac{\rho^-_{r+r^*}}{2} \|U'V'^{\top} + U^1V^{2\top} + U^2V^{*\top}\|_F^2 \\
&\geq \langle \nabla R(A_{t-1}), U^1V^{2\top} \rangle + \langle \nabla R(A_{t-1}), U^2V^{*\top} \rangle + \frac{\rho^-_{r+r^*}}{2} \left( \|U^1V^{2\top}\|_F^2 + \|U^2V^{*\top}\|_F^2 \right) \\
&\geq 2 \min_{\mathrm{rank}(M) \leq r^*} \left\{ \langle \nabla R(A_{t-1}), M \rangle + \frac{\rho^-_{r+r^*}}{2} \|M\|_F^2 \right\} \\
&= -2 \frac{\|H_{r^*}(\nabla R(A_{t-1}))\|_F^2}{2\rho^-_{r+r^*}} \\
&\geq -r^* \frac{\|\nabla R(A_{t-1})\|_2^2}{\rho^-_{r+r^*}}
\end{aligned}$$

where the first equality follows by noticing that the columns of $V'$ and $V^1$ are orthogonal to those of $V^2$ and the columns of $U'$ and $U^1$ are orthogonal to those of $U^2$, and applying Lemma 9.6.7. The last equality is a direct application of Lemma 9.6.6 and the last inequality states that the largest squared singular value is not smaller than the average of the top $r^*$ squared singular values. Therefore we have concluded that

$$\|\nabla R(A_{t-1})\|_2^2 \geq \frac{\rho^-_{r+r^*}}{r^*} \left( R(A_{t-1}) - R(A^*) \right)$$

Plugging this back into the smoothness inequality, we get

$$R(A_{t-1}) - R(A_t) \geq \frac{1}{2r^*\kappa} (R(A_{t-1}) - R(A^*))$$

or equivalently

$$R(A_t) - R(A^*) \leq \left( 1 - \frac{1}{2r^*\kappa} \right) \left( R(A_{t-1}) - R(A^*) \right).$$

Therefore after $L = 2r^*\kappa \log \frac{R(A_0) - R(A^*)}{\varepsilon}$ iterations we have

$$\begin{aligned}
R(A_T) - R(A^*) &\leq \left( 1 - \frac{1}{2r^*\kappa} \right)^L (R(A_0) - R(A^*)) \\
&\leq e^{-\frac{L}{2r^*\kappa}} (R(A_0) - R(A^*)) \\
&\leq \varepsilon
\end{aligned}$$

Since $A_0 = \mathbf{0}$, the result follows. □

**Proof of Theorem 8.2.2 (local search)**

*Proof.* Similarly to Section 9.6, we let $A_{t-1}$ be the current solution before iteration $t - 1 \geq 0$. Let $u \in \mathbb{R}^m$ and $v \in \mathbb{R}^m$ be left and right singular vectors of matrix $\nabla R(A)$, i.e. unit vectors maximizing $|\langle \nabla R(A), uv^{\top} \rangle|$ and let

$$\mathcal{B}_t = \{B | B = A_{t-1} + \eta uv^T - \sigma_{\min} xy^{\top}, \eta \in \mathbb{R}\},$$

where $\sigma_{\min} xy^\top = A_{t-1} - H_{r-1}(A_{t-1})$ is the rank-1 term corresponding to the minimum singular value of $A_{t-1}$. By smoothness we have

