New Techniques for Continuous Optimization and Fast Algorithms for Flow

Jonah Sherman

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2017-232
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2017/EECS-2017-232.html

December 18, 2017

New Techniques for Continuous Optimization and Fast Algorithms for Flow

By

Jonah Sherman

A thesis submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy
in
Computer Science
in the
Graduate Division
of the
University of California, Berkeley

Committee in charge:

Professor Umesh Vazirani, Chair
Professor Satish Rao
Professor Ilan Adler

Fall 2017

Abstract

New Techniques for Continuous Optimization and Fast Algorithms for Flow

by

Jonah Sherman

Doctor of Philosophy in Computer Science
University of California, Berkeley
Professor Umesh Vazirani, Chair

Spielman and Teng's nearly linear time algorithm for solving Laplacian systems was a breakthrough in combining techniques from discrete and continuous optimization. In this dissertation we adapt their approach to solve more general flow problems, overcoming long-standing barriers in continuous optimization. As a result we solve longstanding open questions in optimization by presenting nearly-linear time approximation algorithms for the undirected versions of several network flow problems, including *maximum flow*, *optimal transportation*, *maximum concurrent multi-commodity flow*, and *sparsest cut* problems.

Our contributions include two general tools. The first is a scheme for boosting the performance of iterative solvers for continuous optimization problems. These boosters use recursive composition to achieve residual error that decreases exponentially in the number of iterations of the solver.

The second is a technique for designing iterative solvers that bypasses the infamous ℓ_∞ barrier for strongly convex regularization. Instead we introduce the notion of an area convex regularizer, and show that such regularizers suffice in designing good iterative solvers. Moreover, we show how to design regularizers satisfying this weaker property for the problem of approximately solving $AX \leq B$ over right stochastic X, which in turn implies the algorithm for maximum concurrent multi-commodity flow.

Contents

Chapter 1

Introduction

The evolution of the fields of continuous optimization and combinatorial optimization have been largely disjoint, using a very different set of tools and techniques, until the seminal work of Spielman and Teng[47], leading to a flurry of activity[30, 18, 33, 17, 44, 26, 45, 41]. This dissertation makes the connection between these two fields even deeper, in the process resolving long standing open questions in combinatorial optimization by presenting nearly-linear time approximation algorithms for the undirected versions of several network flow problems, including *maximum concurrent multi-commodity flow*, *optimal transportation*, and *sparsest cut*.

Fixed-demand network flow problems are a special case of the following kind of continuous optimization problem: find x of minimum norm $\|x\|$ satisfying a given linear system $Ax = b$. The norm depends on the particular problem of interest. Continuous optimization algorithms such as fast classical iterative solvers use dual methods and relax the equality constraints to instead yield x with $\|x\| \leq \|x_{opt}\|$ satisfying $Ax - b \leq \delta\|A\|\|x_{\mathsf{opt}}\|$. For flow problems the equality constraints represent flow conservation and must be satisfied exactly — an approximate solutions means finding $x : \|x\| \leq (1+\epsilon)\|x_{opt}\|$. Our overall approach to using continuous methods for flow problems is inspired by Spielman and Teng[47]'s nearly-linear time algorithm for solving Laplacian systems, equivalent to the Euclidean norm case. Euclidean minimum-norm problems are well-understood; classical iterative solvers such as steepest descent and conjugate gradient methods produce approximately optimal solutions with residual error $\delta = \|Ax - b\|$ exponentially small in iteration count, with the exact rate depending strongly on the *condition number* of the square matrix AA^* [1]. *Preconditioning* transforms the problem to have better condition number. Spielman and Teng's main contribution is a nearly-linear time algorithm for constructing good preconditioners for graph Laplacians. Our algorithms are obtained by attempting to "port" their algorithm from ℓ_2 to ℓ_1, ℓ_∞, and other norms.

The first obstacle to doing so is that existing solvers for non-Euclidean problems either require $t = n^{\Omega(1)}$ iterations (e.g. interior point) or converge much more slowly, with residual error $\delta = t^{-O(1)}$. Our first contribution, in chapter 2, is a scheme for boosting

[1] solutions with polynomially small error can be rounded to achieve exact equality, and therefore require only logarithmic number of iterations

the convergence of such weak iterative solvers via composition to get the desired residual error exponentially small in iteration count. This leads to a framework for designing fast combinatorial algorithms, using three parts:

1. Preconditioner
2. Weak iterative solver
3. Boosting via composition

Max-flow and optimal transportation correspond to ℓ_∞ and ℓ_1 norms, for which the multiplicative weights method[42, 23, 5] achieves residual error $\|Ax - b\|$ polynomially small in the iteration count. By stopping such solvers once the error is small enough and recursively composing, the error may be instead reduced to be exponentially small. The remaining miniscule error is then eliminated by routing it naively through a spanning tree.

The remaining obstacle for nearly linear time algorithms for max-flow and min-cost flow/transportation problems is the design of suitable combinatorial problem-specific preconditioner constructions. A preconditioner may be viewed as a κ-factor approximation to the *value* of a minimum-norm problem. Following Spielman and Teng[47], the main paradigm is to combine two ideas: approximation, and hierarchical routing. Approximation replaces G with a simpler $G' \approx G$; hierarchical routing recursively reduces flow problems on G to flow problems on smaller subgraphs of G. To precondition max-flow, we estimate the congestion required to route some demands in a capacitated graph by checking the congestion across a specific family of cuts. The cuts come from recursively applying decompositions due to Madry[33] utilizing techniques from Spielman and Teng[47]. To precondition transportation, we embed the cost metric into ℓ_1, and then design a simple hierarchical routing scheme inspired by the multigrid algorithm for grid Laplacians, and the Barnes-Hut algorithm for n-body simulation[8]. We present the preconditioner constructions in chapter 3.

Maximum-flow is used as a subroutine in other graph algorithms, including the *Sparsest Cut* problem. Prior work on that problem follows two lines. The first[7, 4] concerns the best approximation-factor achievable in polynomial time. The second[28, 40] concerns fast, practical algorithms that compute sparse cuts using a small number of max-flows. In chapter 5, we tie those two lines together, presenting an algorithm achieving the best-known approximation factor of the former while using only a small number of max-flows, as in the latter. Using our nearly-linear time maximum flow algorithm yields a nearly-linear time approximation to sparsest cut achieving the best known asymptotic approximation factor.

For multicommodity maximum concurrent flow, the relevant norm is a ℓ_∞-operator norm, for which nearly-linear time solvers are not known, even with small constant error. This is the infamous ℓ_∞ barrier for the *strongly-convex* regularization approach of Nesterov [37]. That barrier is responsible for stalled progress on algorithms for several important optimization problems. For single-commodity flow and transportation, this is a barrier towards achieving an ϵ^{-1} dependance of running time on the error. For multi-commodity flow, the consequences are even more severe, preventing nearly-linear time altogether. In chapter 4, we show how to break that barrier by using regularizers satisfying a dramatically weaker property we call

area-convexity. Using area-convex regularization, we obtain a nearly-linear time approximation algorithm for maximum concurrent multicommodity flow, and more generally, a fast algorithm for approximately solving $AX \leq B$ over right-stochastic X.

1.1 Solver Algorithm and Interpretation

The algorithm proceeds by approximately solving a saddle point problem

$$\min_{x \in \mathcal{X}} \max_{y \in \mathcal{Y}} y \cdot (b - Ax)$$

over two compact-convex sets $\mathcal{X}, \mathcal{Y}$.

By a standard reduction, we may reduce our attention to *canonical* problems of the form,

$$\max_{q \in \mathcal{Q}} \min_{p \in \mathcal{P}} q \cdot p$$

Nesterov's smoothing approximates a non-differentiable objective function

$$\eta(p) = \max_{q \in \mathcal{Q}} q \cdot p$$

using a strictly convex *regularizer* $\phi : \mathcal{Q} \to \mathbb{R}$, and considering,

$$\tilde{\eta}(p) = \max_{q \in \mathcal{Q}} q \cdot p - \phi(q)$$

The regularizer ϕ is chosen to balance two tradeoffs. First, it should be small, so $\tilde{\eta} \approx \eta$. Simultaneously, it should be *strongly convex*, so that $\tilde{\eta}$ is smooth, allowing first-order optimization (e.g. steepest descent) to be applied. Thus, applicability of the method depends significantly on whether such small strongly convex regularizers exist for the set of interest.

Saddle point problems may be viewed as a two-player zero-sum game, and regularization-based algorithms may be viewed as simulating repeated play of that game using adaptive policies. For $t = 0, 1, \ldots$, the $\mathcal{Q}$ player chooses $q'(t) \in \mathcal{Q}$, the $\mathcal{P}$ player chooses $p'(t) \in \mathcal{P}$, and the $\mathcal{Q}$ player receives payoff $q'(t) \cdot p'(t)$. The algorithms maintain cumulative history,

$$(q, p)(t) = \sum_{s=0}^{t-1} (q', p')(s)$$

Players use a *regularized best-response*:

$$\begin{aligned} q'(t) &= \arg\max_{q' \in \mathcal{Q}} q' \cdot p(t) - \phi_1(q') \\ p'(t) &= \arg\min_{p' \in \mathcal{P}} p' \cdot q(t) + \phi_2(p') \end{aligned}$$

After T steps, the algorithm outputs *average* history: $\frac{1}{T}(q, p)(T)$.

More concisely, the sequence $z(t) = (q(t), p(t))$ is defined by $z(0) = 0$ and

$$z(t+1) - z(t) = F(z(t))$$

where F is *response function*

$$F(q,p) = \arg \max_{q' \in \mathcal{Q}, p' \in \mathcal{P}} q' \cdot p - p' \cdot q - \phi_1(q') - \phi_2(p')$$

A large family of algorithms follow this paradigm, using variations of the update rule.

- The naive algorithm just described follows

$$z(t+1) - z(t) = F(z(t))$$

- Nemirovski's algorithm[36] uses the response to the *future* state, instead of the present state.

$$z(t+1) - z(t) = F(z(t+1))$$

  Assuming F is a contraction, the step may be computed by fixed-point iteration.

- Nesterov's algorithm responds to the future state that *would* occur if the naive algorithm was used.

$$z(t+1) - z(t) = F(z(t) + F(z(t)))$$

- We modify Nesterov's algorithm slightly, scaling up the predictor step:

$$z(t+1) - z(t) = F(z(t) + 2F(z(t))$$

Prior work shows convergence assuming ϕ_1, ϕ_2 are *strongly convex* with respect to dual norms. A function ϕ is strongly convex with respect to $\|\cdot\|$ if,

$$\phi\left(\frac{1}{2}(x+y)\right) \leq \frac{1}{2}\left(\phi(x) + \phi(y)\right) - \frac{1}{8}\|x-y\|^2$$

The existence of such regularizers depends strongly on the sets and norms. For the simplex, the negative entropy function has range $\log n$ and is strongly-convex with respect to $\|\cdot\|_1$. On the other hand, simple induction shows any ϕ strongly convex with respect to $\|\cdot\|_\infty$ on its unit ball has range at least $n/2$. This is the infamous ℓ_∞ barrier.

We overcome that barrier by *jointly* regularizing with a single function $\phi(q', p')$ satisfying a property we call *area-convexity*.

$$\phi\left(\frac{1}{3}(x+y+z)\right)) \leq \frac{1}{3}\left(\phi(x) + \phi(y) + \phi(z)\right) - \mathsf{Area}(\triangle xyz)$$

Here, area is defined with respect to the canonical 2-form,

$$\omega = \sum_i \mathrm{d}q_i \wedge \mathrm{d}p_i$$

We remark that our algorithm and the area convexity condition were initially obtained via an alternative perspective, that we now outline.

The various update rules may be viewed as numerical integration rules to evolve the system,

$$\frac{\mathrm{d}}{\mathrm{d}t}z = F(z(t))$$

Strong-convexity of the regularizer implies $z \mapsto F(z)$ is globally Lipschitz. Area-convexity implies the map is *locally* Lipschitz.

The dynamics may be alternatively defined via a single real-valued *Hamiltonian* function,

$$\mathcal{H}(q,p) = \max_{q' \in \mathcal{Q}, p' \in \mathcal{P}} q' \cdot p - p' \cdot q - \phi(q', p')$$

The dynamics are completely determined by $\mathcal{H}$ by *Hamilton's equations*,

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}q &= \frac{\mathrm{d}}{\mathrm{d}p}\mathcal{H}(q,p) \\ \frac{\mathrm{d}}{\mathrm{d}t}p &= -\frac{\mathrm{d}}{\mathrm{d}q}\mathcal{H}(q,p) \end{aligned}$$

By construction, $\frac{\mathrm{d}\mathcal{H}}{\mathrm{d}t} = 0$. Such systems are extensively studied in physics, engineering, and economics. In our context, $\mathcal{H}$ is the *cumulative* error, and q, p are the cumulative primal-dual pairs. In mechanics, $\mathcal{H}$ is Energy, q is position, and p is momentum. Flow problems on a graph $G = (V, E)$ with demands $b \in \mathbb{R}^V$ tie the two contexts together, as the Hamiltonian function corresponding to the continuous version of the algorithm encodes a physical model of a spring network that transports momentum. That is, the instantaneous flow $f'(e)$ through an edge e is interpreted as a force (i.e., flow rate of momentum.

Chapter 2

Minimum Norm Problems

Let $\mathcal{X}, \mathcal{Y}$ be finite dimensional vector spaces, where $\mathcal{X}$ is also a Banach space, and let $A \in \mathrm{Lin}(\mathcal{X}, \mathcal{Y})$ be fixed throughout this section. We consider the problem of finding a minimal norm pre-image of a specified b in the image of A; that is, finding $x \in \mathcal{X}$ with $Ax = b$ and $\|x\|_{\mathcal{X}}$ minimal.

Let x_{opt} be an exact solution, with $Ax_{\mathsf{opt}} = b$, and $\|x_{\mathsf{opt}}\|_{\mathcal{X}}$ minimal. There are multiple notions of approximate solutions for this problem. The most immediate is $x \in \mathcal{X}$ with $Ax = b$ and

$$\frac{\|x\|}{\|x_{\mathsf{opt}}\|} \leq \alpha \tag{2.1}$$

. We call such x an $(\alpha, 0)$-solution; we shall be interested in finding $(1 + \epsilon, 0)$-solutions for small ϵ. A weaker notion of approximation is obtained by further relaxing $Ax = b$ to $Ax \approxeq b$. Quantifying that requires more structure on $\mathcal{Y}$, so let us further assume $\mathcal{Y}$ to also be a Banach space. In that case, we say x is an (α, β)-solution if equation 2.1 holds and

$$\frac{\|Ax - b\|}{\|A\|\|x_{\mathsf{opt}}\|} \leq \beta \tag{2.2}$$

The practical utility of such solutions depends on the application. However, the weaker notion has the distinct advantage of being approachable by a larger family of algorithms, such as penalty and dual methods, by avoiding equality constraints. We discuss such existing algorithms in section 2.2, but mention that they typically yield $(1, \epsilon)$-solutions after some number of iterations. For ℓ_2, the iteration dependency on ϵ is $\mathcal{O}(\log(1/\epsilon))$, while for some more general norms it is $\epsilon^{-\mathcal{O}(1)}$.

2.1 Recursive Composition and Generalized Condition Numbers

We now consider how to trade an increase in α for a decrease in β. Residual recursion suggests a natural strategy: after finding an (α, β)-solution, recurse on $\tilde{b} = b - Ax$. More precisely, we define the *composition* of two algorithms as follows.

Definition 2.1.1. *Let $F_1, F_2 : \mathbb{R}^n \to \mathbb{R}^m$. The* composition $F_2 \diamond F_1$ *is defined by,*

$$(F_2 \diamond F_1)(b) = F_1(b) + F_2(b - AF_1(b))$$

Success of composition depends on $\|\tilde{b}\|$ being small implying $\tilde{b}$ has a small-norm pre-image. That is, it depends on the relation between $\|\tilde{b}\|$ and $\|\tilde{b}\|_{\mathsf{opt}}$. The extent to which that is true is quantified by the *condition number* of A. The condition number in ℓ_2 may be defined several ways, resulting in the same quantity. When generalized to arbitrary norms, those definitions differ. We recall two natural definitions, following Demko[20]. Our definitions differ slightly in not requiring A to be surjective.

Definition 2.1.2. *The* linear condition number *is defined by*

$$\kappa_{\mathcal{X}\to\mathcal{Y}}(A) = \min\{\|A\|\|\mathbf{G}\| : \mathbf{G} \in \mathrm{Lin}(\mathcal{Y}, \mathcal{X}), A\mathbf{G}A = A\}$$

The non-linear condition number *of $A : \mathcal{X} \to \mathcal{Y}$ is defined similarly, but does not require* $\mathbf{G}$ *to be linear. That is,*

$$\tilde{\kappa}_{\mathcal{X}\to\mathcal{Y}}(A) = \min\{\|A\|\|\mathbf{G}\| : \mathbf{G} : \mathcal{Y} \to \mathcal{X}, A \circ \mathbf{G} \circ A = A\}$$

where $\|\mathbf{G}\| = \sup\{\|\mathbf{G}b\|_X / \|b\|_{\mathcal{Y}} : b \in \mathcal{Y}\}$.

Of course, $\tilde{\kappa} \leq \kappa$. Having defined the condition number, we may now state how composition affects the approximation parameters.

Lemma 2.1.3 (Composition)**.** *Let F_i be an $(\alpha_i, \beta_i/\tilde{\kappa})$-algorithm for $A : \mathcal{X} \to \mathcal{Y}$, where A has non-linear condition number $\tilde{\kappa}$ Then, the composition $F_2 \circ F_1$ is an $(\alpha_1 + \alpha_2\beta_1, \beta_1\beta_2/\tilde{\kappa})$-algorithm for the same problem.*

Before proving lemma 2.1.3, we state two useful corollaries. The first concerns the result of recursively composing a $(\alpha, \beta/\tilde{\kappa})$-algorithm with itself.

Corollary 2.1.4. *Let F be a $(\alpha, \beta/\tilde{\kappa})$-algorithm for $\beta < 1$. Let F^t be the sequence formed by iterated composition, with $F^1 = F$, $F^{t+1} = F^t \circ F$. Then, F^t is a $(\alpha/(1-\beta), \beta^t/\tilde{\kappa})$-algorithm.*

Proof. By induction on t. To start, an $(\alpha, \beta/\tilde{\kappa})$-algorithm is trivially a $(\alpha/(1-\beta), \beta/\tilde{\kappa})$-algorithm. Assuming the claim holds for F^t, lemma 2.1.3 implies F^{t+1} is a $(\alpha + \alpha\beta/(1-\beta), \beta^{t+1}/\tilde{\kappa})$-algorithm. □

We observe that to obtain $(1 + \mathcal{O}(\epsilon), \delta)$-solution, only the first solver in the chain need be very accurate.

Corollary 2.1.5. *Let F be a $(1+\epsilon, \epsilon/2\tilde{\kappa}$-solver and G be a $(2, 1/2\tilde{\kappa})$-solver. Then, $G^t \circ F$ is a $(1 + 5\epsilon, \epsilon 2^{-t-1}/\tilde{\kappa})$-solver.*

For some problems, there is a very simple $(M, 0)$-solver known. A final composition with that solver serves to eliminate the error.

Corollary 2.1.6. *Let F be a $(1+\epsilon, \epsilon\delta/\tilde{\kappa})$-solver and G be a $(M, 0)$-solver. Then $G \circ F$ is a $(1 + \epsilon(1 + \delta M), 0)$-solver.*

2.1.1 Proof of lemma 2.1.3

Since F_1 is an $(\alpha_1, \beta_1/\tilde{\kappa})$-algorithm, we have

$$\begin{aligned} \|x\| &\leq \alpha_1 \|x_{\mathsf{opt}}\| \\ \frac{\|\tilde{b}\|}{\|A\|} &\leq \frac{\beta_1}{\tilde{\kappa}} \|x_{\mathsf{opt}}\| \end{aligned}$$

By the definition of $\tilde{\kappa}$,

$$\|\tilde{x}_{\mathsf{opt}}\| \leq \tilde{\kappa} \frac{\|\tilde{b}\|}{\|A\|} \leq \beta_1 \|x_{\mathsf{opt}}\|$$

Since F_2 is an $(\alpha_2, \beta_2/\tilde{\kappa})$-algorithm,

$$\begin{aligned} \|\tilde{x}\| &\leq \alpha_2 \|\tilde{x}_{\mathsf{opt}}\| \leq \alpha_2 \beta_1 \|x\|_{\mathsf{opt}} \\ \frac{\|A\tilde{x} - \tilde{b}\|}{\|A\|} &\leq \frac{\beta_2}{\tilde{\kappa}} \|\tilde{x}_{\mathsf{opt}}\| \leq \frac{\beta_1 \beta_2}{\tilde{\kappa}} \end{aligned}$$

The conclusion follows from $\|x + \tilde{x}\| \leq \|x\| + \|\tilde{x}\|$.

2.2 Example Solvers

The recursive composition technique takes existing (α, β)-approximation algorithms and yields new algorithms with different approximation parameters. In this section, we discuss the parameters of existing well-known base solvers.

Let $A : \mathbb{R}^m \to \mathbb{R}^n$ be linear and fixed throughout this section. To concisely describe results and ease comparison, for a norm $\|\cdot\|_{\mathcal{X}}$ on $\mathbb{R}^m$ and a norm $\|\cdot\|_{\mathcal{Y}}$ in $\mathbb{R}^n$, we write $(\alpha, \beta)_{\mathcal{X} \to \mathcal{Y}}$ to denote an (α, β) solution with respect to $A : \mathcal{X} \to \mathcal{Y}$. We quantify complexity in terms of *simple iterations*. A simple iteration consists of operations taking $\mathcal{O}(n+m)$ time, plus $O(1)$ applications of A and A^*.

Classical iterative methods for Euclidean problems provide a useful goalpost for comparison.

Theorem 2.2.1 ([15])**.**

The steepest-descent *algorithm produces a* $(1, \delta)_{2 \to 2}$*-solution after* $\mathcal{O}(\kappa_{2 \to 2}(A)^2 \log(1/\delta))$ *simple iterations*

The conjugate gradient *algorithm produces a* $(1, \delta)_{2 \to 2}$*-solution after* $\mathcal{O}(\kappa_{2 \to 2}(A) \log(1/\delta))$ *simple iterations.*

For non-Euclidean problems, existing nearly-linear time algorithms use the multiplicative weights method[42, 23], or more generally Nesterov's smoothing technique[37]. There are many existing algorithms for p-norm minimization. For lack of space, we simply state the results for ℓ_1 and ℓ_∞ versions sufficient for the max-flow and min-cost-flow results. We remark that these solvers may be obtained using either the more general framework of Nesterov[37], or with slightly worse parameters via the *multiplicative weights* method[23]. In the full version, we state results for a much larger family of norms.

Theorem 2.2.2 ([37]). *There is a* $(1,\delta)_{1\to 1}$*-solver using* $O(\log(m)\delta^{-2})$ *simple iterations.*

Theorem 2.2.3 ([37]). *There is a* $(1,\delta)_{\infty\to\infty}$*-solver using* $O(\log(n)\delta^{-2})$ *simple iterations.*

Chapter 3

Preconditioning Network Flow

In this chapter we discuss network flow problems. Let $\mathcal{G} = (V, E, c)$ be an undirected graph with n vertices and m edges, and edge weights $c \in \mathbb{R}^E$. While undirected, we assume the edges are oriented arbitrarily. A *single-commodity flow* $f \in \mathbb{R}^E$ specifies a quantity $f(e)$ to be transported along each edge $f(e)$. The *divergence* of a flow f at vertex x is the net quantity transported out of x by f. We write $D : \mathbb{R}^E \to \mathbb{R}^V$ for the divergence operator.

A *single-commodity flow problem* is specified by a *demands* $b \in \mathbb{R}^V$, and requires finding a flow $f \in \mathbb{R}^E$ satisfying $Df = b$, minimizing some cost function. We assume G connected and $\sum_x b(x) = 0$, so the problem is feasible.

The cost notion determines the flow problem. In the *maximum-flow* problem, the objective is to minimize the congestion

$$\max_e \frac{|f(e)|}{c(e)}$$

In the *transportation* problem, the objective is to minimize the total cost,

$$\sum_e c(e)|f(e)|$$

Let $C \in \mathbb{R}^{E \times E}$ be diagonal, with c along that diagonal. Then, max-flow and transportation respectively seek to minimize $\|C^{-1}f\|_\infty$ and $\|Cf\|_1$.

Our main results are nearly-linear time algorithms for those problems.

Theorem 3.0.4. *There are $(1+\epsilon)$-approximation algorithms for maximum flow and transportation in undirected graphs running in $m^{1+o(1)}\epsilon^{-2}$ time.*

To obtain fast algorithms, we construct preconditioners. For maximum-flow, we prove the following theorem in section 3.2.

Theorem 3.0.5. *There is an algorithm that given G, takes $m^{1+o(1)}$ time and outputs a data structure of size $n^{1+o(1)}$ that efficiently represents a preconditioner P with $\kappa_{\infty\to\infty}(PDC) \leq n^{o(1)}$. Given the data structure, P and P^* can be applied in $n^{1+o(1)}$ time.*

For transportation, we prove the following theorem in section 3.3.

Theorem 3.0.6. *There is a randomized algorithm that, given a length-graph $G = (V, E, c)$, takes $\mathcal{O}(m \log^2 n + n^{1+o(1)})$ time and outputs a $n^{1+o(1)} \times n$ matrix P such that every column of P has $n^{o(1)}$ non-zero entries and $\kappa_{1\to 1}(PDC^{-1}) \leq n^{o(1)}$.*

Having constructed the preconditioners, we use corollary 2.1.6 to chain together solvers. For maximum-flow, we use the theorem 2.2.3; for transportation, we use theorem 2.2.2. To terminate the chain, we need $(\alpha, 0)$ solvers with large alpha. For maximum-flow, routing through a maximum-capacity spanning tree yields a $(m, 0)$-solver. For transportation, routing through a minimum-cost spanning tree yields a $(n-1, 0)$-solver.

3.1 Related Work

Much of the early work on flow considers the general, directed edge case. Goldberg and Rao[24] solve directed maximum flow in $\tilde{O}(m \min(m^{1/2}, n^{2/3}) \log(\epsilon^{-1}))$ time. Ahuja, Goldberg, Orlin, and Tarjan[2] solve directed minimum-cost flow in $\tilde{\mathcal{O}}(nm \log(\epsilon^{-1}))$ time.

In a breakthrough, Spielman and Teng[47] solve Laplacian linear systems $\tilde{O}(m)$ time, initiating a flurry of new algorithmic results achieved by using the solver as a black box. Daitch and Spielman[18] show how to use the Laplacian solver with interior-point methods to obtain an $\tilde{\mathcal{O}}(m^{3/2} \log^2 \epsilon^{-1})$-time algorithm for minimum-cost flow. Christiano, Kelner, Madry, Spielman, and Teng approximate maximum flows in $\tilde{O}(m^{4/3}\epsilon^{-2})$ time[17]. Their approach uses the nearly linear-time Laplacian solver of Spielman and Teng[47] to take steps in the minimization of a softmax approximation of the edge congestions. Each step involves minimizing a weighted ℓ_2-norm of the congestions. While a naive analysis yields immediately yields a method that makes $\tilde{O}(\sqrt{m})$ such iterations (because $\|\cdot\|_2$ approximates $\|\cdot\|_\infty$ by a $\sqrt{m}$ factor), they present a surprising and insightful analysis showing in fact only $\tilde{O}(m^{1/3})$ such ℓ_2 iterations suffice. The maximum-flow-specific parts of [17] are quite simple, needing only to maintain the weights W and then using the Spielman-Teng solver as a black box. Hui Han, Madry, Miller, and Peng[16] adapt those ideas to approximate undirected transportation in $\tilde{O}(m^{4/3}\epsilon^{-2})$ time.

A second flurry of algorithmic results followed by opening the Laplacian solver box and leveraging its ideas more directly. Madry[33] approximates a large family of cut problems to within $n^{o(1)}$ in $m^{1+o(1)}$ time. The present author[44] and Kelner *et. al.*[26] independently approximate maximum flow in $m^{1+o(1)}\epsilon^{-2}$ time. Both approaches combine Madry's cut approximator with boosting via multiplicative weights. The main distinction is Kelner *et. al.* explicitly construct a $n^{o(1)}$-approximate *flow* algorithm (i.e., oblivious routing scheme) and then boost that, while we boost the cut algorithm directly. Generalizing and isolating the problem-specific boosting and approximation parts of our earlier work[44] leads to our general framework[45]. Our presentation in this chapter combines those two works.

3.2 Preconditioning Maximum Flow

In this section we prove theorem 3.0.5, using a construction of Madry[33], itself based on a construction of Spielman and Teng[47].

Definition 3.2.1 (Madry[33]). *A j-tree is a graph formed by the union of a forest with j components, together with a graph H on j vertices, one from each component. The graph H is called the core.*

Theorem 3.2.2 (Madry[33]). *For any graph G and $t \geq 1$, we can find in time $\tilde{O}(tm)$ a distribution of t graphs (λ_i, G_i) such that,*

- *Each G_i is a $O(m \log m/t)$-tree, with a core containing at most m edges.*
- *G_i dominates G on all cuts.*
- *$\sum_i \lambda_i G_i$ can be routed in G with congestion $\tilde{O}(\log n)$.*

We briefly remark that while the statement of theorem 3.2.2 in [33] contains an additional logarithmic dependence on the *capacity-ratio* of G, that dependence is easily eliminated[44]. Our construction will simply apply theorem 3.2.2 recursively, sparsifying the core on each iteration. To accomplish that, we use an algorithm of Benczúr and Karger[11].

Theorem 3.2.3 (Benczúr, Karger[11]). *There is an algorithm* `Sparsify`(G, ε) *that, given a graph G with m edges, takes $\tilde{O}(m)$ time and returns a graph G' with $m' = O(n\varepsilon^{-2} \log n)$ edges such that the capacity of cuts in the respective graphs satisfy*

$$G \leq G' \leq (1+\varepsilon)G$$

Further, the edges of G' are scaled versions of a subset of edges in G.

We now present the algorithm for computing the data structure representing a congestion-approximator. The algorithm `ComputeTrees` assumes its input is sparse; our top-level data-structure is constructed by invoking `ComputeTrees`(`Sparsify`$(G, 1), n^{1/k}$), where k is the parameter of theorem 3.0.5.

`ComputeTrees`(G, t):

- If $n = 1$, return.
- Using theorem 3.2.2, compute distribution $(\lambda_i, G_i)_i^{t'}$ of $\max(1, n/t)$-trees.
- Pick the t graphs of largest λ_i, throw away the rest, and scale the kept λ_i to sum to 1.
- For $i = 1, \ldots, t$:
  - $H_i' \leftarrow$ `Sparsify`$(H_i, 1)$, where H_i is the core of G_i
  - $L_i \leftarrow$ `ComputeTrees`(H_i', t)
- Return the list $L = (\lambda_i, F_i, L_i)_{i=1}^t$ where F_i is the forest of G_i.

The analysis of `ComputeTrees` correctness will make use of another algorithm for sampling trees. The `SampleTree` procedure is only used for analysis, and is not part of our flow algorithm.

`SampleTree`$(L = (\lambda_i, F_i, L_i)_i^t)$:

- Pick i with probability λ_i.
- Output $F_i + $ `SampleTree`(L_i)

Lemma 3.2.4. *Let G have $\tilde{O}(n)$ edges, and set $L \leftarrow$ `ComputeTrees`(G, t). Then, every tree in the sample space of* `SampleTree`(L) *dominates G on all cuts, and* $\mathbf{E}[$`SampleTree`$(L)]$ *is routable in G with congestion $\log(n)^{\log(n)/log(t)}$. Further, the computation of L takes $\tilde{O}(tn)$ time.*

Proof. By induction on n. For $n = 1$ the claim is vacuous, so suppose $n = t^{k+1}$. Since G has $O(n \log n)$ edges, the distribution output by theorem 3.2.2 will have $O(t \log^2 n)$ entries. We have $H_i' \geq H_i$ and $H_i + F_i \geq G$. Furthermore, the inductive hypothesis implies that every tree T_i in `SampleTree`(L_i) dominates H_i'. Then,

$$T_i + F_i \geq H_i' + F_i \geq H_i + F_i = G_i \geq G$$

Sparsifying the distribution from $O(t \log^2 n)$ to t scales λ_i by at most $O(\log^2 n)$, so that $\sum_{i=1}^t \lambda_i G_i$ is routable in G with congestion at most $\log^2 n$ larger than the original distribution. Since $H_i' \leq 2H_i$, by the multicommodity max-flow/min-cut theorem[32] H_i' is routable in H_i with congestion $O(\log n)$. By the inductive hypothesis, $\mathbf{E}[$`SampleTree`$(L_i)]$ is routable in H_i' with congestion $\log^{O(k)}(n)$. It follows then that $\mathbf{E}[$`SampleTree`$(L)]$ is routable in G with congestion at most $\log^{O(k+1)}(n)$.

Finally, `ComputeTrees` requires $\tilde{O}(tn)$ time to compute the distribution, another $\tilde{O}(tn)$ time to sparsify the cores, and then makes t recursive calls on sparse graphs with n/t vertices. It follows that the running time of `ComputeTrees` is $\tilde{O}(tn)$. □

Lemma 3.2.5. *Let R be the matrix that has a row for each forest edge in our data structure, and $(Rb)_e$ is the congestion on that edge when routing b. If* $\mathbf{E}[$`SampleTree`$(L)]$ *is routable in G with congestion κ, then $\kappa_{\infty\to\infty}(RDC) \leq \kappa$. Further, R has $\tilde{O}(tn)$ rows.*

Proof. Since the capacity of each tree-edge dominates the capacity of the corresponding cut in G, $\|b\|_{\mathsf{opt}} \geq \|Rb\|_\infty$. On the other hand, b can be routed in every tree with congestion $\|Rb\|_\infty$. By routing a $\mathbf{Pr}[T]$ fraction of the flow through tree T, we route b in $\mathbf{E}[$`SampleTree`$(L)]$ with congestion $\|Rb\|_\infty$. But then b can be routed in G while congesting by at most an κ factor larger.

The total number of edges in R satisfies the recurrence $E(n) \leq nt + tE(n/t)$ as each edge is either in one of the t toplevel forests, or in one of the t subgraphs. □

Having constructed our representation of R, it remains only to show how to multiply by R and $R^\top$. We use the following lemmas as subroutines, which are simple applications of leaf-elimination on trees.

Lemma 3.2.6. *There is an algorithm* `TreeFlow` *that, given a tree T and a demand vector b, takes $O(n)$ time and outputs for each tree edge, the flow along that edge when routing b in T.*

Lemma 3.2.7. *There is an algorithm* `TreePotential` *that, given a tree T annotated with a price p_e for each edge, takes $O(n)$ time and outputs a vector of vertex potentials v such that, for any i, j, the sum of the prices on the path from j to i in T is $v_i - v_j$.*

We begin with computing R. We take as input the demand vector b, and then annotate each forest edge e with the congestion r_e induced by routing b through a tree containing e.

`Compute`$R(b, L = (\lambda_i, F_i, T_i)_i^t)$:

- For $i = 1, \ldots, t$:
  - Let T be the tree formed by taking F_i, adding a new vertex s, and an edge from s to each core-vertex of F_i. Augment b with demand zero to the new vertex.
  - $f \leftarrow$ `TreeFlow`(b, T).
  - Set $r_e \leftarrow f_e/c_e$ for each forest edge in F_i.
  - Set b' to a vector indexed by core-vertices, with b'_j equal to the flow on the edge from s to core-vertex j.
  - `ComputeR`(b', T_i).

Lemma 3.2.8. *The procedure* `Compute`$R(b, L)$ *correctly annotates each edge e with $r_e = (Rb)_e$, and takes $\tilde{O}(tn)$ time.*

Proof. Let $L = (\lambda_i, F_i, T_i)_i^t$. We argue by induction on the depth of recursion. Fix a level and index i. Observe that the cut in G induced by cutting a forest edge is the same regardless of what tree T lies on the core: it is the cut that separates the part of F_i not containing the core from the rest of the vertices. It follows that we may place *any* tree on the core vertices, invoke `TreeFlow`, and obtain the flow on each forest edge. Next, for each component S of F_i, the total excess b_S must enter S via the core vertex. It follows that in a flow routing b on $F_i + T'$, for any tree T', the restriction of that flow to T' must have excess b_S on the core vertex of S, so it suffices to find a flow in the core with demands $b'_j = b_{S_j}$. But routing b in $F_i + T$ will place exactly b_{S_j} units of flow on the edge from s to core-vertex j.

The running time consists of t invocations of `TreeFlow` each taking $O(n)$ time, plus t recursive calls on graphs of size n/t, for a total running time of $\tilde{O}(tn)$. □

To compute $R^\top$, we assume each forest edge e has been annotated with a price p_e that must be paid by any flow per unit of congestion on that edge, and output potentials v such that $v_i - v_j$ is the total price to be paid for routing a unit of flow from j to i.

`Compute`$R^\top(L = (\lambda_i, F_i, T_i)_i^t)$:

- $v \leftarrow 0$
- For $i = 1, \ldots, t$:
  - $v' \leftarrow$ `Compute`$R^\top(T_i)$.
  - Let T be the tree formed by taking F_i, adding a vertex s, and an edge from s to each core-vertex of F_i. Set $q_e = p_e/c_e$ for each forest edge, and $q_e = v'_j$ for edge e from s to core-vertex j.
  - $v'' \leftarrow$ `TreePotential`(T, q)
  - Add v'' to v after removing the entry for s.
- Return v

Lemma 3.2.9. *Given edges annotated with per-congestion prices, the procedure* `Compute`$R^\top(L)$ *correctly returns potentials v such that $v_k - v_j$ is the cost per unit of flow from vertex j to k.*

Proof. Let $L = (\lambda_i, F_i, T_i)_i^t$. We argue by induction on the depth of recursion. Fix a level; a flow must pay its toll to each G_i, so the resulting potential equals the sum of the potentials for each i. Fix an index i. A unit of flow from j to k is first routed from j to the core-vertex of the component of F_i containing j, then to the core-vertex of the component containing k, and then finally to k. By induction, we assume that v' yields potentials that give the per-unit costs of routing between core-vertices. Placing a star on the core with the edge from s to core-vertex j having per-unit cost v'_j preserves those costs. If p_e is the price of an edge per unit of congestion, then $q_e = p_e/c_e$ is the price of an edge per unit of flow. It follows that the total toll paid is the same as the toll paid in T; thus, the potentials output by `TreePotential`(T, q) are correct.

The running time consists of t recursive calls to `Compute`$R^\top$ on graphs of size n/t, plus t invocations of `TreePotential` each taking $O(n)$ time, for a total running time of $\tilde{O}(tn)$. □

3.3 Preconditioning Min-Cost Flow

A cost function $c \in \mathbb{R}^E$ induces an intrinsic metric $\mathrm{d} : V \times V \to \mathbb{R}$, with distances determined by shortest (i.e., minimum cost) paths. We say a function $\phi \in \mathcal{V}$ is *L-Lipschitz* iff $|\phi(x) - \phi(y)| \leq L\mathrm{d}(x, y)$. The *value* of the minimum-cost flow depends only on the metric, not on the cost graph inducing that metric. Thus, we may forget about the costs c, and restrict our attention to the metric d.