$$\begin{aligned}
&R(A_{t-1}) - R(A_t) \\
&\geq \max_{B \in \mathcal{B}_t} \{R(A_{t-1}) - R(B)\} \\
&\geq \max_{B \in \mathcal{B}_t} \left\{ -\langle \nabla R(A_{t-1}), B - A_{t-1} \rangle - \frac{\rho_2^+}{2} \|B - A_{t-1}\|_F^2 \right\} \\
&= \max_{\eta \in \mathbb{R}} \left\{ -\langle \nabla R(A_{t-1}), \eta uv^\top - \sigma_{\min} xy^\top \rangle - \frac{\rho_2^+}{2} \|\eta uv^\top - \sigma_{\min} xy^\top\|_F^2 \right\} \\
&\geq \max_{\eta \in \mathbb{R}} \left\{ -\langle \nabla R(A_{t-1}), \eta uv^\top \rangle - \eta^2 \rho_2^+ - \sigma_{\min}^2 \rho_2^+ \right\} \\
&= \max_{\eta \in \mathbb{R}} \left\{ \eta \|\nabla R(A_{t-1})\|_2 - \eta^2 \rho_2^+ - \sigma_{\min}^2 \rho_2^+ \right\} \\
&= \frac{\|\nabla R(A_{t-1})\|_2^2}{4\rho_2^+} - \sigma_{\min}^2 \rho_2^+ \,,
\end{aligned}$$

where in the last inequality we used the fact that $\langle \nabla R(A_{t-1}), xy^\top \rangle = 0$ following from Lemma 9.6.8, as well as Lemma 9.6.1.

On the other hand, by strong convexity,

$$R(A^*) - R(A_{t-1}) \geq \langle \nabla R(A_{t-1}), A^* - A_{t-1} \rangle + \frac{\rho_{r+r^*}^-}{2} \|A^* - A_{t-1}\|_F^2 \,.$$

Let $A_{t-1} = UV^\top$ and $A^* = U^* V^{*\top}$. We write

$$A^* = U^* V^{*\top} = (U^1 + U^2)(V^1 + V^2)^\top = U^1 V^{1\top} + U^1 V^{2\top} + U^2 V^{*\top}$$

where $U^1$ is a matrix where every column of $U^*$ is replaced by its projection on $\mathrm{im}(U)$ and $U^2 = U^* - U^1$ and similarly $V^1$ is a matrix where every column of $V^*$ is replaced by its projection on $\mathrm{im}(V)$ and $V^2 = V^* - V^1$. By setting $U' = (-U \mid U^1)$ and $V' = (V \mid V^1)$ we can write

$$A^* - A_{t-1} = U' V'^\top + U^1 V^{2\top} + U^2 V^{*\top}$$

where $\mathrm{im}(U') = \mathrm{im}(U)$ and $\mathrm{im}(V') = \mathrm{im}(V)$. Also, note that

$$\mathrm{rank}(U^1 V^{2\top}) \leq \mathrm{rank}(V^2) \leq \mathrm{rank}(V^*) = \mathrm{rank}(A^*) \leq r^*$$

and similarly $\operatorname{rank}(U^2 V^{*\top}) \leq r^*$. So we now have

$$
\begin{aligned}
&\langle \nabla R(A_{t-1}), A^* - A_{t-1} \rangle + \frac{\rho^-_{r+r^*}}{2} \|A^* - A_{t-1}\|_F^2 \\
&= \langle \nabla R(A_{t-1}), U' V'^\top + U^1 V^{2\top} + U^2 V^{*\top} \rangle + \frac{\rho^-_{r+r^*}}{2} \|U' V'^\top + U^1 V^{2\top} + U^2 V^{*\top}\|_F^2 \\
&= \langle \nabla R(A_{t-1}), U^1 V^{2\top} + U^2 V^{*\top} \rangle + \frac{\rho^-_{r+r^*}}{2} \|U' V'^\top + U^1 V^{2\top} + U^2 V^{*\top}\|_F^2 \\
&= \langle \nabla R(A_{t-1}), U^1 V^{2\top} + U^2 V^{*\top} \rangle + \frac{\rho^-_{r+r^*}}{2} \left( \|U' V'^\top\|_F^2 + \|U^1 V^{2\top}\|_F^2 + \|U^2 V^{*\top}\|_F^2 \right) \\
&\geq \langle \nabla R(A_{t-1}), U^1 V^{2\top} \rangle + \langle \nabla R(A_{t-1}), U^2 V^{*\top} \rangle + \frac{\rho^-_{r+r^*}}{2} \left( \|U^1 V^{2\top}\|_F^2 + \|U^2 V^{*\top}\|_F^2 \right) \\
&+ \frac{\rho^-_{r+r^*}}{2} \|U' V'^\top\|_F^2 \\
&\geq 2 \min_{\operatorname{rank}(M) \leq r^*} \left\{ \langle \nabla R(A_{t-1}), M \rangle + \frac{\rho^-_{r+r^*}}{2} \|M\|_F^2 \right\} + \frac{\rho^-_{r+r^*}}{2} \|U' V'^\top\|_F^2 \\
&= -2 \frac{\|H_{r^*}(\nabla R(A_{t-1}))\|_F^2}{2\rho^-_{r+r^*}} + \frac{\rho^-_{r+r^*}}{2} \|U' V'^\top\|_F^2 \\
&\geq -r^* \frac{\|\nabla R(A_{t-1})\|_2^2}{\rho^-_{r+r^*}} + \frac{\rho^-_{r+r^*}}{2} \|U' V'^\top\|_F^2
\end{aligned}
$$

where the second equality follows from the fact that $\langle \nabla R(A_{t-1}), uv^\top \rangle = 0$ for any $u \in \operatorname{im}(U), v \in \operatorname{im}(V)$, the third equality from the fact that $\operatorname{im}(U^2) \perp \operatorname{im}(U') \cup \operatorname{im}(U^1)$ and $\operatorname{im}(V^2) \perp \operatorname{im}(V')$ and by applying Lemma 9.6.7, and the last inequality from the fact that the largest squared singular value is not smaller than the average of the top $r^*$ squared singular values. Now, note that since $\operatorname{rank}(U^1 V^{1\top}) \leq r^* < r = \operatorname{rank}(UV^\top)$,

$$
\begin{aligned}
\|U' V'^\top\|_F^2 &= \|U^1 V^{1\top} - UV^\top\|_F^2 \\
&= \sum_{i=1}^{r} \sigma_i^2 (U^1 V^{1\top} - UV^\top) \\
&\geq \sum_{i=1}^{r} (\sigma_{i+r^*}(UV^\top) - \sigma_{r^*+1}(U^1 V^{1\top}))^2 \\
&= \sum_{i=r^*+1}^{r} \sigma_i^2 (UV^\top) \\
&\geq (r - r^*) \sigma_{\min}^2 (UV^\top) \\
&= (r - r^*) \sigma_{\min}^2 (A_{t-1}) \,,
\end{aligned}
$$

where we used the fact that $\operatorname{rank}(U^1 V^{1\top}) \leq r^*$ together with Lemma 9.6.5. Therefore we have concluded that

$$
\|\nabla R(A_{t-1})\|_2^2 \geq \frac{\rho^-_{r+r^*}}{r^*} \left( R(A_{t-1}) - R(A^*) \right) + \frac{(\rho^-_{r+r^*})^2 (r - r^*)}{2r^*} \sigma_{\min}^2
$$

Plugging this back into the smoothness inequality and setting $\widetilde{\kappa} = \frac{\rho_2^+}{\rho_{r+r^*}^-}$, we get

$$\begin{aligned} R(A_{t-1}) - R(A_t) &\geq \frac{1}{4r^*\widetilde{\kappa}}(R(A_{t-1}) - R(A^*)) + \left(\frac{\rho_{r+r^*}^-(r-r^*)}{8r^*\widetilde{\kappa}} - \rho_2^+\right)\sigma_{\min}^2(A_{t-1}) \\ &\geq \frac{1}{4r^*\widetilde{\kappa}}(R(A_{t-1}) - R(A^*)) \end{aligned}$$

as long as $r \geq r^*(1 + 8\widetilde{\kappa}^2)$, or equivalently,

$$R(A_t) - R(A^*) \leq \left(1 - \frac{1}{4r^*\widetilde{\kappa}}\right)(R(A_{t-1}) - R(A^*)).$$