Lemma 3.3.1. *If $\mathrm{d}(x, y) \leq \tilde{\mathrm{d}}(x, y)$ for all $x, y \in V$, then for all $b \in \mathbb{R}^V$,*

$$\|b\|_{\mathsf{opt}(\mathrm{d})} \leq \|b\|_{\mathsf{opt}(\tilde{\mathrm{d}})}$$

We recall that in some cases, the identity operator is a good preconditioner; for max-flow, such is the case in constant-degree expanders. For min-cost flow, the analagous case is a metric where distances between any pair of points differ relatively by small factors.

Definition 3.3.2. *Let $U \subseteq V$. We say U is* r-separated *if* $\mathrm{d}(x, y) \geq r$ *for all* $x, y \in U$ *with* $x \neq y$. *We say U is* R-bounded *if* $\mathrm{d}(x, y) \leq R$ *for all* $x, y \in U$.

Lemma 3.3.3. *Let $U \subset V$ be R-bounded and r-separated with respect to* d. *Let $b \in \mathbb{R}^V$ be demands supported on U. Then,*

$$\frac{r}{2}\|b\|_1 \leq \|b\|_{\mathsf{opt}(\mathrm{d})} \leq \frac{R}{2}\|b\|_1$$

Proof. Note $\frac{r}{2}\|b\|_1$ is exactly the min-cost of routing b in the metric where all distances are exactly r (consider the star graph with edge lengths $r/2$, with a leaf for each $x \in U$). The inequalities follow by monotonicity. □

A basic concept in metric spaces is that of an *embedding*: $\psi : V \to \tilde{V}$, with respective metrics $\mathrm{d}, \tilde{\mathrm{d}}$. An embedding has *distortion* L if for some $\mu > 0$,

$$1 \leq \mu \frac{\tilde{\mathrm{d}}(\psi(x), \psi(y))}{\mathrm{d}(x, y)} \leq L$$

Metric embeddings are quite useful for preconditioning; if d embeds into $\tilde{\mathrm{d}}$ with distortion L, then the monotonicity lemma implies the relative costs of flow problems differ by a factor L. It follows that if we can construct a preconditioner P for the min-cost flow problem on $\tilde{\mathrm{d}}$, the same P may be used to precondition min-cost flow on d, with condition number at most L-factor worse.

Our preconditioner uses embeddings in two ways. The preconditioner itself is based on a certain hierarchical routing scheme. Hierarchical routing schemes based on embeddings into tree-metrics have been studied extensively(see e.g. [21]), and immediately yield preconditioners with polylogarithmic condition number. However, we are unable to efficiently implement those schemes in nearly-linear time. Instead, we apply Bourgain's $\mathcal{O}(\log n)$-distortion embedding[14] into ℓ_1. That is, by paying an $\mathcal{O}(\log n)$ factor in condition number, we may completely restrict our attention to approximating min-cost flow in ℓ_1 space.

Theorem 3.3.4 (Bourgain's Embedding[14])**.** *Any n-point metric space embeds into ℓ_p space with distortion $\mathcal{O}(\log n)$. If* d *is the intrinsic metric of a length-graph $G = (V, E)$ with m edges, there is a randomized algorithm that takes $\mathcal{O}(m \log^2 n)$ time and w.h.p. outputs such an embedding, with dimension $\mathcal{O}(\log^2 n)$.*

The complexity of constructing and evaluating our preconditioner is *exponential* in the embedding's dimension. Therefore, we reduce the dimension to $\Theta(\sqrt{\log n})$, incuring another distortion factor of $\exp(\mathcal{O}(\sqrt{\log n}))$.

Theorem 3.3.5 (Johnson-Lindenstrauss[25, 19]). *Any n points in Euclidean space embed into k-dimensional Euclidean space with distortion $n^{\mathcal{O}(1/k)}$, for $k \leq \Omega(\log n)$. In particular, $k = \mathcal{O}(\log n)$ yields $\mathcal{O}(1)$ distortion, and $k = \mathcal{O}(\sqrt{\log n})$ yields $\exp(\mathcal{O}(\sqrt{\log n}))$ distortion.*

If the original points are in $d > k$ dimensions, there is a randomized algorithm that w.h.p outputs such an embedding in $\mathcal{O}(nd)$ time.

Our reduction consists of first embedding the vertices into ℓ_2 via theorem 3.3.4, reducing the dimension to $k = \mathcal{O}(\sqrt{\log n})$ using lemma 3.3.5, and then using the identity embedding of ℓ_2 into ℓ_1, for a total distortion of $\exp(O(\sqrt{\log n}))$.

To estimate routing costs in ℓ_1, we propose an extremely simple hierarchical routing algorithm in ℓ_1, and show it achieves a certain competetive ratio. The routing scheme applies to the exponentially large graph corresponding to a lattice discretization of the k-dimensional cube in ℓ_1; after describing such a scheme, we show that if the demands in that graph are supported on n vertices, the distributed routing algorithm finds no demand to route in most of the graph, and the same routing algorithm may be carried out in a $k-d$-tree instead of the entire lattice.

On the lattice, the routing algorithm consists of sequentially reducing the support set of the demands from a lattice with spacing 2^{-t-1} to a lattice with spacing 2^{-t}, until all demand is supported on the corners of a cube containing the original demand points. On each level, the demand on a vertex not appearing in the coarser level is eliminated by pushing its demand to a distribution over the aligned points nearby.

3.3.1 Lattice Algorithm

For $t \in \mathbb{Z}$, let $V_t = (2^{-t}\mathbb{Z})^k$ be the lattice with spacing 2^{-t}. We design a routing scheme and preconditioner for min-cost flow in ℓ_1, with initial input demands b_T supported on a bounded subset of V_T. For $t = T, T-1, \ldots$, the routing scheme recursively reduces a min-cost flow problem with demands supported on V_t to a min-cost flow problem with demands supported on the sparser lattice V_{t-1}. Given demands b_t supported on V_t, the scheme consists of each vertex $x \in V_t$ pulling its demand $b_t(x)$ uniformly from the closest points to x in V_{t-1} along any shortest path, the lengths of which are at most $k2^{-t}$. For example, a point $x = (x_1, \ldots, x_k) \in V_1$ with $j \leq k$ non-integer coordinates and demand $b_1(x)$ pulls $b_1(x)2^{-j}$ units of flow from each of the 2^j points in V_0 corresponding to rounding those j coordinates in any way. By construction, all points in $V_t \setminus V_{t-1}$ have their demands met exactly, and we are left with a reduced problem residual demands b_{t-1} supported on V_{t-1}. As we show, eventually the demands are supported entirely on the 2^k corners of a hypercube forming a bounding box of the original support set, at which point the the simple scheme of distributing the remaining demand uniformly to all corners is a k-factor approximation.

We do not actually carry out the routing; rather, we state a crude upper-bound on its cost, and then show that bound itself exceeds the true min-cost by a factor of some κ. For the preconditioner, we shall only require the sequence of reduced demands $b_t, b_{t-1}, \ldots$, and not the flow actually routing them.

Theorem 3.3.6.

$$\|b_t\|_{\mathsf{opt}} \leq \sum_{s=0}^{t} k2^{-s}\|b_s\|_1 \leq 2k(t+1)\|b_t\|_{\mathsf{opt}}$$

Assuming theorem 3.3.6 holds, we take our preconditioner to be a matrix P such that,

$$\|Pb_T\|_1 = \sum_{t=0}^{T} k2^{-t}\|b_t\|_1$$

The proof of theorem 3.3.6 uses two essential properties of the routing scheme described. The first is that the cost incurred in the reduction from b_t to b_{t-1} is not much larger than the min-cost of routing b_t. The second is the reduction does not increase costs. That is, the true min-cost for routing the reduced problem b_{t-1} is at most that of routing the original demands b_t.

Lemma 3.3.7. *Let b_{t-1} be the reduced demands produced when the routing scheme is given b_t. Then,*

1. *The cost incurred by that level of the scheme is at most $k2^{-t}\|b_t\|_1 \leq 2k\|b_t\|_{\mathsf{opt}}$*
2. *The reduction is non-increasing with respect to min-cost: $\|b_{t-1}\|_{\mathsf{opt}} \leq \|b_t\|_{\mathsf{opt}}$*

Assuming lemma 3.3.7, we now prove 3.3.6. We argue the sum is an upper bound on the total cost incurred by the routing scheme, when applied to b_t. The left inequality follows immediately, as the true min-cost is no larger.

The total cost of the scheme is at most the cost incurred on each level, plus the cost of the final routing of b_0. Lemma 3.3.7(a) accounts for all terms above $s = 0$. For the final routing, the demands b_0 are supported in $\{0,1\}^k$, so the cost of routing all demand to a random corner is at most $\frac{1}{2}k\|b_0\|_1$.

For the right inequality, we observe

$$\sum_{s=0}^{t} k2^{-t}\|b_t\|_1 \leq \sum_{s=0}^{t} 2k\|b_s\|_{\mathsf{opt}}$$

The inequality follows termwise, using lemma 3.3.7(a) for $s \geq 1$; the $s = 0$ case follows from lemma 3.3.3 because V_0 is 1-separated.

3.3.2 Proof of Lemma 3.3.7

For the first part, each vertex $x \in V_t$ routes its demand $b_t(x)$ to the closest points in V_{t-1} any shortest path. As such paths have length at most $k2^{-t}$, each point x contributes at most $k2^{-t}|b_t(x)|$.

For the second part, by scale-invariance it suffices to consider $t = 1$. Moreover, it suffices to consider the case where b_1 consists of a unit demand from between two points $x, y \in V_1$ with $\|x - y\|_1 = \frac{1}{2}$. To see this, we observe any flow routing arbitrary demands b_1 consists

of a sum of such single-edge flows. As the reduction is linear, if the cost of each single-edge demand-pair is not increased, then the cost their sum is not increased. By symmetry, it suffices to consider x, y with $x = (0, z)$ and $y = (\frac{1}{2}, z)$. Then, the reduction distributes x's demand to a distribution over $(0, z')$ where $z' \sim Z'$; y's demand is split over $(0, z')$ and $(1, z')$ where $z' \sim Z'$. Therefore, half of the demand at $(0, z')$ is cancelled, leaving the residual problem of routing $1/2$ unit from $(0, z')$ to $(1, z')$. That problem has cost $1/2$, by routing each fraction directly from $(0, z')$ to $(1, z')$.

Chapter 4

Area-Convexity and Multicommodity Flow

4.1 Introduction

Biaffine saddle point problems are fundamental in optimization, generalizing linear programming in a way that explicitly introduces duality. The problem is to compute

$$\min_{x \in \mathcal{X}} \max_{y \in \mathcal{Y}} \mathfrak{A}(x, y) \tag{4.1}$$

where $\mathcal{X}, \mathcal{Y}$ are compact-convex subsets of real finite dimensional vector spaces, and $\mathfrak{A}$ is biaffine. The problem of deciding if there is an $x \in \mathcal{X}$ satisfying a linear system $Ax = b$ reduces to (4.1) by taking $\mathfrak{A}(x, y) = y \cdot (Ax - b)$, and choosing $\mathcal{Y}$ according to a desired measure of error. In single-commodity network flow problems on a graph $G = (V, E)$, $\mathcal{X} \subset \mathbb{R}^E$ represents feasible flows, while $Ax = b \in \mathbb{R}^V$ specifies the outflux demands at each vertex; in k-commodity problems, x, b are expanded to k-column matrices X, B, and the outflux constraints become $AX = B$.

Conjugate regularization is one of the most powerful existing tools for solving the general problem (4.1). The technique, implicitly used by the multiplicative weights method[42, 23] for the unit simplex case, is developed more generally by Nesterov[37]. A non-differentiable objective function

$$\eta(x) = \max_{y \in \mathcal{Y}} \mathfrak{A}(x, y)$$

is approximated by choosing a strictly convex function $\phi : \mathcal{Y} \to \mathbb{R}$, and considering

$$\tilde{\eta}(x) = \max_{y \in \mathcal{Y}} \mathfrak{A}(x, y) - \phi(y)$$

Strict convexity of ϕ ensures $\tilde{\eta}$ is differentiable, allowing first-order optimization methods (e.g. steepest descent) to be applied. For first-order methods to work well, the first-order approximation to $\tilde{\eta}$ should be a good one; that is, $\tilde{\eta}$ should be smooth. Nesterov[37] shows that *strong convexity* of ϕ implies a corresponding smoothness property of $\tilde{\eta}$. Roughly, ϕ is strongly-convex if for any two points x, y, $\frac{1}{2}(\phi(x) + \phi(y))$ exceeds $\phi(\frac{1}{2}(x + y))$ by an amount

proportional to $\|x-y\|^2$. Nesterov[38] presents two different algorithms. The first, we call *half-regularized*, regularizes only one of the two sets, and takes roughly $|\phi|\epsilon^{-2}$ iterations, where $|\phi|$ denotes the size of ϕ's range. The second, we call *fully regularized*, uses *two* regularizers ϕ_X, ϕ_Y, one for each set $\mathcal{X}, \mathcal{Y}$, and takes roughly $\sqrt{|\phi_X||\phi_Y|}\epsilon^{-1}$ iterations.

Therefore, designing fast algorithms using Nesterov's framework requires choosing ϕ to carefully balance two requirements. On the one hand, ϕ should be small so that $\eta \approxeq \tilde{\eta}$. On the other hand, it should be strongly convex for first-order techniques to work well. Unfortunately, for some sets, most notably ℓ_∞ balls, those two properties are irreconcilable; any ϕ that is strongly-convex with respect to $\|\cdot\|_\infty$ on $[-1,1]^n$ has $|\phi| \geq \frac{n}{2}$. The latter ℓ_∞ barrier is responsible for stalled progress on algorithms for several important optimization problems. For single-commodity flow, the barrier prevents use of fully-regularized methods with faster ϵ^{-1} dependance. For multicommodity flow, the consequences are even more severe. In that case, $\mathcal{X}$ is the set of $m \times k$ right-stochastic matrices (i.e., m simplices), and $\mathcal{Y}$ is a corresponding polar. There, $\mathcal{X}$ contains a m-dimensional ℓ_∞ ball as before, but $\mathcal{Y}$ also includes a k-dimensional ℓ_∞ ball, so nontrivial approximation requires $\Omega(\min(m,k))$ iterations. The fastest current algorithm for undirected multicommodity flow, due to Kelner *et al.*[26], requires $\Omega(mk^2)$ time for m edges and k commodities, with one factor of k attributable to the ℓ_∞ barrier.

In this paper, we show how to break that barrier by using regularizers satisfying a dramatically weaker property we call *area-convexity*. Roughly, ϕ is area-convex if for any *three points* x, y, z, $\frac{1}{3}(\phi(x)+\phi(y)+\phi(z))$ exceeds $\phi(\frac{1}{3}(x+y+z))$ by an amount proportional to the area of the triangle defined by the convex hull of x, y, z. Further, the notion of area used is *intrinsic* to the problem, obtained by first reducing (4.1) to a case where $\mathfrak{A}$ is antisymmetric, and then interpreting it as a two-form. In 4.1.1, we define area-convexity more precisely and present a modified version of Nesterov's algorithm[38] that requires only area-convexity for convergence. We also show that for twice-differentiable ϕ, area-convexity is approximately equivalent to purely local property of ϕ's Hessian, much like the case of regular convexity.

Area-convexity is weaker than strong-convexity; three points on a common line have zero area, so it does not even imply strict convexity. Roughly, area-convexity permits ϕ to be only slightly convex along a line u, if it is very convex along lines v where the uv plane has large area. To exploit that, we *jointly* regularize $\mathcal{X} \oplus \mathcal{Y}$ with a single function $\phi(x,y)$ that *does not* separate into $\phi_1(x) + \phi_2(y)$.

The "outer" iteration steps taken by many different algorithms[29, 37, 36, 38], including our own, are quite similar. In 4.2.3, we observe that when the step size is taken to zero, they all reduce to a single common *continuous* algorithm that integrates a simple *Hamiltonian System*. Each algorithm effectively uses a different numerical integration rule to discretize the system. The weaker properties of our regularizers, namely non-separability and area-convexity, translate to continuous algorithms belonging to more general classes of such dynamical systems. In particular, joint regularization yields non-separable Hamiltonian systems.

Having developed the more general framework, we obtain faster algorithms for fundamental optimization problems by constructing small area-convex regularizers for sets where small strongly-convex regularizers do not exist. Our main technical result, stated in 4.1.2,

is a nearly-linear time algorithm for approximately solving inequalities $AX \leq B$ over right-stochastic matrices X; if such an X exists, the algorithm outputs $\tilde{X}$ with $A\tilde{X} \leq B + \epsilon\|A\|R$ for some right-stochastic R. Several important problems including ℓ_1/ℓ_∞-regression[16], optimal transportation[45], and maximum concurrent flow easily reduce to the stochastic matrix problem, so we obtain faster algorithms for those problems. In particular, by using existing work on preconditioning single-commodity maximum flow[44, 26, 45, 41], we immediately obtain a nearly linear time approximation algorithm for maximum concurrent flow.

Applying general optimization algorithms to network flow problems often results in numerical update rules that have intuitive interpretations as simulating physical models of flow networks (e.g. mass-spring or capacitor-inductor systems), and our algorithm is no exception. Combining that interpretation with the continuous Hamiltonian view allows the area-convex regularizer to be understood as a dynamically self-tuning network element. Roughly, to have a discrete step of time t approximate continuous evolution for time t, the system's oscillation modes should have frequency $\mathcal{O}(1/t)$. The separable Hamiltonian systems resulting from separate regularization of primal and dual spaces yield network models containing separate elements (e.g, springs, masses) for the primal (position) and dual (momentum) parts of the phase space, with total energy being the sum of both. In contrast, our system consists of joint elements that dynamically tune their impedances with respect to both parts *jointly* to guarantee low frequency oscillation. We remark our algorithmic results were actually obtained via that approach, namely attempting to design physical models of systems that solve flow problems and have bounded frequency response. We have reformulated it via regularization for sake of familiarity to optimization researchers.

4.1.1 Area Convexity

We consider a general bi-affine saddle point problem of the form

$$\min_{x \in \mathcal{X}} \max_{y \in \mathcal{Y}} \mathfrak{A}(x, y) \tag{4.2}$$

where

$$\mathfrak{A}(x, y) = y \cdot Ax - b \cdot y - c \cdot x$$

Here, $\mathcal{X}, \mathcal{Y}$ are compact-convex in real finite dimensional vector spaces, A is a linear operator, and b, c are linear functionals. Throughout this paper, boldface is used to denote problem inputs or other objects that remain fixed throughout the course of an algorithm, while non-boldface is used for outputs or other objects that vary. A pair $(x, y) \in \mathcal{X} \oplus \mathcal{Y}$ is said to be ϵ-*optimal* for (4.2) iff,

$$\epsilon \geq \max_{x' \in \mathcal{X}, y' \in \mathcal{Y}} \mathfrak{A}(x, y') - \mathfrak{A}(x', y) \tag{4.3}$$

Our goal in approximately solving 4.2 is to find an ϵ-optimal pair for small ϵ.