Therefore after $L = 4r^*\widetilde{\kappa}\log\frac{R(A_0)-R(A^*)}{\varepsilon}$ iterations we have

$$\begin{aligned} R(A_T) - R(A^*) &\leq \left(1 - \frac{1}{4r^*\widetilde{\kappa}}\right)^L (R(A_0) - R(A^*)) \\ &\leq e^{-\frac{L}{4r^*\widetilde{\kappa}}}(R(A_0) - R(A^*)) \\ &\leq \varepsilon \end{aligned}$$

Since $A_0 = \mathbf{0}$ and $\widetilde{\kappa} \leq \kappa_{r+r^*}$, the result follows. □

**Tightness of the analysis**

It is important to note that the $\kappa_{r+r^*}$ factor that appears in the rank bounds of both Theorems 8.2.1 and 8.2.2 is inherent in these algorithms and not an artifact of our analysis. In particular, such lower bounds based on the restricted condition number have been previously shown for the problem of sparse linear regression. More specifically, [63] showed that there is a family of instances in which the analogues of Greedy and Local Search for sparse optimization require the sparsity to be $\Omega(s^*\kappa')$ for constant error $\varepsilon > 0$, where $s^*$ is the optimal sparsity and $\kappa'$ is the *sparsity*-restricted condition number. These instances can be easily adjusted to give a *rank* lower bound of $\Omega(r^*\kappa_{r+r^*})$ for constant error $\varepsilon > 0$, implying that the $\kappa$ dependence in Theorem 8.2.1 is tight for Greedy. Furthermore, specifically for Local Search, [14] additionally showed that there is a family of instances in which the analogue of Local Search for sparse optimization requires a sparsity of $\Omega(s^*(\kappa')^2)$. Adapting these instances to the setting of rank-constrained convex optimization is less trivial, but we conjecture that it is possible, which would lead to a rank lower bound of $\Omega(r^*\kappa_{r+r^*}^2)$ for Local Search.

We present the following lemma, which essentially states that sparse optimization lower bounds for Orthogonal Matching Pursuit (OMP, [135]) (resp. Orthogonal Matching Pursuit with Replacement (OMPR, [87])) in which the optimal sparse solution is also a global optimum, immediately carry over (up to constants) to rank-constrained convex optimization lower bounds for Greedy (resp. Local Search).

**Lemma 9.6.9.** *Let $f \in \mathbb{R}^n \to \mathbb{R}$ and $x^* \in \mathbb{R}^n$ be an $s^*$-sparse vector that is also a global minimizer of $f$. Also, let $f$ have restricted smoothness parameter $\beta$ at sparsity level $s + s^*$ for some $s \geq s^*$ and restricted strong convexity parameter $\alpha$ at sparsity level $s+s^*$. Then we can define the rank-constrained problem, with $R : \mathbb{R}^{n\times n} \to \mathbb{R}$,*

$$\min_{\text{rank}(A)\leq s^*} R(A) := f(\text{diag}(A)) + \frac{\beta}{2}\|A - \text{diag}(A)\|_F^2, \tag{9.87}$$

*where* $\operatorname{diag}(A)$ *is a vector containing the diagonal of* $A$*. $R$ has rank-restricted smoothness at rank $s + s^*$ at most $2\beta$ and rank-restricted strong convexity at rank $s + s^*$ at least $\alpha$. Suppose that we run $t$ iterations of OMP (resp. OMPR) starting from a solution $x$, to get solution $x'$, and similarly run $t$ iterations of Greedy (resp. Local Search) starting from solution $A = \operatorname{diag}(x)$ (where* $\operatorname{diag}(x)$ *is a diagonal matrix with $x$ on the diagonal) to get solution $A'$. Then $A'$ is diagonal and* $\operatorname{diag}(A') = x'$*. In other words, in this scenario OMP and Greedy (resp. OMPR and Local Search) are equivalent.*