We begin by applying a standard but useful reduction of (4.2) to a purely bilinear, *self-dual* form that will be more concise and convenient. By increasing the dimension by one

and augmenting $\mathcal{X}, \mathcal{Y}$ with a constant coordinate, we may move the purely linear terms b, c into A, leaving the purely bilinear problem

$$\min_{x \in \mathcal{X}} \max_{y \in \mathcal{Y}} y \cdot Ax \tag{4.4}$$

In a self-dual problem, $\mathcal{X} = \mathcal{Y}$ and A is *alternating* (i.e., $A^* = -A$). We reduce (4.4) to self-dual form by setting $\mathcal{C} := \mathcal{X} \oplus \mathcal{Y}$ and $J(x, y) = (A^* y, -Ax)$, leaving

$$\min_{z \in \mathcal{C}} \max_{z' \in \mathcal{C}} z' \cdot Jz \tag{4.5}$$

The *value* of a self-dual problem is zero. Having reduced (4.2) to (4.5), we now take (4.5) to be the general problem under consideration. That is, for a real finite dimensional linear space $\mathbb{Z}$, we are given an alternating linear operator $J : \mathbb{Z} \to \mathbb{Z}^*$ together with compact-convex $\mathcal{C} \subset \mathbb{Z}$, and we seek approximate solutions to (4.5).

We assume the set $\mathcal{C}$ to be represented by an oracle minimizing a regularizer ϕ. Note that even if $\mathcal{C} = \mathcal{X} \oplus \mathcal{Y}$, such a function may generally depend jointly on both blocks.

Definition 4.1.1. *A δ-approximate minimization oracle(δ-AMO) for $\phi : \mathcal{C} \to \mathbb{R}$ takes input $a \in \mathbb{Z}^*$, and outputs $z \in \mathcal{C}$ such that,*

$$a \cdot z - \phi(z) + \delta \geq \sup_{z' \in \mathcal{C}} a \cdot z' - \phi(z') =: \phi^*(a)$$

We assume ϕ has an additional property that we call *area convexity*. A function ϕ is area-convex if for any three points x, y, z, the value of ϕ at their mean lies below the mean of their values by an amount proportional to the area of the triangle defined by their convex hull. We view J as a two-form to intrinsically define the oriented area of such a triangle as $\frac{1}{2}(z - y) \cdot J(x - y)$.

Definition 4.1.2. *ϕ is area-convex with respect to J on a convex set $\mathcal{C}$, iff, for all $x, y, z \in \mathcal{C}$,*

$$\phi\left(\frac{x + y + z}{3}\right) \leq \frac{1}{3}\left(\phi(x) + \phi(y) + \phi(z)\right) - \frac{1}{3\sqrt{3}}(z - y)J(y - x)$$

Our main general result is that area-convex regularization suffices for fast saddle-point algorithms.

Theorem 4.1.3. *Let J be alternating, $\mathcal{C}$ compact-convex, and $\phi : \mathcal{C} \to [-\rho, 0]$. Suppose ϕ is area-convex with respect to $2\sqrt{3}J$ on $\mathcal{C}$, and Φ is a δ-AMO for ϕ. Define a sequence by $z(0) = 0$,*

$$z(t + 1) = z(t) + \tilde{\Phi}(Jz(t))$$

where

$$\tilde{\Phi}(a) = \Phi(a + 2J\Phi(a))$$

Then, for all $t > 0$, we have $\frac{1}{t} z(t) \in \mathcal{C}$ and,

$$\delta + \rho/t \geq \max_{z' \in \mathcal{C}} z' \cdot J(z(t)/t)$$

In particular, for $t = \rho\epsilon^{-1}$, a $\delta + \epsilon$-approximate solution is obtained in $\rho\epsilon^{-1}$ iterations, each requiring $\mathcal{O}(1)$ calls to Φ, $\mathcal{O}(1)$ applications of J, and $\mathcal{O}(1)$ vector additions.

We prove theorem 4.1.3 in 4.2.1, remarking that both the sequence defined and the (quite short) proof of its convergence closely follows Nesterov[38].

The definition of area-convexity is simplest to state and *use*. However, it is not the simplest to satisfy. Like convexity, we may alternatively characterize the notion *locally* for e.g. twice-differentiable ϕ; first, we must consider quadratic ϕ. We recall an operator Q is *positive semidefinite*(PSD), and write $Q \succeq 0$, if $z \cdot Qz \geq 0$ for all z; we write $Q \succeq \mathbf{P}$ to denote $Q - \mathbf{P} \succeq 0$.

We state two simple but useful lemmas about quadratic functions. Proofs follow directly from the definition of area-convexity; we include them in appendix 4.5. We also remark the constant in definition 4.1.1 is chosen to eliminate constants from lemma 4.1.4.

Lemma 4.1.4. *The function $z \mapsto \frac{1}{2} z \cdot Qz$ is area-convex with respect to J if and only if,*

$$\begin{bmatrix} Q & -J \\ J & Q \end{bmatrix} \succeq 0 \tag{4.6}$$

We use the condition (4.6) of lemma 4.1.4 frequently enough to introduce the more concise notation

$$Q \succeq \imath J : \iff \begin{bmatrix} Q & -J \\ J & Q \end{bmatrix} \succeq 0 \tag{4.7}$$

We work only with real vector spaces, so the notation is unambiguous, but we remark the notation is consistent with and motivated by the complex operator $Q - \imath J$ being PSD.

Lemma 4.1.5. *Let $Q \in \mathbb{R}^{2\times 2}$ be symmetric. Then,*

$$Q \succeq \imath \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

if and only if $Q \succeq 0$ and $\det(Q) \geq 1$

If ϕ is twice-differentiable on a set $\mathcal{K}$, we let $\mathrm{d}\phi : \mathcal{K} \to \mathbb{Z}^*$ denote the derivative, and $\mathrm{d}^2\phi : \mathcal{K} \to (\mathbb{Z} \to \mathbb{Z}^*)$ denote the Hessian. We may approximately characterize area-convexity locally.

Theorem 4.1.6. *Let ϕ be twice differentiable on a convex set $\mathcal{K}$.*

1. *If ϕ is area-convex w.r.t J on the interior of $\mathcal{K}$, then $\mathrm{d}^2\phi(z) \succeq \imath J$ for all z in the interior of $\mathcal{K}$.*
2. *If $\mathrm{d}^2\phi(z) \succeq \imath J$ for all $z \in \mathcal{K}$, then ϕ is area-convex w.r.t $\frac{1}{3}J$ on $\mathcal{K}$.*

   If, further, ϕ is continuous on the closure of $\mathcal{K}$, then ϕ is area-convex w.r.t $\frac{1}{3}J$ on the closure of $\mathcal{K}$.

We prove theorem 4.1.6 in 4.2.2; we remark that while the factor of three is not tight, the factor is not one either.

4.1.2 Right-Stochastic Matrix Problems

We consider saddle point problems between right-stochastic matrices $\triangle_k^m$ and a corresponding polar $(\triangle_k^n)^\oplus$.

$$\begin{aligned}\triangle_k^m &= \{X \in \mathbb{R}^{m\times k} : X \geq 0, \forall e \sum_j X_{ej} = 1\} \\ (\triangle_k^m)^\oplus &= \{Y \in \mathbb{R}^{m\times k} : Y \geq 0, \sum_e \max_j Y_{ej} \leq 1\}\end{aligned}$$

Note we identify $X \in (\mathbb{R}^k)^m$ with $X \in \mathbb{R}^{m\times k}$, so that $X_e \in \mathbb{R}^k$ is the e^{th} row of X and $(X_{ej}) = (X_e)_j$.

Our main technical result is a nearly-linear time algorithm for solving saddle point problems between such sets.

Theorem 4.1.7. *There is an algorithm that given $B \in \mathbb{R}^{n\times k}, C \in \mathbb{R}^{m\times k}$, and $A \in \mathbb{R}^{n\times m}$ with $\|A\|_{\infty\to\infty} \leq 1$, takes $\tilde{\mathcal{O}}(k\mathsf{nnz}(A)\epsilon^{-1})$ time and outputs $X \in \triangle_k^m, Y \in (\triangle_k^n)^\oplus$ such that,*

$$\epsilon \geq \max_{X'\in\triangle_k^m, Y'\in(\triangle_k^n)^\oplus} \mathfrak{A}(Y', X) - \mathfrak{A}(Y, X')$$

where,

$$\mathfrak{A}(Y, X) = \mathrm{tr}[Y^*AX] - \mathrm{tr}[C^*X] - \mathrm{tr}[B^*Y]$$

Here, $\mathsf{nnz}(A)$ is the number of nonzero entries in A; we assume A has no all-zero rows or columns, so $\mathsf{nnz}(A) \geq \max m, n$. We prove theorem 4.1.7 in section 4.3, by constructing area-convex regularizers for $\triangle_k^m \oplus (\triangle_k^n)^\oplus$. We remark that using existing strongly-convex regularization frameworks, a half-regularized approach yields $\tilde{\mathcal{O}}(\min(k, m)\ k\mathsf{nnz}(A)\epsilon^{-2})$ time while the fully-regularized approach yields $\tilde{\mathcal{O}}(\sqrt{mk}k\mathsf{nnz}(A)\epsilon^{-1})$ time.

Theorem 4.1.7 is useful for approximately solving inequality systems over right-stochastic matrices; we state an immediate corollary.

Corollary 4.1.8. *There is an algorithm that given $B \in \mathbb{R}^{n\times k}$, and $A \in \mathbb{R}^{n\times m}$ with $\|A\|_{\infty\to\infty} \leq 1$, takes $\tilde{\mathcal{O}}(k\mathsf{nnz}(A)\epsilon^{-1})$ time and outputs either,*

1. *$X \in \triangle_k^m$ such that $AX \leq B + \epsilon R$ where $R \in \triangle_k^n$.*
2. *$Y \geq 0$ such that $\mathrm{tr}[Y(AX - B)] > 0$ for all $X \in \triangle_m^k$.*

4.1.3 Multicommodity Flow

Let $G = (V, E)$ be *directed* graph, and let $D \in \mathbb{R}^{V\times E}$ be the *discrete divergence* operator for G. A single-commodity flow $f \in \mathbb{R}^E$ specifies a quantity $f_e \geq 0$ be transported along edge e, while $(Df)_v$ is the net quantity transported out of vertex v. A k-commodity flow $F \in (\mathbb{R}^k)^E$ specifies quantities $F_{ej} \geq 0$ for each commodity j and edge e. Then, $(DF)_{vj}$ is the net quantity of commodity j transported out of v by F; if $DF = B$, we say F *routes* B. A flow F is said to be *feasible* with respect to *capacities* $c \in \mathbb{R}^E$ if $c_e \geq \sum_j F_{ej}$ for each

edge e. Letting $C \in \mathbb{R}^{E \times E}$ be diagonal with $C_{ee} = c_e$, a feasible flow F may be written as $F = CX$ for right-*sub*-stochastic X.

In the *maximum-concurrent flow* problem, we are given $G = (V, E)$ with capacities $c \in \mathbb{R}^E$, demands $B \in (\mathbb{R}^k)^V$, and we must feasibly route λB for $\lambda > 0$ as large as possible. If λ_{opt} is the true such maximum, then a flow F is a $(1-\epsilon)$-approximation to maximum-concurrent flow if it is feasible and $DF = \lambda B$ where $\lambda \geq (1-\epsilon)\lambda_{\mathsf{opt}}$.

In an undirected graph, there are actually two notions of feasibility, corresponding to whether edges are *half-duplex* (i.e., share capacity in each direction) or *full-duplex* (i.e., have full capacity in each direction). The full-duplex version represents an undirected edge e of capacity c_e as two directed edges of opposite orientation, each of capacity c_e. In the half-duplex case, the condition $F \geq 0$ is dropped, the sign of F_{ej} specifies direction of flow, and feasibility becomes

$$c_e \geq \sum_j |F_{ej}|$$

We distinguish the two because existing work on the undirected case we are aware of[27, 26] uses the half-duplex definition, while the full-duplex variant is more clearly related to the stochastic matrix problem.

Our main result concerning multicommodity flow is that the undirected variants may be approximately solved in nearly-linear time. The algorithms also output dual solutions.

Theorem 4.1.9. *There are $(1-\epsilon)$-approximation algorithms for both duplex variants of undirected maximum concurrent flow that run in $\tilde{\mathcal{O}}(mk\epsilon^{-1})$ time. Both algorithms also output dual solutions certifying $(1-\epsilon)$-optimality.*

We sketch the algorithm in section 4.4, but remark no new flow-specific work is actually required; theorem 4.1.9 follows by substituting the algorithm of theorem 4.1.7 for the alternatives used by existing work[26, 44, 45] in a straightforward way.

Those works leverage ideas introduced by Spielman and Teng[47] to solve Laplacian systems in nearly-linear time. Kelner, Miller, and Peng[27] present an algorithm requiring $\tilde{\mathcal{O}}(m^{4/3}\mathrm{poly}(k, \epsilon^{-1}))$ time using the Laplacian Solver as a black-box. Kelner *et al.*[26] reduce that to $m^{1+o(1)}k^2\epsilon^{-2}$ time. In that bound, one factor of k comes from the need for at least k iterations due to the infamous ℓ_∞ regularization barrier.

4.2 Area Convexity

In this section, we prove the claims stated in 4.1.1, and also observe the saddle point algorithm may be viewed as discretizing a continuous algorithm that solves the problem. The analysis we present closely follows a simplified version of Nesterov's[38] dual extrapolation method for variational inequalities, as applied to the case of self-dual bilinear saddle point problems.

4.2.1 Proof of Theorem 4.1.3

Define,

$$\eta(a) = \sup_{z' \in \mathcal{C}} z'a$$

Our goal is to find $z \in \mathcal{C}$ with $\eta(Jz) \leq \epsilon$. Recall,

$$\phi^*(a) = \sup_{z' \in \mathcal{C}} z'a - \phi(z')$$

Since $\phi \leq 0$, we observe,

$$\eta(a) \leq \phi^*(a)$$

Additionally, because η is positive homogeneous (i.e, $\eta(ta) = t\eta(a)$), we actually have the following lemma.

Lemma 4.2.1. *For $t > 0$, and $a \in \mathbb{Z}^*$,*

$$\eta\left(\frac{a}{t}\right) \leq \frac{\phi^*(a)}{t}$$

The proof of theorem 4.1.3 proceeds by observing the sequence has the property that $z(t) \in t\mathcal{C}$, yet $\phi^*(Jz(t))$ does not increase by much. The latter is claimed by the following lemma.

Lemma 4.2.2 (Stepping Lemma)**.** *Let $J, \phi, \tilde{\Phi}, \delta$ be as in theorem 4.1.3. Then, for any a,*

$$\phi^*(a + J\tilde{\Phi}(a)) \leq \phi^*(a) + \delta$$

Before proving the stepping lemma, let us observe it proves the theorem. First, $\phi^*(Jz(0)) = \phi^*(0) \leq \rho$. By the stepping lemma, we have,

$$\phi^*(Jz(t+1)) = \phi^*(Jz(t) + J\tilde{\Phi}(Jz(t))) \leq \phi^*(Jz(t)) + \delta$$

By induction, $\phi^*(Jz(t)) \leq \rho + \delta t$. By lemma 4.2.1, $\eta(J(z(t)/t)) \leq \delta + \rho/t$.

We now prove the stepping lemma. Without loss of generality, we may assume $a = 0$, or else consider $\hat{\phi}(x) = \phi(x) - a \cdot x$. Let $x = \Phi(0)$, and $y = \tilde{\Phi}(0) = \Phi(2Jx)$. Let $z \in \mathcal{C}$ be arbitrary. Our goal is to show,

$$z \cdot Jy - \phi(z) \leq \phi^*(0) + \delta$$

We have three points $x, y, z \in \mathcal{C}$. Since ϕ is area-convex w.r.t $2\sqrt{3}J$ on $\mathcal{C}$,

$$(z-y) \cdot J(y-x) \quad \leq \quad \frac{1}{2}(\phi(z) + \phi(y) + \phi(x) - 3\phi((x+y+z)/3))$$

Certainly, $-\phi((x+y+z)/3) \leq \phi^*(0)$, while our assumption of Φ is that $-\phi(x) \geq \phi^*(0) - \delta$, so we have,

$$(z-y) \cdot J(y-x) \leq \phi^*(0) + \frac{1}{2}(\phi(z) + \phi(y) + \delta) \tag{4.8}$$

Applied to y, our assumption of Φ implies,

$$2(z-y)\cdot Jx - \phi(z) + \phi(y) \le \delta$$

Equivalently,

$$(z-y)\cdot Jx - \phi(z) \le \frac{1}{2}(\delta - \phi(y) - \phi(z)) \tag{4.9}$$

Combining (4.8) and (4.9) yields,

$$z\cdot Jy - \phi(z) = (z-y)\cdot J(y-x) + (z-y)\cdot Jx - \phi(z) \le \phi^*(0) + \delta$$

4.2.2 Proof of Theorem 4.1.6

Without loss of generality let 0 be in the interior of $\mathcal{C}$, and $\mathrm{d}^2\phi(0) = Q$. Let $x, y \in \mathbb{Z}$. Then, for sufficiently small t, by area-convexity,

$$\frac{1}{3t^2}(\phi(tx) + \phi(ty) + \phi(-ty - tx)) - \phi(0)) \ge \frac{1}{\sqrt{3}} y \cdot Jx$$

Taking the limit of the left side, the function $\frac{1}{2}x\cdot Qx$ must be area-convex, and the first direction follows by lemma 4.1.4.

For the other direction, we begin with a convenient lemma.

Lemma 4.2.3. *Let $\mathcal{C}$ be an equilateral triangle of unit height in $\mathbb{R}^2$, and suppose $\psi : \mathcal{C} \to (-\infty, 0]$ is convex with $\det(\mathrm{d}^2\psi) \ge 1$ on the interior of $\mathcal{C}$. Then, $\psi(x) \le -\frac{2}{27}$ where x is the midpoint of the triangle.*

Assuming lemma 4.2.3, we prove theorem 4.1.6. Suppose $\mathrm{d}^2\phi \succeq \imath J$ on $\mathcal{K}$, and let $u, v, w \in \mathcal{K}$.

If $(u-v)\cdot J(v-w) = 0$, the result is trivial, as $\mathrm{d}^2\phi \succeq \imath J$ implies $\mathrm{d}^2\phi \succeq 0$. Otherwise, without loss of generality we assume $u = -v$ and $\phi(u) = \phi(v) = \phi(w) = 0$; the former may be assumed by translating $\mathcal{K}$, and the latter by observing

$$\frac{1}{3}\left(\phi(u) + \phi(v) + \phi(w)\right) - \phi\left(\frac{1}{3}(u+v+w)\right)$$

is invariant under adding any affine term to ϕ. Then, $(u-v)\cdot J(w-v) = 2u\cdot Jw$, so our goal is to show $\phi\left(\frac{1}{3}(u+v+w)\right) \le -\frac{2}{9\sqrt{3}} u\cdot Jw$.

Define $A : \mathbb{R}^2 \to \mathbb{Z}$ by $A(x,y) = yw + \sqrt{3}xu$, so that A maps an equilateral triangle of unit height to the convex hull of u, v, w. Let $\psi(x,y) = \phi(A(x,y))$. Then,

$$\mathrm{d}^2\psi(x,y) = A^*(\mathrm{d}^2\phi)(A(x,y))A \succeq \imath A^* J A = \alpha\imath \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

where $\alpha = \sqrt{3}u\cdot Jw$. By lemma 4.1.4, we have $\det(\mathrm{d}^2\psi) \ge \alpha^2$. By lemma 4.2.3, we have,

$$\psi(0, 1/3) \le -\frac{2}{27}\alpha = -\frac{2}{9\sqrt{3}} u\cdot Jw$$

For the second part, suppose ϕ is continuous on the closure of $\mathcal{K}$, and let u, v, w be in that closure. Then there is a sequence of triples $(u_n, v_n, w_n) \in \mathcal{K}^3$ converging to (u, v, w). Then, as we have just established,

$$\frac{1}{3}\left(\phi(u_n) + \phi(v_n) + \phi(w_n)\right) - \phi\left(\frac{1}{3}(u_n + v_n + w_n)\right) \geq \frac{1}{9\sqrt{3}}(u_n - v_n) \cdot J(w_n - u_n)$$

The result follows by continuity of ϕ and J.