*Proof.* Note that for any solution $\widehat{A}$ of $R$ we have $R(\widehat{A}) \geq f(\operatorname{diag}(\widehat{A})) \geq f(x^*)$, with equality only if $\widehat{A}$ is diagonal. Furthermore, $\operatorname{rank}(\operatorname{diag}(x^*)) \leq s^*$, meaning that $\operatorname{diag}(x^*)$ is an optimal solution of (9.87). Now, given any diagonal solution $A$ of (9.87) such that $A = \operatorname{diag}(x)$, we claim that one step of either Greedy or Local Search keeps it diagonal. This is because

$$\nabla R(A) = \operatorname{diag}(\nabla f(x)) + \frac{\beta}{2}(A - \operatorname{diag}(A)) = \operatorname{diag}(\nabla f(x))\,.$$

Therefore the largest eigenvalue of $\nabla R(A)$ has corresponding eigenvector $\mathbf{1}_i$ for some $i$, which implies that the rank-1 component which will be added is a multiple of $\mathbf{1}_i\mathbf{1}_i^\top$. For the same reason the rank-1 component removed by Local Search will be a multiple of $\mathbf{1}_j\mathbf{1}_j^\top$ for some $j$. Therefore running Greedy (resp. Local Search) on such an instance is identical to running OMP (resp. OMPR) on the diagonal. □

Together with the lower bound instances of [63] (in which the global minimum property is true), it immediately implies a rank lower bound of $\Omega(r^*\kappa_{r+r^*})$ for getting a solution with constant error for rank-constrained convex optimization. On the other hand, the lower bound instances of [14] give a *quadratic* lower bound in $\kappa$ for OMPR. The above lemma cannot be directly applied since the sparse solutions are not global minima, but we conjecture that a similar proof will give a rank lower bound of $\Omega(r^*\kappa_{r+r^*}^2)$ for rank-constrained convex optimization with Local Search.

**Addendum to Section 8.4**

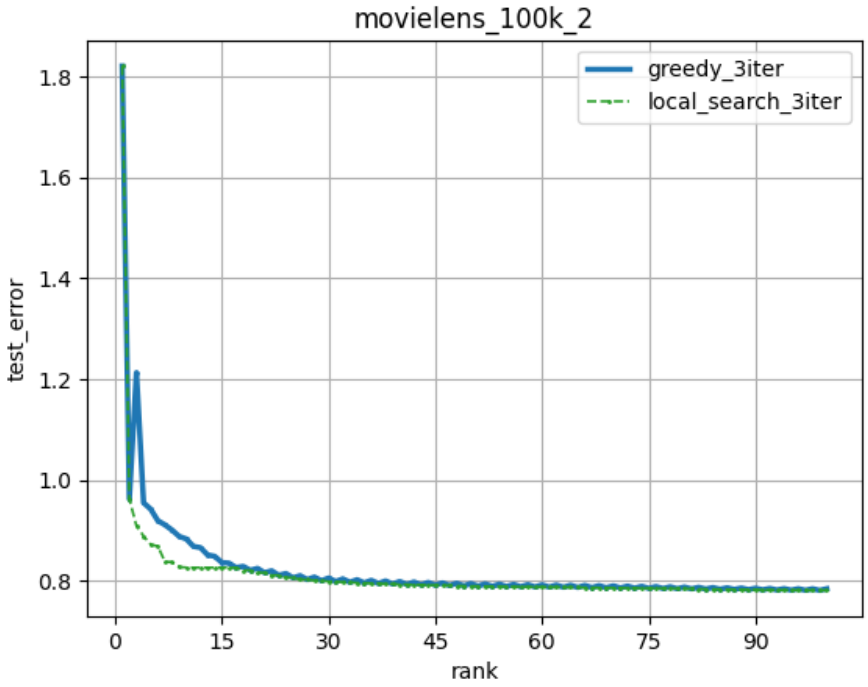
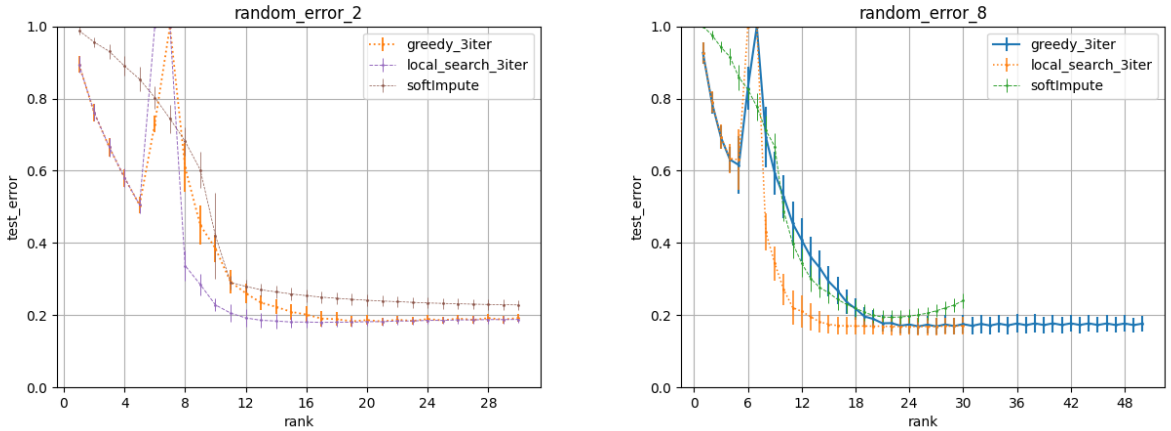