We now prove lemma 4.2.3. Without loss of generality, let $\mathcal{C}$ be centered at $(0, 1/3)$. Let $\triangle\psi$ be the Laplacian of ψ. Then, using lemma 4.1.5, and the fact that $\operatorname{tr}[Q] \geq 2\sqrt{\det Q}$ for $0 \preceq Q \in \mathbb{R}^{2\times 2}$,

$$\triangle\psi = \operatorname{tr}[\mathrm{d}^2\psi] \geq 2\sqrt{\det \mathrm{d}^2\psi} \geq 2$$

The function η defined by,

$$\eta(x, y) = \frac{1}{2}y(y - 1 - \sqrt{3}x)(y - 1 + \sqrt{3}x)$$

is the solution to the Poisson equation $\triangle\eta = 2$ on the interior and $\eta = 0$ on the boundary for an equilateral triangle, and $\eta(0, 1/3) = -2/27$. Then $\psi - \eta$ is subharmonic on the interior of $\mathcal{C}$ and non-positive on the boundary, so it is non-positive on the interior too.

We remark a tight bound would be obtained by using a solution to the Monge-Ampère equation, rather than the Poisson equation.

4.2.3 Hamiltonian Systems

Suppose ϕ is strictly convex, and suppose Φ is a 0-approximator for ϕ^*; that is,

$$\Phi(a) = \arg\max_{z' \in \mathcal{C}} a \cdot z' - \phi(z')$$

Then, ϕ^* is strictly convex with $(\mathrm{d}\phi^*)(a) = \Phi(a)$. A naive version of the algorithm might perform the step,

$$z(t + 1) = z(t) + \Phi(Jz(t))$$

Consider now a *continuous* dynamical system defined by $z(0) = 0$, and

$$(\mathrm{d}z)(t) = \Phi(Jz(t))\mathrm{d}t \tag{4.10}$$

Assuming a solution to the system exists, we have,

$$\mathrm{d}(\phi^*(Jz(t))) = (\mathrm{d}\phi^*)(Jz(t)) \cdot J(\mathrm{d}z)(t) = (\mathrm{d}z)(t) \cdot J(\mathrm{d}z)(t)\mathrm{d}t = 0$$

That is, the quantity $\phi^*(Jz)$ is conserved. Indeed, if we define, $\mathcal{H}(z) = \phi^*(Jz)$, we recover the generalized Hamilton equations,

$$J(\mathrm{d}z) = (\mathrm{d}\mathcal{H})(z(t))\mathrm{d}t$$

We may view various algorithms[36, 38], including that of theorem 4.1.3 as effectively discretizing (4.10) using various numerical integration rules.

We now examine a certain sense in which area-convexity is *tight*. Suppose $z' = \Phi(Jz)$ is in the interior of $\mathcal{C}$, and ϕ is twice differentiable at z'. Then, second-order duality theory[43] implies $(\mathrm{d}^2\phi^*)(Jz) = (\mathrm{d}^2\phi(z'))^{-1}$. That is, $(\mathrm{d}(\Phi \circ J))(z) = (\mathrm{d}^2\phi(z'))^{-1}J$. For higher-order stepping rules that involve iterating a map of the form $z' \mapsto \Phi(z + hJz')$, we may expect convergence only if the spectral radius of $h(\mathrm{d}^2\phi(z'))^{-1}J$ is below unity; in particular, for $h = 1$, convergence requires $(\mathrm{d}^2\phi(z'))^{-1}J$ be a contraction. That is,

$$J^*(\mathrm{d}^2\phi(z')^{-1})J \preceq \mathrm{d}^2\phi(z')$$

On the other hand, $J^*Q^{-1}J \preceq Q$ is exactly equivalent to $Q \succeq \imath J$ for invertible Q. Therefore, by theorem 4.1.6, strict area-convexity is, equivalent to *local* Lipschitz continuity of the time evolution operator. In contrast, strong convexity is equivalent to *global* Lipschitz continuity with respect to a fixed norm.

4.3 Stochastic Matrix Problems

We prove theorem 4.1.7 by constructing small area-convex regularizers, together with a fast δ-AMO. We take A to be fixed throughout the rest of the section, and define $\bar{A}$ by $\bar{A}_{ve} = |A_{ve}|$. We represent the domain by explicitly incorporating an extra $w \in \triangle_n$ bounding row-maximums for Y.

$$\begin{aligned}\mathcal{C} &:= \mathcal{C}_1 \oplus \triangle_k^m \\ \mathcal{C}_1 &:= \{(w, Y) : w \in \triangle_n, Y \geq 0, Y_{vj} \leq w_v\}\end{aligned}$$

Following the reduction in 4.1.1, we let $J(w, Y, X) = (0, AX, -A^*Y)$. Note we have ignored the linear terms C, B; as stated in 4.1.1, they may be incorporated by augmenting $\mathcal{C}$, J with an additional dimension.

Our regularizer repeatedly uses a gadget function $\mathsf{port}(w, y, x)$ of three *scalar* variables, defined for $x \geq 0$, $0 \leq y \leq w$, by

$$\mathsf{port}(w, y, x) = x\frac{y^2}{2w} + 3wx\log x \tag{4.11}$$

Note that if $w = 0$, we must have $y = 0$, and setting $\mathsf{port}(0, 0, x) = 0$ yields continuity at $0, 0, x$.

Using port, we define ϕ as,

$$\phi(w, Y, X) = \alpha \sum_{e,v,j} \bar{A}_{ev}\mathsf{port}(w_v, Y_{vj}, X_{ej}) + \alpha\beta \sum_v w_v \log w_v \tag{4.12}$$

We set $\alpha := 18, \beta := 27(2 + \log k)^2$.

To use theorem 4.1.3 we must show three things. First, ϕ must not be too large.

Lemma 4.3.1. *$|\phi(z)| \leq \mathcal{O}(\log(n)\log^2(k))$ for all $z \in \mathcal{C}$.*

We remark the massive β factor causes the second sum in ϕ to dominate.

Second, ϕ must have the required area-convexity property.

Lemma 4.3.2. *ϕ is area-convex with respect to $2\sqrt{3}J$ on $\mathcal{C}$*

We prove lemma 4.3.2 in 4.3.1. Roughly, after defining some useful notation, we are able to write expressions such as,

$$\mathrm{d}^2\mathsf{port}(w, y, x) \approx \frac{w}{x}(\mathrm{d}x)^2 + \frac{x}{w}(\mathrm{d}y)^2 \succeq \imath \mathrm{d}y \wedge \mathrm{d}x$$

and then, substituting w_v, Y_{vj}, X_{ej}, simply sum over e, v, j. The actual proof differs only in more precisely specifying "$\approx$", and defining notation.

Finally, we must implement an approximate minimization oracle for ϕ; given $a \in \mathbb{R}^n, B \in \mathbb{R}^{n\times k}, C \in \mathbb{R}^{m\times k}$, we must find $(w, Y, X) \in \mathcal{C}$ with,

$$a \cdot w + \mathrm{tr}[B^*Y] + \mathsf{T}[C^*X] - \phi(w, Y, X) \geq \phi^*(a, B, C) - \delta \tag{4.13}$$

We remark that a is not actually required because the w-block of J is zero, so the saddle point algorithm will only ever call the AMO with $a = 0$; we include it to formally satisfy the definition of an AMO.

To implement an AMO, we observe that maximizing over (w, Y) or X *separately* is easy, and then alternate between maximizing over each block. That is, we define an AMO by algorithm 1.

Algorithm 1: Approximate Minimizer for ϕ

Input: $a \in \mathbb{R}^n, B \in \mathbb{R}^{n\times k}, C \in \mathbb{R}^{m\times k}, \delta > 0$

Output: $(w, Y, X) \in \mathcal{C}$ such that
$a \cdot w + \mathrm{tr}[B^*Y] + \mathrm{tr}[C^*X] - \phi(w, Y, X) \geq \phi^*(0, B, C) - \delta$

Initialize $X \in \triangle_k^m$ arbitrarily

repeat $\mathcal{O}(\log \delta^{-1})$ **times**

  $(w, Y) \leftarrow \arg\max_{(w,Y)\in\mathcal{C}_1} w \cdot a + \mathrm{tr}[B^*Y] - \phi(w, Y, X)$

  $X \leftarrow \arg\max_{X\in\triangle_k^m} \mathrm{tr}[C^*X] - \phi(w, Y, X)$

end

Lemma 4.3.3. *For $\delta > 0$, algorithm 1 is a δ-AMO for ϕ and runs in $\mathcal{O}(k\mathsf{nnz}(A)\log(\delta^{-1}))$ time.*

We prove lemma 4.3.3 in 4.3.3. We also give formulas to implement the block maximization in 4.3.2, but remark they follow immediately by computing $\mathrm{d}\phi$, and we shall not discuss them in detail. In particular, for fixed w, Y, maximizing X reduces to maximizing entropy plus a linear term as it does for the common entropy-regularizer; for fixed X, after Y is eliminated, the same is true of w.

We now prove theorem 4.1.7 using those lemmas. Given ϵ, choose $\delta = \epsilon/2$, and set Φ to be algorithm 1 for that choice of δ. Then, the algorithm of theorem 4.1.3 requires $\mathcal{O}(\log(n)\log^2(k)\epsilon^{-1})$ iterations to find an ϵ-approximate solution $(w, Y, X) \in \mathcal{C}$. Each requires applying J and calling Φ twice, which takes $\mathcal{O}(k\mathsf{nnz}(A)\log(\delta^{-1}))$ time by theorem 4.3.3. Therefore, the overall running time is $\tilde{\mathcal{O}}(k\mathsf{nnz}(A)\epsilon^{-1})$.

4.3.1 Proof of Lemma 4.1.2

For two vectors a, b, we write $a \wedge b := a \otimes b - b \otimes a$, and we write $a^2 = a \otimes a$. Note that $a^2 \succeq 0$. By identifying $\{Y_{vj}, w_v, X_{ej}\}$ with functions from $\mathcal{C}$ to $\mathbb{R}$ that return the respective coordinates, we write $\mathrm{d}Y_{vj}, \mathrm{d}X_{ej}, \mathrm{d}w_v$ to denote the standard basis of the dual space $\mathbb{Z}^*$.

Define,

$$\mathcal{C}^+ := \{(w, Y, X) \in \mathcal{C} : w > 0, X > 0\}$$

We define a tensor field Q on $\mathcal{C}^+$ via,

$$Q(w, Y, X) = \alpha \sum_{e,v,j} \bar{A}_{ve} \left(\frac{X_{ej}}{w_v} (\mathrm{d}Y_{vj})^2 + 3\frac{w_v}{X_{ej}} (\mathrm{d}X_{ej})^2 \right) + \alpha\beta \sum_v \frac{(\mathrm{d}w_v)^2}{w_v} \tag{4.14}$$

While diagonal in the standard basis, Q approximates $\mathrm{d}^2\phi$ quite well.

Lemma 4.3.4. *For $z \in \mathcal{C}^+$,*

$$\frac{1}{3} Q(z) \preceq \mathrm{d}^2\phi(z) \preceq 2Q(z)$$

Before proving lemma 4.3.4, we use it to complete the proof of lemma 4.3.2.

For $(w, Y, X) \in \mathcal{C}^+$, using lemma 4.1.5 yields,

$$\begin{aligned}
\mathrm{d}^2\phi(w, Y, X) &\succeq \frac{1}{3} Q(w, Y, X) \\
&\succeq \alpha \sum_{e,v,j} \bar{A}_{ve} \left(\frac{X_{ej}}{3w_v} (\mathrm{d}Y_{vj})^2 + \frac{w_v}{X_{ej}} (\mathrm{d}X_{ej})^2 \right) \\
&\succeq \frac{\alpha}{\sqrt{3}} \sum_{e,v,j} A_{ve} \imath \mathrm{d}Y_{ej} \wedge \mathrm{d}X_{ej} \\
&= \frac{\alpha}{\sqrt{3}} \imath J
\end{aligned}$$

Since the closure of $\mathcal{C}^+$ is $\mathcal{C}$, and $\alpha = 18$, lemma 4.3.2 follows from theorem 4.1.6.

We now prove lemma 4.3.4. Let $\phi = \phi_1 + \phi_2$, and $Q = Q_1 + Q_2$, where ϕ_2 contains all $w_v \log w_v$ terms and Q_2 contains all $(\mathrm{d}w_v)^2$ terms. Note that $\mathrm{d}^2\phi_2 = Q_2$.

We use a useful lemma about the gadget. Note the gadget is just a function on $\mathbb{R}^3$ (note w, y, x are scalars in lemma 4.3.5), so the lemma follows by computing the components of the 3×3 matrix $\mathrm{d}^2\mathsf{port}$; for convenience we give the components in appendix 4.6.

Lemma 4.3.5 (Gadget Lemma)**.** *Let $w > 0$, $|y| \leq w$ and $0 < x \leq 1$. Then,*

$$\frac{1}{3} M(w, y, x) - E(w, x) \preceq \mathrm{d}^2\mathsf{port}(w, y, x) \preceq 2M(w, y, z) + E(w, x)$$

where,

$$\begin{aligned}
M(w, y, x) &= \frac{x}{w} (\mathrm{d}y)^2 + \frac{3w}{x} (\mathrm{d}x)^2 \\
E(w, x) &= 18x(2 - \log x)^2 \frac{(\mathrm{d}w)^2}{w}
\end{aligned}$$

The gadget lemma makes the argument described in the beginning of the section rigorous. In particular, if not for the E terms, we would have $\mathrm{d}^2\mathsf{port}(w,y,x) \succeq \Omega(1)\mathrm{d}y\wedge\mathrm{d}x$. The extra terms in ϕ have been added precisely to counter that problem.

Define,

$$R(w,X) = 18\alpha \sum_{e,v,j} \bar{A}_{ve} X_{ej} (2 - \log X_{ej})^2 \frac{(\mathrm{d}w_v)^2}{w_v}$$

Applying the gadget lemma 4.3.5 to each term of ϕ_1 yields,

$$\frac{1}{3}Q_1(w,Y,X) - R(w,X) \preceq \mathrm{d}^2\phi_1(w,Y,X) \preceq 2Q_1(w,Y,X) + R(w,X)$$

Adding $Q_2(w,Y,X)$, and noting $Q_2 = \mathrm{d}^2\phi_2$, it suffices to show $R(w,Y,X) \preceq \frac{2}{3}Q_2(w,Y,X)$. Since the function $t \mapsto t(2-\log t)^2$ is concave on $[0,1]$, every e satisfies,

$$\sum_j X_{ej}(2 - \log X_{ej})^2 \le (2 + \log k)^2$$

Therefore,

$$R(w,X) \preceq 18(2+\log k)^2 \alpha \sum_{e,v} \bar{A}_{ve} \frac{(\mathrm{d}w_v)^2}{w_v}$$

Recalling $\|\bar{A}\|_{\infty\to\infty} \le 1$, the choice $\beta = 27(2+\log k)^2$ yields the lemma.

4.3.2 Block Maximization

The block-maximizers use some common functions, defined here.

$$\begin{aligned}
\mathrm{chop}(t) &= \max(0, \min(1,t)) \\
\mathrm{srect}(b,x) &= \max_{t\in[0,1]} tb - \frac{x}{2}t^2 = \begin{cases} 0, & \text{if } b \le 0 \\ \frac{b^2}{2x}, & \text{if } 0 < b < x \\ b - \frac{1}{2}x & \text{else} \end{cases} \\
\mathrm{expwts}(x)_i &= \frac{\exp(x_i)}{\sum_j \exp(x_j)}
\end{aligned}$$

4.3.3 Convergence of AMO

Let $a \in \mathbb{R}^n, B \in \mathbb{R}^{n\times k}, C \in \mathbb{R}^{m\times k}$ be fixed throughout this subsection, and define,

$$\eta(w,Y,X) := \phi^*(a,B,C) - a\cdot w - \mathrm{tr}[B^*Y] - \mathrm{tr}[C^*X] + \phi(w,Y,X)$$

Our goal is to show the AMO approximately minimizes $\eta(z)$ over $z \in \mathcal{C}$.

Algorithm 2: Maximizing X

Input: $C \in \mathbb{R}^{m\times k}, (w, Y) \in \mathcal{C}_1$
Output: $X \in \triangle_k^m$ maximizing $\operatorname{tr}[C^*X] - \phi(w, Y, X)$

$\bar{w} \leftarrow \bar{A}^* w$
$Z_{vj} \leftarrow \alpha \frac{Y_{vj}^2}{2w_v}$
$\bar{Z} \leftarrow \bar{A}^* Z$
$X_e \leftarrow \operatorname{expwts}\left(\frac{C_e - \bar{Z}_e}{3\alpha \bar{w}_e}\right)$

Algorithm 3: Maximizing w, Y

Input: $a \in \mathbb{R}^n, B \in \mathbb{R}^{n\times k}, X \in \triangle_k^m$
Output: $(w, Y) \in \mathcal{C}_1$ maximizing $a \cdot w + \operatorname{tr}[B^*Y] - \phi(w, Y, X)$
$\bar{X} \leftarrow \bar{A}X$
$s_e \leftarrow \sum_j X_{ej} \log X_{ej}$
$\bar{s} \leftarrow \bar{A}s$
$b_v \leftarrow \sum_j \operatorname{srect}(B_{vj}, \alpha \bar{X}_{vj})$
$w \leftarrow \operatorname{expwts}\left(\frac{a+b-3\alpha\bar{s}}{\alpha\beta}\right)$
$Y_{vj} \leftarrow w_v \operatorname{chop}(B_{vj}/\alpha\bar{X}_{vj})$

For the convergence proof, we use the fact that w does not change much over the course of the algorithm. To see that, we consider the formula given in algorithm 3. Since $-\log k \leq s_e \leq 0$, and $\|\bar{A}\|_{\infty\to\infty} \leq 1$, we also have $-\log k \leq \bar{s}_v \leq 0$. Noting

$$\operatorname{srect}(t, 0) - \frac{x}{2} \leq \operatorname{srect}(t, x) \leq \operatorname{srect}(t, 0)$$

we have,

$$-\frac{1}{2} \leq \sum_j \operatorname{srect}(B_{vj}, \bar{X}_{vj}) - \sum_j \operatorname{srect}(B_{vj}, 0) \leq 0$$

where we used,

$$\sum_j \bar{X}_{vj} \leq \|\bar{X}\|_{\infty\to\infty} \leq \|\bar{A}\|_{\infty\to\infty}\|X\|_{\infty\to\infty} \leq 1$$

Since each coordinate $\bar{s}_v, b_v$ varies by at most $\mathcal{O}(\log k)$, and $\beta = \Omega(\log^2 k)$, it follows that each $\log w_v$ varies by at most $\mathcal{O}(1/\log(k))$. That is, we have established the following lemma.

Lemma 4.3.6. *Let $X, X' \in \triangle_k^m$. Let (w, Y) minimize $\eta(w, Y, X)$ and (w', Y') minimize $\eta(w', Y', X')$. Then, $(1-\gamma)w \leq w' \leq (1+\gamma)w$ for $\gamma = \mathcal{O}(1/\log k)$.*

We establish convergence of alternating minimization using analysis of Beck[9]. We state a simpler form using only quadratic norms, and using twice-differentiability, as that is all we require..

Theorem 4.3.7 (Beck[9]). *Let $\mathcal{C}_1, \mathcal{C}_2$ be compact-convex, and $\psi : \mathcal{C}_1 \oplus \mathcal{C}_2 \to \mathbb{R}$. Let $q \in \mathcal{C}_1$ be given, let $p \in \mathcal{C}_2$ minimize $\psi(q, p)$, and let $q' \in \mathcal{C}_1$ minimize $\psi(q', p)$. Let $\Pi(q, p) = (q, 0)$ project onto the first block. Suppose there is a quadratic form M, and $\kappa > 0$ such that, for all $q'' \in \mathcal{C}_1, p'' \in \mathcal{C}_2$,*

$$\mathrm{d}^2\psi(q'', p'') \succeq M \succeq \Pi \mathrm{d}^2\psi(q'', p)\Pi$$

Then, for any $q'' \in \mathcal{C}_1, p'' \in \mathcal{C}_2$,

$$\psi(q, p) - \psi(q', p) \geq \frac{1}{\kappa}(\psi(q, p) - \psi(q'', p''))$$

Let w, Y be given and fixed for the rest of the subsection, let X minimize $\eta(w, Y, X)$, and let w', Y' minimize $\eta(w', Y', X)$. Define,

$$\begin{aligned}
\Pi(w'', Y'', X'') &:= (w'', Y'', 0) \\
\tilde{\mathcal{C}}_1 &:= \{(w'', Y'') \in \mathcal{C}_1 : \frac{1}{2}w \leq w'' \leq 2w\} \\
\tilde{\mathcal{C}}_2 &:= \{X'' \in \triangle_k^m : X'' \geq \frac{1}{2}X\} \\
M &:= \frac{1}{12}\Pi Q(w, Y, X)\Pi
\end{aligned}$$

Note the definition of Q in (4.14), together with $\tilde{\mathcal{C}}_i$ imply that for $(w'', Y'') \in \tilde{\mathcal{C}}_1, X'' \in \tilde{\mathcal{C}}_2$, we have,

$$\Pi Q(w'', Y'', X'')\Pi \succeq \frac{1}{4}\Pi Q(w, Y, X)\Pi \succeq \frac{1}{8}\Pi Q(w'', Y'', X)\Pi$$

Then, by lemma 4.3.4,

$$\begin{aligned}
\mathrm{d}^2\eta(w'', Y'', X'') &\succeq \frac{1}{3}Q(w'', Y'', X'') \\
&\succeq \frac{1}{3}\Pi Q(w'', Y'', X'')\Pi \\
&\succeq M \\
&\succeq \frac{1}{24}\Pi Q(w'', Y'', X)\Pi \\
&\succeq \frac{1}{48}\Pi \mathrm{d}^2\eta(w'', Y'', X)\Pi
\end{aligned}$$

Lemma 4.3.6 establishes that $(w_{\mathsf{opt}}, Y_{\mathsf{opt}}) \in \tilde{\mathcal{C}}_1$, but have no guarantee about X_{opt}. However, we do have $\frac{1}{2}(X_{\mathsf{opt}} + X) \in \tilde{\mathcal{C}}_2$. Thus, for the convex combination $(w'', Y'', X'') = \frac{1}{2}(w_{\mathsf{opt}} + w, Y_{\mathsf{opt}} + Y, X_{\mathsf{opt}} + X)$, we have $\eta(w'', Y'', X'') \leq \frac{1}{2}\eta(w, Y, X)$, and Beck's theorem yields,

$$\eta(w, Y, X) - \min_{(w', Y') \in \mathcal{C}_1} \eta(w', Y', X) \geq \frac{1}{96}\eta(w, Y, X)$$

That is, each (w, Y) block-minimization reduces η by a constant factor.

4.4 Multicommodity Flow

In this section, we sketch a proof of theorem 4.1.9. We remark the algorithm to solve multicommodity flow using the approximate inequality solver of theorem 4.1.7 is essentially identical to our earlier algorithms[44, 45] that solve single-commodity flow using multiplicative weights.

We first observe the half-duplex variant with demands B easily reduces to the full-duplex version by doubling the number of commodities and considering demands $[B| - B]$ (i.e., the $n \times 2k$ matrix formed by concatenating the columns of $-B$ to the columns of B). A full-duplex flow routing those demands has the form $[F^+|F^-]$, where both $F^+, -F^-$ route B. Furthermore recall that each undirected edge e is replaced with two directed edges in the full-duplex variant, so we now have *four* quantities $F^+_{+ej}, F^+_{-ej}, F^-_{-ej}, F^-_{+ej}$. We may define a flow on the undirected edges by setting,

$$\tilde{F}_{ej} = \frac{1}{2}\left(F^+_{+ej} - F^+_{-ej} - F^-_{+ej} + F^-_{-ej}\right)$$

Lemma 4.4.1. *Suppose $[F^+|F^-]$ is $(1-\epsilon)$-optimal for the full-duplex problem with demands $[B| - B]$. Then, $\tilde{F}$ is $(1-\epsilon)$-optimal for the half-duplex problems with demands B.*

We shall instead prefer to work with a feasibility version closer to our stochastic matrix problem; given λ, B, we either feasibly route $(1-\epsilon)\lambda B$ or else output a certificate showing $\lambda_{\mathsf{opt}} < \lambda$. An approximate maximum-concurrent flow is easily found by binary search over λ. Furthermore, λ is redundant in the feasibility version: our task given λ, B is the same as when given $1, \lambda B$. Therefore, our task is reduced to the following: given B, either route $(1-\epsilon)B$ or else show B is infeasible.

Recall a feasible flow F may be expressed as $F = CX$ for right-sub-stochastic X; we would like to consider right-stochastic matrices instead. Let $b = Dc - B\mathbf{1}$, where $\mathbf{1}$ is the all-1 vector, and add an extra commodity with demands b, to obtain a new problem with demands $[B|b]$. We argue the original demands are feasible iff the new demands are routable by a flow where all edges use full capacity. Clearly, if $[F|f]$ is feasible and routes $[B|b]$, then F is feasible and routes B. On the other hand, if F feasibly routes B, then $f = c - F\mathbf{1}$ has $Df = D(c - F\mathbf{1}) = Dc - B\mathbf{1} = b$. That is, $[F|f]$ has $[F|f]\mathbf{1} = c$ and routes $[B|b]$.

Thus, our task is reduced to deciding if there is right-stochastic X satisfying $DCX = B$. The latter implies $RDCX = RB$ for any R. We may use corollary 4.1.8 to approximate such equality problems by doubling the rows to have two $RDCX \leq RB$ and $-RDCX \leq -RB$, to obtain lemma 4.4.2.

Lemma 4.4.2. *Let R be any matrix with $\|RDC\|_{\infty\to\infty} \leq 1$. Then, there is an algorithm that takes $\tilde{\mathcal{O}}(k\mathsf{nnz}(RDC)\epsilon^{-1})$ time and outputs either,*

1. *A feasible flow F such that $\|R(DF - B)\|_{\infty\to\infty} \leq \epsilon$*
2. *A dual solution Y showing B is infeasible to route.*

The conclusion is not quite what we want, due to the nonzero error $DF - B$. To eliminate that error, we follow the same paradigm we introduced originally for single-commodity

flow[44]. The idea is to observe that if we could guarantee the residual demands $\tilde{B} = B - DF$ are small in the sense that they are routable in G with capacities ϵc, we could recurse on the problem with demands $\tilde{B}$; after $\mathcal{O}(\log m)$ recursions, anything left is so tiny that even routing through e.g. a maximal spanning tree contributes at most $\mathcal{O}(\epsilon)$ congestion. On the other hand, the conclusion we have is that $|R(DF - B)\|_{\infty\to\infty} \leq \epsilon$. The main object we introduced in earlier max-flow work[44] is the *congestion-approximator*. The definition may be extended to multicommodity problems as follows. Let $\mathsf{opt}(B)$ be the maximum λ such that λB is feasible. Then,

Definition 4.4.3. *A κ-congestion-approximator is a matrix R such that, for any demands B,*

$$\|RB\|_\infty \leq \mathsf{opt}(B)^{-1} \leq \kappa \|RB\|_{\infty\to\infty}$$

The first part implies $\|RDC\|_{\infty\to\infty} \leq 2$, while the second implies that if we apply lemma 4.4.2 with $\epsilon' = \Omega(\epsilon/\kappa)$, any residual error may be feasibly routed with capacities $\epsilon c/3$.

Our original approximator construction[44] was not sparse; fortunately Peng[41] shows how to construct sparse approximators, with even better parameters.

Theorem 4.4.4 (Peng[41])**.** *There is an algorithm that, given a capacitated graph G with n vertices and m edges, takes $\tilde{\mathcal{O}}(m)$ time and outputs a $\log^{\mathcal{O}(1)}(n)$-congestion-approximator R for G with $\mathcal{O}(n)$ rows, each nonzero in $\mathcal{O}(\log n)$ columns.*

Thus, for Peng's approximator, RDC has $\tilde{\mathcal{O}}(m)$ nonzero entries.

4.5 Area-Convexity of Quadratics

To prove lemma 4.1.4, we may take a triangle centered at the origin without loss of generality. For x, y, if $z = -y - x$, then,

$$(y - z) \cdot J(x - z) = 3y \cdot Jx$$

Thus, ϕ is area-convex w.r.t J iff for all x, y,

$$\frac{1}{6}\left(x \cdot Qx + y \cdot Qy + (x + y) \cdot Q(x + y)\right) \geq \frac{1}{\sqrt{3}} yJx$$

That is,

$$0 \preceq \begin{bmatrix} 2Q & Q - \sqrt{3}J \\ Q + \sqrt{3}J & 2Q \end{bmatrix} =: \mathbf{M}$$

On the other hand, we defined $Q \succeq \imath J$ by,

$$0 \preceq \begin{bmatrix} Q & -J \\ J & Q \end{bmatrix} := \tilde{\mathbf{M}}$$

We now observe the latter two conditions are equivalent, because,

$$\mathbf{M} = A^* \tilde{\mathbf{M}} A \qquad \text{where } A = \begin{bmatrix} \sqrt{2} & \frac{1}{\sqrt{2}} \\ 0 & \sqrt{\frac{3}{2}} \end{bmatrix}$$

To prove lemma 4.1.5, it suffices to observe it holds for diagonal Q.

4.6 Three Port Gadget

Recall,

$$\mathsf{port}(w, y, x) = x\frac{y^2}{2w} + 3wx\log x$$

Then,

$$\mathrm{d}^2\mathsf{port}(w, y, x) = \begin{bmatrix} \frac{xy^2}{w^3} & -\frac{xy}{w^2} & 3\log(x) - \frac{y^2}{2w^2} + 3 \\ -\frac{xy}{w^2} & \frac{x}{w} & \frac{y}{w} \\ 3\log(x) - \frac{y^2}{2w^2} + 3 & \frac{y}{w} & 3\frac{w}{x} \end{bmatrix}$$

Chapter 5

Sparsest Cut

5.1 Introduction

We consider the problem of partitioning a graph into relatively independent pieces in the sense that not too many edges cross between them. Two concrete optimization problems arising in that context are the SPARSEST CUT and BALANCED SEPARATOR problems. We are given an undirected weighted graph G on n vertices, where each edge xy has capacity G_{xy} (we identify a graph with its adjacency matrix). The *edge expansion* of a cut $(S, \overline{S})$ is $h(S) = \frac{\sum_{x \in S, y \in \overline{S}} G_{xy}}{\min\{|S|, |\overline{S}|\}}$. The SPARSEST CUT problem is to find a cut $(S, \overline{S})$ minimizing $h(S)$; we write $h(G)$ to denote the value of such a cut. The BALANCED SEPARATOR problem has the same objective but the additional constraint that $\min\{|S|, |\overline{S}|\} \geq \Omega(n)$. Both problems are NP-hard, so we settle for approximation algorithms.

Most of the original work on graph partitioning focused on achieving the best approximation factor and falls into one of two themes. The first is based on multicommodity flow, using the fact that if a graph H of known expansion can be routed in G via a feasible flow, then $h(H) \leq h(G)$. If H is some fixed graph, finding the best possible lower bound is equivalent to solving the maximum concurrent flow problem; i.e., maximizing α such that $F \leq G$ and $D \geq \alpha H$ where $D_{xy} = \sum_{p: x \leftrightarrow y} f_p$ is the *demand graph* and $F_{xy} = \sum_{p \ni xy} f_p$ is the *flow graph* of the underlying flow. By taking H to be the complete graph, Leighton and Rao showed an upper bound of $h(G) \leq O(\log n)\alpha^* h(H)$ for the optimal α^*, yielding an $O(\log n)$ approximation. The other theme is the discrete Cheeger's inequality of Alon and Milman[3] characterizing the relationship between cuts and the spectrum of a graph's Laplacian matrix. In particular, if G has maximum degree d, then $\lambda_2(\mathcal{L}_G)/2 \leq h(G) \leq \sqrt{2d\lambda_2(\mathcal{L}_G)}$, where $\lambda_2(\mathcal{L}_G)$ is second smallest eigenvalue of G's Laplacian. The two themes are incomparable, as the latter is a better approximation when G is an expander (i.e., $h(G)/d$ is large) while the former is better when G has sparse cuts.

Arora, Rao, and Vazirani naturally combined the two themes. Rather than embedding a fixed graph H of known expansion, they embed an arbitrary H and then certify H's expansion via $\lambda_2(\mathcal{L}_H)$[7]. Since $\lambda_2(\mathcal{L}_H) \geq \alpha$ is equivalent to $\mathcal{L}_H \succeq \frac{\alpha}{n}\mathcal{L}_K$, where K is the complete graph,

the problem of finding the best such lower-bound can be cast as a semidefinite program:

$$\max \alpha \qquad s.t. \quad \frac{\alpha}{n}\mathcal{L}_K \preceq \mathcal{L}_D, \quad F \leq G \tag{5.1}$$

They showed that for the optimal α^*, one has an upper bound of $h(G) \leq O(\sqrt{\log n})\alpha^*$, yielding the currently best known approximation factor. Shortly thereafter, Arora, Hazan, and Kale designed a primal-dual algorithm to approximately solve (5.1) in $\tilde{O}(n^2)$ time using multicommodity flows[4].

More recently, researchers have focused on designing efficient algorithms for graph partitioning that beat the quadratic multicommodity flow barrier. Khandekar, Rao, and Vazirani designed a simple primal-dual framework for constructing such algorithms based on the *cut-matching game* and showed one could achieve an $O(\log^2 n)$ approximation in that framework using polylog max-flows[28]. Arora and Kale designed a very general primal-dual framework for approximately solving SDPs[6] using the *matrix multiplicative weights* method. They showed efficient algorithms for several problems could be designed in their framework, including an $O(\log n)$-approximation to SPARSEST CUT using polylog max-flows. They also showed one could achieve an $O(\sqrt{\log n})$-approximation in their framework using multicommodity flows, simplifying the previous algorithm of [4]. Orecchia *et al.* extended the cut-matching game framework of [28] to achieve an $O(\log n)$ approximation[40]. They present two slightly different algorithms, and remarkably, their second algorithm is the same as Arora and Kale's, even though they never explicitly mention any SDP. They also showed a lower bound of $\Omega(\sqrt{\log n})$ on the approximation factor achievable in the cut-matching framework, suggesting the framework might precisely capture the limits of current approximation algorithms and posed the question of whether $O(\sqrt{\log n})$ could be efficiently achieved in that framework.

5.1.1 Contribution

We tie those two lines of work together by simultaneously achieving the $O(\sqrt{\log n})$ approximation factors of the former with the nearly max-flow running time of the latter.

Theorem 5.1.1. *For any $\varepsilon \in [O(1/\log(n)), \Omega(1)]$, there is an algorithm to approximate the* SPARSEST CUT *and* BALANCED SEPARATOR *problems to within a factor of $O(\sqrt{\log(n)/\varepsilon})$ using only $O\left(n^{\varepsilon}\log^{O(1)}(n)\right)$ max-flows.*

Theorem 5.1.1 effectively subsumes the results of [4, 28, 6, 40], as taking $\varepsilon = \Theta(1/\log(n))$ yields an $O(\log(n))$ approximation using polylog max-flows, while any constant $\varepsilon < 1/2$ achieves an $O(\sqrt{\log(n)})$ approximation in sub-quadratic $\mathcal{O}(m^{1+\epsilon})$ time using the max-flow algorithm of chapter 3. We also show the cut-matching game framework of [28] can not achieve an approximation better than $\Omega(\log(n)/\log\log(n))$ without re-routing flow.

We build heavily on Arora and Kale's work, achieving our improvement by replacing their use of a black-box multicommodity flow solver with a specialized one that makes use of the additional structure present in the flow instances that arise. We begin in section 5.2 by reviewing the nature of those flow problems, as well as the main ideas behind the algorithms

of [4, 6, 40]. Having clarified the connection to partitioning, we also state our main technical result, theorem 5.2.3. In section 5.3 we describe the details of our algorithm, the correctness of which follows immediately from theorem 5.2.3. The proof of theorem 5.2.3 appears in section 5.4. Our lower-bound for the cut-matching game is then discussed in 5.5.

5.2 Expander Flows

Expander-flow based algorithms all work by approximately solving (5.1), either explicitly as in [4, 6], or implicitly as in [28, 40], by iteratively simulating play of its corresponding two-player zero-sum game. The game has two players: the embedding player and the flow player. The embedding player chooses a non-trivial embedding $\mathcal{V} = (\mathbf{v}_1, \ldots, \mathbf{v}_n) \in (\mathbb{R}^d)^n$ of the vertices of G. The flow player chooses a feasible flow $F \leq G$ supporting demands D with the goal of routing flow between points that are far away in the embedding. More precisely, the payoff to the flow player is:

$$\Phi(\mathcal{V}, D) = \frac{\sum_{x<y} D_{xy} \|\mathbf{v}_x - \mathbf{v}_y\|^2}{\frac{1}{n}\sum_{x<y} \|\mathbf{v}_x - \mathbf{v}_y\|^2}$$

For given demands D, the best response for the embedding player is the one-dimensional embedding given by an eigenvector of $\mathcal{L}_D$ of eigenvalue $\lambda_2(\mathcal{L}_D)$, yielding a value of $\lambda_2(\mathcal{L}_D)$. On the other hand, for a given embedding, the best response for the flow player is a solution to the weighted maximum multicommodity flow problem given by

$$\max \sum_{x<y} D_{xy} \|\mathbf{v}_x - \mathbf{v}_y\|^2 \qquad s.t. \quad F \leq G \tag{5.2}$$

The frameworks of [28, 6, 40] start with an initial embedding $\mathcal{V}^1$, such as all points roughly equidistant. On a given iteration t, the algorithm presents $\mathcal{V}^t$ to the flow player, who must either respond with demands D^t of value $\Phi(\mathcal{V}^t, D^t) \geq 1$, or a cut C^t of expansion at most κ, where κ is the desired approximation factor. In the latter case the algorithm terminates; in the former case, the demands are used to update the embedding for the next iteration. The precise update differs among each algorithm, but essentially vertices x, y with large D^t_{xy} will be squeezed together in the embedding. The analysis of [28, 6, 40] show that after T iterations, for sufficiently large T, their adaptive strategies actually played nearly as well as they could have in hindsight, in that

$$\lambda_2\left(\frac{\mathcal{L}_{D^1} + \cdots + \mathcal{L}_{D^T}}{T}\right) \geq \Omega(1)$$

Since averaging T feasible flows yields a feasible flow, after T iterations the graph $D = (D^1 + \cdots + D^T)/T$ with $\lambda_2(\mathcal{L}_D) \geq \Omega(1)$ has been routed in G. Thus, for a given graph, the algorithm either routes an $\Omega(1)$-expander-flow in G or else finds a cut of expansion κ. Using a binary search and scaling the edge capacities appropriately yields an $O(\kappa)$ approximation algorithm.