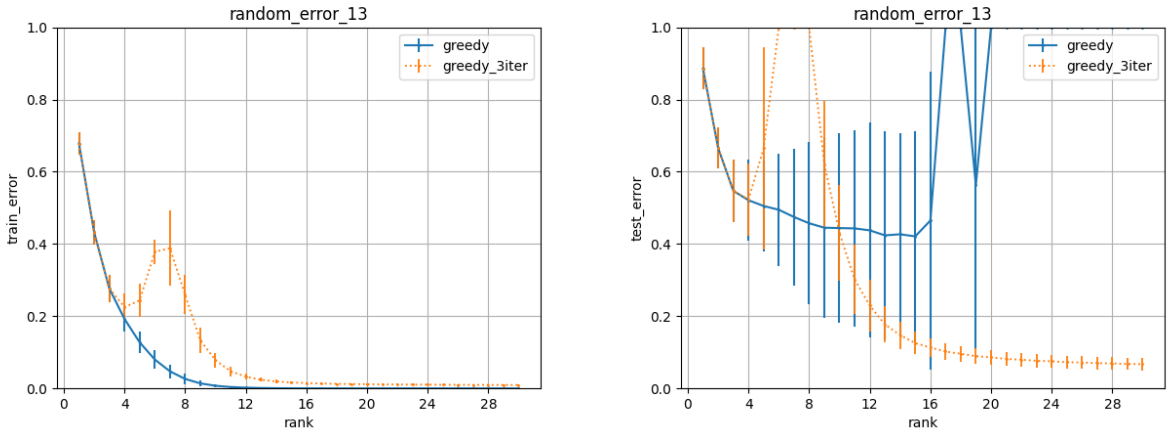

Figure 9-2: One of the splits of the Movielens 100K dataset. We can see that for small ranks the Fast Local Search solution is better and more stable, but for larger ranks it does not provide any improvement over the Fast Greedy algorithm.




(a) $k = 10$, $p = 0.5$, $SNR = 1$

(b) $k = 10$, $p = 0.3$, $SNR = 3$

Figure 9-3: Test error vs rank in the matrix completion problem of Section 8.4.2. Bands of $\pm 1$ standard error are shown.




(a) Train error vs rank

(b) Test error vs rank

Figure 9-4: Performance of greedy with fully solving the inner optimization problem (left) and applying 3 iterations of the LSQR algorithm (right) in the matrix completion problem of Section 8.4.2. $k = 5$, $p = 0.2$, $SNR = 10$. Bands of $\pm 1$ standard error are shown. This experiment shows why it is crucial to apply some kind of regularization to the Fast Greedy and Fast Local Search algorithms for machine learning applications.