The embedding can be updated in nearly linear time, and $T = O(\log^{O(1)}(n))$, so the running time of such algorithms is dominated by the running time of the flow player. By sparsifying G (using e.g. [10]), we can and shall assume it has $m = O(n \log n)$ edges. Using Fleischer's multicommodity flow algorithm[22] as a black box, a nearly optimal pair of primal/dual solutions to (5.2) can be computed in $\tilde{O}(n^2)$ time. Note that (5.2) has demand weights for every pair of vertices, so $\Omega(n^2)$ space is required to even explicitly write it down. On the other hand, each $\mathbf{v}_x \in \mathbb{R}^{O(\log n)}$, so the weights $\|\mathbf{v}_x - \mathbf{v}_y\|^2$ are all implicitly stored in only $O(n \log n)$ space. Therefore, making use of the additional geometric structure of these instances is crucial to achieving sub-quadratic time. Implicit in all of [28, 6, 40] is a specialized algorithm to approximately solve (5.2). The actual algorithm used is the same in all three, and those algorithms differ only in their strategy for the embedding player.

In the next two subsections, we briefly sketch the single-commodity and multicommodity flow based algorithms of [6], and then describe how we tie the two together. In particular, our algorithm is essentially an "algorithmetization" of the multicommodity flow algorithm's analysis. For the rest of the section, suppose we have an embedding $\mathcal{V}$ with $\sum_{x<y} \|\mathbf{v}_x - \mathbf{v}_y\|^2 = n^2$, and let us further assume that the points are unique and $\|\mathbf{v}_x\| \leq 1$ for all x; i.e., the diameter is not much more than the average distance.

5.2.1 Using Single-Commodity Flows

Consider first the absolute simplest case, where $d = 1$ and the points are simply numbers in $[-1, 1]$. It is easy to see that since the points are in $[-1, 1]$, unique, and have average squared-distance $\Omega(1)$, there must be some interval $[a, b]$, where $b - a = \Omega(1) =: \sigma$ and the set of points to the left of a, $A = \{x : \mathbf{v}_x \leq a\}$ and to the right of b, $B = \{y : \mathbf{v}_y \geq b\}$ have $|A| = |B| = \Omega(n) =: 2cn$. A natural way to try to push flow far along this line would be to shrink A and B down to single vertices and then compute a max-flow from A to B.

`FlowAndCut`$(\kappa, c, w_1, \ldots, w_n \in \mathbb{R})$:

- Sort $\{w_x\}$, let A be the $2cn$ nodes x with least w_x and B be those with greatest w_y.
- Add two vertices s, t. Connect s to each $x \in A$ and t to each $y \in B$ with edges of capacity κ.
- Output the max-flow/min-cut for $s - t$.

Consider invoking `FlowAndCut`$(\kappa, c, \mathbf{v}_1, \ldots, \mathbf{v}_n)$ with $\kappa = c^{-1}\sigma^{-2}$. If the max-flow is at least κcn, then since all flow must cross the gap $[a, b]$, we have pushed κcn units of flow across a squared-distance of σ^2, achieving a solution D with $\Phi(\mathcal{V}, D) \geq (\kappa cn\sigma^2)/n = 1$. Otherwise, if the min-cut is at most κcn, then at most cn of the added κ-capacity edges are cut, so at least cn vertices must remain on each side and the cut has expansion at most κ. That is, for dimension one a $\kappa = O(1)$ approximation is obtained.

The approach of [6, 40] is to reduce the general case to the one-dimensional case by picking a random standard normal vector u and projecting each $\mathbf{v}_x$ along u, yielding the 1-dimensional embedding $w_x = \mathbf{v}_x \cdot u$. The fact that the points are in the unit ball and have

average distance $\Omega(1)$ implies that with probability $\Omega(1)$, there is a gap $[a, b]$ with $b - a = \Omega(1) = \sigma$ as before. Applying the previous analysis, we either find a cut of expansion $O(1)$ or a flow with $\sum_{x<y} D_{xy}(w_x - w_y)^2 \geq n$. Then, the Gaussian tail ensures that distances could not have been stretched too much along u: with high probability $(w_x - w_y)^2 \leq O(\log n)\|\mathbf{v}_x - \mathbf{v}_y\|^2$ for every pair x, y. Thus, $\Phi(\mathcal{V}, D) \geq \Omega(1/\log(n))$, yielding an $O(\log n)$ approximation.

5.2.2 Using Multi-Commodity Flows

Arora, Rao, and Vazirani showed that, if a *best* response D^* to $\mathcal{V}$ has $\Phi(\mathcal{V}, D^*) \leq 1$, one can find a cut of expansion $O(\sqrt{\log n})$. Supposing the optimal solution to (5.2) has value at most n, there must be a solution to the dual problem of value at most n. The dual assigns lengths $\{w_e\}$ to the edges of G, aiming to minimize $\sum_e G_e w_e$ subject to the constraints that the shortest-path distances between each x, y under $\{w_e\}$ are at least $\|\mathbf{v}_x - \mathbf{v}_y\|^2$. Arora and Kale show the existence of such a dual solution implies that projecting the points along a random u and running `FlowAndCut`with $\kappa = \Theta(\sqrt{\log n})$ must yield a cut of capacity at most κcn with probability $\Omega(1)$.

If not, then a flow of value at least κcn is returned for $\Omega(1)$ of the directions u along which A and B are σ-separated. For simplicity, assume that the flows actually correspond to a matching between A and B. That is, each x has either zero flow leaving, or else has exactly κ flow going to a unique y along a single path. On the one hand, that matching is routed in G along cn flowpaths, each carrying flow κ. On the other hand, the total volume of G is only $\sum_e G_e w_e = n$, so $\Omega(n)$ of those flowpaths must have length at most $O(1/\kappa)$ under $\{w_e\}$.

For each u, let $M(u)$ be the matching consisting of those demand pairs routed along such short paths. Then, according to the following definition, M is an $(\Omega(1), \Omega(1))$-matching-cover.

Definition 5.2.1. *A (σ, δ)-**matching-cover** for an embedding $\{\mathbf{v}_x\}$ is a collection $\{M(u)\}_{u \in \mathbb{R}^d}$ of directed matchings satisfying the following conditions.*

- *Stretch: $(\mathbf{v}_y - \mathbf{v}_x) \cdot u \geq \sigma$ for all $(x, y) \in M(u)$*
- *Skew-symmetry: $(x, y) \in M(u)$ iff $(y, x) \in M(-u)$*
- *Largeness: $\mathbf{E}_u [|M(u)|] \geq \delta n$*

For a list of vectors $u_1, \ldots, u_R$, let $M(u_1, \ldots, u_R)$ denote the graph that contains edge (x, y) iff there exist $x_0, \ldots, x_R$ with $x_0 = x, x_R = y$ and $(x_{r-1}, x_r) \in M(u_r)$ for all $r \leq R$. For the empty list, let $M()$ denote the graph where each vertex has a directed self-loop. Note that $M(u_1, \ldots, u_R)$ is not a matching, but rather a graph with maximum in-degree and out-degree one.

Furthermore, M has the property that for each edge $(x, y) \in M(u)$, the distance between x and y under $\{w_e\}$ is at most $O(1/\kappa)$. The following theorem holds for M.

Theorem 5.2.2 ([31], refining [7])**.** *Let M be a $(\Omega(1), \Omega(1))$-matching-cover for $\{\mathbf{v}_x\}$. Then, there are vertices x, y and $u_1, \ldots, u_R$ where $R \leq O(\sqrt{\log n})$ such that $(x, y) \in M(u_1, \ldots, u_R)$ and $\|\mathbf{v}_x - \mathbf{v}_y\|^2 \geq L$.*

In other words, there are vertices x, y with $\|\mathbf{v}_x - \mathbf{v}_y\|^2 \geq L$ that are only R matching hops away in M. Applying theorem 5.2.2, there are vertices x, y with $\|\mathbf{v}_x - \mathbf{v}_y\|^2 \geq L$ but of distance only $RO(1/\kappa)$ under $\{w_e\}$. Choosing $\kappa = O(R/L) = O(\sqrt{\log n})$ yields a contradiction to the assumption that $\{w_e\}$ is dual feasible.

5.2.3 Results

Our improvement comes from being able to achieve an $O(\sqrt{\log n})$ gap between cut and flow solutions, as in the latter case, while still only using single-commodity flows, as in the former case. Recall the case of $d > 1$ was reduced to the $d = 1$ case by projecting along a random vector and bounding the squared-stretch by $O(\log n)$. Indeed, the stretch could be nearly that much, so simply pushing flow along a single direction will not allow us to achieve anything better; in fact, that is the main idea behind our lower bound for the cut-matching game.

To do better, we need to do something more sophisticated than simply push flow along a single direction. A natural idea is to try to pick several directions $u_1, \ldots, u_R$, push flow along each of them, and then try to glue the flows together to actually push flow far away globally. One motivation for such an approach is that it seems to be the next simplest thing to do, following that of using only a single direction. The second and most crucial motivation is to observe that *such an approach is strongly suggested by the analysis for the multicommodity flow algorithm just sketched.* To see that, suppose the typical flowpath along a random u routes between points of squared-distance Δ. Theorem 5.2.2 says we can always augment $R = O(\sqrt{\log n})$ such flowpaths to route demand between points of squared-distance $L = \Omega(1)$, at the cost of possibly raising congestion by a factor of R. Thus, either $\Delta \geq L/R = \Omega(1/\sqrt{\log n})$, or else augmenting together R typical flowpaths and scaling down by R maintains feasibility and increases the objective of (5.2).

Unfortunately, theorem 5.2.2 doesn't say anything at all about *finding* such directions u, or whether the same $u_1, \ldots, u_R$ will simultaneously work for many vertices. To analyze such an algorithm, we need a stronger, algorithmic version of theorem 5.2.2. Our main technical contribution is such a theorem.

Theorem 5.2.3. *For any $1 \leq R \leq \Theta(\sqrt{\log(n)})$, there is $L \geq \Theta(R^2/\log(n))$ and an (efficiently sample-able) distribution D over $(\mathbb{R}^d)^{\leq R}$ with the following property.*

If M is an $(\Omega(1), \Omega(1))$-matching-cover for $\{\mathbf{v}_x\}$, then the expected number of edges (x, y) with $(x, y) \in M(D)$ and $\|\mathbf{v}_x - \mathbf{v}_y\|^2 \geq L$ is at least $e^{-O(R^2)}n$.

Using theorem 5.2.3 and choosing $R = \Theta(\sqrt{\varepsilon \log(n)})$, $L = \Theta(\varepsilon)$, we simply sample $u_1, \ldots, u_R$ from D, and let our final flowpaths be the concatenation of those along $u_1, \ldots, u_R$. On average, we get $n^{1-\varepsilon}$ such paths, and thus have *simultaneously* routed $n^{1-\varepsilon}$ paths between points of squared distance L using only R single-commodity flows. Using an iterative

re-weighing scheme and repeating $O(n^{\varepsilon}\log^{O(1)} n)$ times, we achieve a feasible flow and an approximation ratio of $O(R/L) = O(\sqrt{\log(n)/\varepsilon})$.

That is, to push flow far away, we sample $u_1, \ldots, u_R$ from D and then iteratively push flow along each direction. The distribution D will essentially consist of picking a random direction u_1, and then choosing u_{r+1} to be a $1 - 1/R$-correlated copy of u_r; i.e., a vector extremely close to u_r. Because u_{r+1} and u_r are so close, it is intuitively clear and easy to argue that *if* flow gets pushed along at each step, it must be pushed far away, as the projections along each u_r will essentially add together. The somewhat counterintuitive fact is that flow actually does get pushed further along in this manner. Even though u_r and u_{r+1} are extremely close together, a significant fraction of vertices that were in the "sink set" along u_r will be in the "source set" along u_{r+1}. That phenomenon is a consequence of measure concentration.

5.3 The Algorithm

While we found it most convenient to discuss expander flows and the corresponding game in the context of the SPARSEST CUT problem, our algorithm applies most directly to BALANCED SEPARATOR, which has a similar SDP relaxation and game. Roughly, the difference is that in the BALANCED SEPARATOR case the embedding player must choose an embedding for which the maximum squared distance between points is not much larger than the average. When the average distance is $\Theta(1)$, this is equivalent to the requirement that $\|\mathbf{v}_x\| \leq O(1)$ assumed earlier in section 5.2. The reduction from SPARSEST CUT to BALANCED SEPARATOR is well-known, and in fact, the unbalanced case is "easy" in the sense that if the cut found is unbalanced, it will be an $O(1)$ approximation to the sparsest cut[7]. In particular, Arora and Kale show that one can either obtain an $O(1)$ cut/flow gap with a single max-flow, or else reduce the problem to the balanced case by finding $\Omega(n)$ points in a ball of radius $O(1)$ that are still spread-out within that ball; for details, we refer the reader to [6].

The precise statement of the results sketched in section 5.2 is the following main lemma of [6].

Lemma 5.3.1 ([6]). *Let $U \subseteq [n]$ be a set of nodes. Suppose we are given vectors $\mathcal{V} = \{\mathbf{v}_x\}_{x\in U}$ of length at most $O(1)$ such that $\sum_{x,y\in U} \|\mathbf{v}_x - \mathbf{v}_y\|^2 = n^2$.*

- *There is an algorithm that uses $O(1)$ expected max-flow computations and outputs either a demand graph D on U of max-degree $O(\log(n))$ that is routable in G with $\Phi(\mathcal{V}, D) \geq 1$ or a balanced cut of expansion $O(\log n)$.*
- *There is an algorithm that uses a single multicommodity flow computation and $O(1)$ expected max-flow computations and outputs either a demand graph D on U of max-degree $O(1)$ that is routable in G with $\Phi(\mathcal{V}, D) \geq 1$ or a balanced cut of expansion $O(\sqrt{\log n})$.*

The importance of the degree is for the running time; if each D^t has max-degree β, then the total number of iterations needed is $O(\beta\log(n))$[6]. To prove theorem 5.1.1, we replace lemma 5.3.1 with the following.

Lemma 5.3.2. *Let U, $\mathcal{V}$ be as in lemma 5.3.1. For any $\varepsilon \in [O(1/\log(n)), \Omega(1)]$, there is an algorithm that uses $O(n^{\varepsilon}\log^{O(1)}(n))$ expected max-flow computations and outputs either a demand graph D on U of max-degree $O(1/\varepsilon)$ routable in G with $\Phi(\mathcal{V}, D) \geq 1$ or a balanced cut of expansion $O(\sqrt{\log(n)/\varepsilon})$.*

For the rest of this section, we prove lemma 5.3.2. We first immediately try to find a cut, using `FlowAndCut`. The parameters c, σ are set by the following lemma.

Lemma 5.3.3 ([7])**.** *Let U, $\mathcal{V}$ be as in lemma 5.3.1. Then, there exist $c, \sigma, \gamma = \Omega(1)$ so for a random u, with probability at least γ the sets A, B in `FlowAndCut`$(\cdot, c, \{\mathbf{v}_x \cdot u\}_{x\in U})$ have $(\mathbf{v}_y - \mathbf{v}_x)\cdot u \geq \sigma$ for all $x \in A, y \in B$.*

Let us call the u described by lemma 5.3.3 *good*, and set $\delta = \gamma c/16$. Let $\varepsilon \in [O(1/\log(n)), \Omega(1)]$ be given so that $R = O(\sqrt{\varepsilon \log n})$ yields an expected size bound of $n^{1-\varepsilon}$ in theorem 5.2.3. Set $L = \Omega(\varepsilon)$ as in theorem 5.2.3, $\kappa = 24R/cL$, and $\beta = 12/cL$. The following easy lemma was sketched in section 5.2.

Lemma 5.3.4 ([28, 6])**.** *If `FlowAndCut`$(\kappa, c, \ldots)$ returns a cut of capacity at most κcn, then the cut is cn-balanced and has expansion at most κ.*

We sample $O(\log(n))$ independent u, and run `FlowAndCut`$(\kappa, c, \{\mathbf{v}_x \cdot u\})$. If we ever find a cut of capacity at most κcn, we immediately output it and stop, yielding a balanced cut of expansion $\kappa = O(\sqrt{\log(n)/\varepsilon})$. Otherwise, with very high probability, we are in the situation where there are at least $\gamma/2$ good u for which a flow of value at least κcn is returned. In the latter scenario, we will find a flow D with $\Phi(\mathcal{V}, D) \geq 1$.

5.3.1 Finding a Flow

We efficiently find a solution to the maximum multicommodity flow problem

$$\begin{aligned} &\max \sum_{x<y} D_{xy}\|\mathbf{v}_x - \mathbf{v}_y\|^2 \\ s.t. \quad &F \leq G, \quad \max_x \deg_D(x) \leq \beta \end{aligned} \tag{5.3}$$

of value at least n. The dual assigns lengths $\{w_e\}$ to edges and $\{w_x\}$ to the vertices, with the constraint that the shortest path distance from x to y under these lengths dominate $\|\mathbf{v}_x - \mathbf{v}_y\|^2$.

$$\begin{aligned} &\min \sum_e G_e w_e + \sum_x \beta w_x \\ s.t. \quad &\forall p : x \leftrightarrow y \quad w_x + w_y + \sum_{e\in p} w_e \geq \|\mathbf{v}_x - \mathbf{v}_y\|^2 \end{aligned}$$

We use the *multiplicative weights* framework to approximately solve (5.3).

Theorem 5.3.5 ([42, 23, 5])**.** *Let $A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m$ with $b > 0$, and consider the following iterative procedure to find an approximate solution to $Ax \le b$.*

Initialize $y^1 \in \mathbb{R}^m$ to the all-1s vector. On iteration t, query an oracle that returns x^t such that $0 \le Ax^t \le \rho b$ and $y^t \cdot Ax^t \le y^t \cdot b$, and then update

$$y_j^{t+1} \leftarrow \left(1 + \eta \frac{(Ax^t)_j}{\rho b_j}\right) y_j^t$$

If $0 < \eta < 1/2$, then after $T = \rho\eta^{-2}\log(n)$ iterations, $A\left(\frac{x^1+\cdots+x^T}{T}\right) \le (1+4\eta)b$.

We use theorem 5.3.5 with $\eta = 1/4$, initializing the dual variables $\{w_e\}, \{w_x\}$ and updating them accordingly. On iteration t, we find a flow (F^t, D^t) of objective value $2n$ that violates the constraints by at most a factor of $\rho = O(n^{2\varepsilon}\log n)$ and

$$\sum_e w_e F_e^t + \sum_x w_x \deg_{D^t}(x) \le \sum_e w_e G_e + \sum_x w_x \beta \tag{5.4}$$

After $T = O(n^{2\varepsilon}\log^2(n))$ rounds, scaling the average flow down by 2 yields a feasible flow of objective value n. Noting that (5.4) and the algorithm of theorem 5.3.5 are invariant to scaling of the dual variables, for convenience we will also scale them on each iteration so that $\sum_e w_e G_e + \sum_x w_x \beta = 2n$. In that case, any flow of objective value $2n$ that only routes along violated or tight paths (those $p : x \leftrightarrow y$ for which $\sum_{e\in p} w_e + w_x + w_x \le \|\mathbf{v}_x - \mathbf{v}_y\|^2$) satisfies (5.4). In our algorithm, we will only route flow along paths $p : x \leftrightarrow y$ for which $\|\mathbf{v}_x - \mathbf{v}_y\|^2 \ge L$, and $w_x, w_y, \sum_{e\in p} w_e \le L/3$.

All flows will come from augmenting flows returned by `FlowAndCut`, where we identify single-commodity flows in $G \cup \{s,t\}$ with multicommodity flows in G in the obvious way. If F is an acyclic $s-t$ flow in $G \cup \{s,t\}$, it is well-known that F can be decomposed into at most m flowpaths. While computing such a decomposition could require $\Omega(nm)$ time, fortunately we need only pseudo-decompose flows in the following sense.

Definition 5.3.6. *If F is an acyclic $s-t$ flow in $G \cup \{s,t\}$ with a flow decomposition $((f_i, p_i))_{i\le m}$, then a list $P = ((f_i, s_i, t_i, \ell_i))_{i\le m}$ where $p_i = s, s_i, \ldots, t_i, t$ and $\sum_{e\in p_i} w_e = \ell_i$ is a **pseudo-decomposition** of F. That is, a pseudo-decomposition is a list containing the amount of flow, second vertex, second-to-last vertex, and length of each flowpath.*

The following two lemmas are easy applications of dynamic trees(see [46]).

Lemma 5.3.7. *Given a flow F on $G \cup \{s,t\}$, a pseudo-decomposition can be computed in $O(m\log n)$ time.*

Lemma 5.3.8. *Given a flow F on $G \cup \{s,t\}$, and a desired scaling vector $(\alpha_1, \ldots, \alpha_m)$, we can compute the flow F' with decomposition $\{(\alpha_k f_k, p_k)\}$ in $O(m\log n)$ time.*

Lemma 5.3.8 allows us to efficiently cherry-pick "good" flowpaths from the flows returned by `FlowAndCut`.

In their analysis, Arora and Kale round the flows returned by `FlowAndCut` to matchings. We do the same, with a small change to ensure doing so does not raise congestion by too much.

`Matching`(u)

- Call `FlowAndCut`$(\kappa, c, \{\mathbf{v}_x \cdot u\})$ and pseudo-decompose the resulting flow into P. Set $M = \emptyset$.
- Throw away any $(f_i, s_i, t_i, \ell_i) \in P$ with $(\mathbf{v}_{t_i} - \mathbf{v}_{s_i}) \cdot u < \sigma$, $f_i < \kappa cn/4m$, $w_{s_i} > L/3$, $w_{t_i} > L/3$, or $\ell_i > L/3R$.
- Greedily match the remaining pairs: iteratively pick $(f_i, s_i, t_i, \ell_i) \in P$, add (s_i, t_i) to M, and remove any $(f_j, s_j, t_j, \ell_j) \in$ with $\{s_i, t_i\} \cap \{s_j, t_j\} \neq \emptyset$.
- Output M.

The following lemma is essentially the same as one used in [6], and follows by the choice of parameters. The congestion bound, which was not needed for their analysis but is needed for our algorithm, comes from the fact that `Matching` discards any flowpath with $f_i \leq \kappa cn/4m$ before scaling any remaining flows to 1.

Lemma 5.3.9. *`Matching` is a (σ, δ)-matching-cover. Furthermore, for each u, the (unit-weighted) demands `Matching`(u) are simultaneously routable in G with congestion at most $4m/\kappa cn$ along flowpaths of length at most $L/3R$ under $\{w_e\}$.*

Proof. The symmetry and stretched properties hold by construction, so we need only establish the largeness property. Let u be a good direction for which the returned flow has value at least κcn, and let D be the corresponding demands. Since u is good, every demand pair is σ-separated along u. Each $x \in U$ has degree at most κ and the total degree is at least $2\kappa cn$. Deleting each path with $f_i \leq \kappa cn/4m$ removes at most $\kappa cn/4$ total flow. Since $\sum_x w_x \beta \leq 2n$ and $\beta = 12/cL$, at most $cn/4$ vertices can have $w_x > L/3$; deleting them removes at most $\kappa cn/4$ units of flow. Finally, since the original flow was feasible in the original graph,

$$\sum_p f_p \ell_p = \sum_e w_e \sum_{p \ni e} f_p \leq \sum_e w_e G_e \leq 2n$$

Since $\kappa = 24R/cL$, at most $\kappa cn/4$ units can flow along paths longer than $L/3R$.

In total, the second step of `Matching` removes at most $3\kappa cn/4$ units of flow, so at least $\kappa cn/2$ total degree survives. Each greedy matching step decreases the total degree by at most 4κ, so at least $cn/8$ pairs must get matched. Thus, the expected size of `Matching`(u) is at least $(\gamma/2)(cn/8) = \delta$.

For the congestion bound, we threw away all paths with flow less than $\kappa cn/4m$, so scaling the remaining paths to 1 yields a flow with congestion at most $4m/\kappa cn$. □

On each iteration, we sample $u_1, \ldots, u_R$ from the distribution D of theorem 5.2.3 and call `Matching`(u_r). Let D' be the unit-weighted graph with an edge (x, y) for each $(x, y) \in$

$\texttt{Matching}(u_1, \ldots, u_R)$ with $\|\mathbf{v}_x - \mathbf{v}_y\|^2 \geq L$. By theorem 5.2.3, the expected size of D' is at least $n^{1-\varepsilon}$, so after $n^\varepsilon/2$ expected trials, we have $|D'| \geq n^{1-\varepsilon}/2$. Applying lemma 5.3.8 again R times, we can compute a flow F' that routes D' in G with congestion $R(4m/\kappa cn) = O(\log(n)/L)$, since $m = O(n \log n)$ by assumption. Note also that D' has max-degree 2.

Then, D' achieves an objective value of at least $|D'|L$, so scaling up by $2n/|D'|L$ yields a solution of value $2n$ that satisfies (5.4) and congests edges by at most an $O(n^\varepsilon \log n)$ factor. Since $\beta = 12/cL = \Omega(1/\varepsilon)$, the degree constraints are also violated by at most an $O(n^\varepsilon)$ factor. The running time is dominated by flow computations, of which there are an expected $O(Rn^\varepsilon)$ in each of $O(n^\varepsilon \log^2(n))$ iterations, for a total of $O(n^{2\varepsilon} \log^{5/2}(n))$ expected max-flows.

5.4 Proof of Theorem 5.2.3

Let M be a (σ, δ)-matching cover. We identify M with a weighted directed graph, where edge (x, y) is has weight $\mathbf{Pr}_u[(x, y) \in M(u)]$. The skew-symmetry condition ensures the weights of (x, y) and (y, x) are the same, as are the in-degree and out-degree of each x. The total out-degree of M is at least δn by assumption. Following [7], we first prune M to a more uniform version by iteratively removing any vertex of out-degree less than $\delta/4$. Doing so preserves skew-symmetry, and at least $\delta n/2$ out-degree remains. It follows that we are left with a matching cover on vertices X, with $|X| \geq \delta n/2$ and every $x \in X$ has out-degree at least $\delta/4$. The pruned M is a $(\sigma, \delta/4)$-uniform-matching-cover.

Definition 5.4.1. *A (σ, δ)-**uniform-matching-cover** of $X \subseteq [n]$ is a $(\sigma, 0)$-matching-cover where every $x \in X$ has in-degree at least δ in M.*

5.4.1 Chaining and Measure Concentration

Let $y \in X$, and let A be the set of u for which y has an out-edge in $M(u)$. The main idea behind the proof of theorem 5.2.2 is the following. Since A and $-A$ are two sets of measure $\Omega(1)$, the isoperimetric profile of Gaussian space implies there must be many $u \in A, \hat{u} \in -A$ that are very close: $\|u - \hat{u}\| \leq O(1)$ (we remark that [7] uses the uniform measure on the sphere, but the same analysis holds for Gaussians after scaling various quantities by $\sqrt{d}$). Choose x, z with $(x, y) \in M(\hat{u})$, $(y, z) \in M(u)$ and observe that

$$\begin{aligned}(\mathbf{v}_y - \mathbf{v}_x) \cdot u &= (\mathbf{v}_y - \mathbf{v}_x) \cdot \hat{u} - (\mathbf{v}_y - \mathbf{v}_x) \cdot (\hat{u} - u) \\ &\geq \sigma - \|\mathbf{v}_x - \mathbf{v}_y\| \|\hat{u} - u\|\end{aligned}$$

Thus, either $\|\mathbf{v}_x - \mathbf{v}_y\| \geq \Omega(\sigma)$, or else $(\mathbf{v}_y - \mathbf{v}_x) \cdot u \geq \sigma/2$. In the former case, a matching edge joins two points of distance $\Omega(1)$. In the latter case, replacing the edge $(y, z) \in M(u)$ with (x, z) yields an edge with $(\mathbf{v}_z - \mathbf{v}_x) \cdot u \geq (3/2)\sigma$. By an inductive argument, the chaining case can be repeated until an edge connects two points of distance $\Omega(1)$. On the one hand, after R chaining steps, we have pairs of points that are R matching-hops apart, $O(1)$ distance apart, and have projection $\Theta(R)$. On the other hand, with high probability, no pair of distance $\Theta(1)$ has projection $\Theta(\sqrt{\log n})$, so the process must end after $\Theta(\sqrt{\log n})$ steps.

To turn the argument into an algorithm, we choose a sequence of highly correlated directions $u_1, \ldots, u_R$. For $R \geq 1$ and $0 \leq \rho \leq 1$, let $\mathcal{N}_\rho^R$ be the distribution of $u_1, \ldots, u_R$ defined by choosing a standard normal u_1, and then choosing each $u_{r+1} \sim_\rho u_r$ to be a ρ-correlated copy of u_r. That is, each of the d coordinate vectors $(u_{1,i}, \ldots, u_{R,i})$ are independently distributed as multivariate normals with covariance matrix $\Sigma_{r,r'} = \rho^{|r-r'|}$. In fact, simply setting $D = \mathcal{N}_{1-1/R}^R$ achieves theorem 5.2.3 for $R \leq O(\log^{1/3}(n))$ and size bound of $e^{-O(R^3)}n$. The barrier is essentially the same as the one that limited the original analysis of [7] to $R = O(\log(n)^{1/3})$. To overcome that barrier, we algorithmetize Lee's improvement[31] by independently sampling uncorrelated $\mathbf{w}_1, \ldots, \mathbf{w}_R$, and then shuffling the two lists together. The idea is that the highly correlated u_r will give us long stretch, while the $\mathbf{w}_r$ will greatly increase the probability of forming a long chain, at the cost of losing some stretch. The sampling algorithm is:

`Sample`(R, ρ)

- Pick $u_1, \ldots, u_R \sim \mathcal{N}_\rho^R$, $\mathbf{w}_1, \ldots, \mathbf{w}_R \sim \mathcal{N}_0^R$.
- Pick a random shuffling of the two lists, pick a random $r \leq R$, and output the first r elements of the shuffled list.

The reason for the randomness is to keep the algorithm trivial, leaving the work to our analysis. We show that there exists a particular shuffling and $r \leq R$ for which `Sample` is good; by randomly guessing, we lose at most a 2^{R+1} factor in our final expectation bound, which is negligible relative to the $e^{-O(R^2)}n$ bound we are aiming for.

Our proof of theorem 5.2.3 closely follows Lee's proof of theorem 5.2.2, the main difference being the use of a stronger isoperimetric inequality. The the standard isoperimetric inequality says that if A is a set of large measure, then for almost points u, a small ball around u has non-empty intersection with A. We use a stronger version, saying that if A is a set of large measure, then for almost all points u, a small ball around u has a significantly large intersection with A.

Lemma 5.4.2. *Let $A \subseteq \mathbb{R}^d$ have Gaussian measure $\delta > 0$. If $u, \hat{u}$ are ρ-correlated with $0 \leq \rho < 1$, then*

$$\mathbf{Pr}_u \left[\mathbf{Pr}_{\hat{u}} \left[\hat{u} \in A \right] < (\varepsilon\delta)^{1/(1-\rho)} \right] < \varepsilon$$

Lemma 5.4.2 is an easy corollary of Borell's reverse hypercontractive inequality [12]; we include a short proof in appendix 5.6. Applications of Borell's result to strong isoperimetric inequalities appear in [35], and we follow the proofs of similar lemmas there.

5.4.2 Definitions

For a matching-cover M and a distribution D over $\mathbb{R}^*$, let $M(D)$ be the random graph $M(u_1, \ldots, u_r)$ where $u_1, \ldots, u_r \sim D$. For a random graph $\mathcal{G}$ and sets $S, T \subseteq [n]$, let $\mu_\mathcal{G}(S, T)$ be the expected number of edges from S to T in $\mathcal{G}$. We say S is γ-*connected* to T in $\mathcal{G}$ if $\mu_\mathcal{G}(S, T) \geq \gamma$. For singleton sets, we omit braces and write $\mu_\mathcal{G}(x, y)$ for the probability that the edge (x, y) is in $\mathcal{G}$.

Two sets that will be useful are,

$$\begin{aligned}\mathsf{Ball}[x;\ell] &= \{y : \|\mathbf{v}_x - \mathbf{v}_y\| \le \ell\}\\ \mathsf{Stretch}[x,\sigma,u] &= \{y : (\mathbf{v}_y - \mathbf{v}_x)\cdot u \ge \sigma\}\end{aligned}$$

We will also work with collections of distributions $D = \{D(u)\}$ over $\mathbb{R}^*$ parameterized by u. Such a collection is itself associated with the distribution induced by sampling a standard normal u and then sampling from $D(u)$.

Definition 5.4.3. *Let D be a distribution collection. We say a vertex x is $(\sigma,\delta,\gamma,\ell)$-**covered** in $M(D)$ if for least δ of u, x is γ-connected to $\mathsf{Stretch}[x,\sigma,u]\cap\mathsf{Ball}[x;\ell]$ in $M(D(u))$.*

5.4.3 Cover Lemmas

Our goal is to exhibit a distribution D such that many vertices x are well-connected to $X\backslash\mathsf{Ball}[x;\sqrt{L}]$ in $M(D)$. To do so, we inductively construct particular distribution collections D^r such that many vertices x are either $e^{-O(Rr)}$-connected to $X\setminus\mathsf{Ball}[x;\sqrt{L}]$ in $M(D^r)$, or else are $(\Omega(r),\Omega(1),e^{-O(rR)},\sqrt{L})$-covered by $M(D^r)$.

We begin with a trivial bound on how much points can be covered.

Lemma 5.4.4. *For $\ell,\gamma,\delta>0$ and arbitrary M,D, no vertex is $(\ell\sqrt{2\log(n/\delta)},\delta,\gamma,\ell)$-covered by $M(D)$.*

Proof. For any $y\in\mathsf{Ball}[x;\ell]$, the probability that $(\mathbf{v}_y-\mathbf{v}_x)\cdot u\ge\beta$ is at most $\exp(-\beta^2/2\ell^2)\le\delta/n$ for $\beta=\ell\sqrt{2\log(n/\delta)}$. It follows that the probability that $\mathsf{Stretch}[x,\ell\sqrt{2\log(n/\delta)},u]\cap\mathsf{Ball}[x;\ell]$ is non-empty is at most $(n-1)\delta/n<\delta$. □

The next lemma says that if a vertex x is connected by D' to a set S of vertices that are covered by D, then x is covered by the concatenation of D' and D.

Lemma 5.4.5. *Let S be a set of vertices such that each $y\in S$ is $(\sigma,\delta,\gamma,\ell)$-covered by $M(D)$. Let x be a vertex with $\mu_{M(D')}(x,S\cap\mathsf{Ball}[x;\ell'])\ge\gamma'$. Then, x is $(\sigma-\sqrt{2\ell'\log(2/\delta)},\delta/4,\gamma\gamma'\delta/4,\ell+\ell')$-covered by $D''(u)=D',D(u)$.*

Proof. Let Γ be the distribution of x's out-neighbor in $M(D')$, conditioned on $S\cap\mathsf{Ball}[x;\ell']$. For each $y\in S$, let A_y be the set of u for which y is γ-connected to $\mathsf{Stretch}[y,\sigma,u]\cap\mathsf{Ball}[y;\ell]$ in $M(D(u))$.

For any fixed $y\in\Gamma$, the quantity $(\mathbf{v}_x-\mathbf{v}_y)\cdot u$ is normal with mean zero and variance $\|\mathbf{v}_y-\mathbf{v}_x\|^2\le\ell'^2$, so the probability (over u) that $y\in\mathsf{Stretch}[x,-\beta,u]$ is at least $1-\exp(-\beta^2/2\ell'^2)\ge 1-\delta/2$ for $\beta=\sqrt{2\ell'\log(2/\delta)}$. Then, for at least $\delta/2$ of u, we have $y\in\mathsf{Stretch}[x,-\beta,u]$ and $u\in A_y$. By averaging, for at least $\delta/4$ of u, at least $\delta/4$ of $y\sim\Gamma$ have $y\in\mathsf{Stretch}[x,-\beta,u]$ and $u\in A_y$. It follows that x is $(\sigma-\beta,\delta/4,\gamma\gamma'\delta/4,\ell+\ell')$-covered by D''. □

Our next lemma is the main chaining step.

Lemma 5.4.6. *Let M be a $(\sigma_0, \cdot)$-matching-cover, T be a set of vertices that are $(\sigma, 1-\delta/2, \gamma, \infty)$-covered in $M(D)$, and S a set of vertices that is $\delta|T|$-connected to T in M. Then, at least $\delta|T|/2$ vertices $x \in S$ are $(\sigma+\sigma_0, \delta|T|/4|S|, \gamma, \infty)$-covered by $D'(u) = u, D(u)$.*

Proof. Let M' be the subgraph of M consisting only of edges from S to T; by assumption the total degree in M' is at least $\delta|T|$. Further remove any edge $(x, y) \in M'(u)$ where $\mu_{M(D(u))}(y, \mathsf{Stretch}[y, \sigma, u]) < \gamma$. The total in-degree remaining is at least $\delta|T|/2$, so there is a set $S' \subseteq S$ of at least $\delta|T|/2$ vertices that have out-degree at least $\delta|T|/4|S|$. Finally, note that if $(x, y) \in M'(u)$, then $\mathsf{Stretch}[y, \sigma, u, \infty] \subseteq \mathsf{Stretch}[x, \sigma+\sigma_0, u, \infty]$, so each $x \in S'$ is $(\sigma+\sigma_0, \delta|T|/4|S|, \gamma, \infty)$-covered by D'. □

To apply lemma 5.4.6, we need to establish covers with δ very close to 1. Consider taking a collection D and then *smoothing* it by replacing $D(u)$ with the average of $D(\hat{u})$ for nearby $\hat{u}$. The next lemma shows that doing so boosts δ to nearly 1, in exchange for a loss in σ and γ.

Lemma 5.4.7. *Let x be $(\sigma, \delta, \gamma, \ell)$-covered by D. Then, x is $(\rho\sigma - 4\ell\sqrt{\log(2/\delta)}, 1-2\delta, \delta^{2/(1-\rho)}\gamma/4, \ell)$-covered by $D'(u) = D(\hat{u})$ where $\hat{u} \sim_\rho u$.*

Proof. Let A be the set of $\hat{u}$ for which x is γ-connected to $\mathsf{Stretch}[x, \sigma, \hat{u}] \cap \mathsf{Ball}[x; \ell]$ in $M(D(\hat{u}))$. For each $\hat{u}$, let $\Gamma(\hat{u})$ be the distribution of x's out-neighbor in $M(D(\hat{u}))$, conditioned on $\mathsf{Stretch}[x, \sigma, \hat{u}] \cap \mathsf{Ball}[x; \ell]$.

For any $\hat{u}$ and $y \in \Gamma(\hat{u})$, the quantity $(\mathbf{v}_y - \mathbf{v}_x) \cdot u$ is normal with mean $\rho(\mathbf{v}_y - \mathbf{v}_x) \cdot \hat{u} \geq \rho\sigma$ and variance $(1-\rho^2)\|\mathbf{v}_y - \mathbf{v}_x\|^2 \leq 2(1-\rho)\ell^2$; it follows that $y \in \mathsf{Stretch}[x, \rho\sigma - \beta, u]$ with probability at least $1 - \exp(-\beta^2/4(1-\rho)\ell^2) \geq 1 - (\delta/2)^{4/(1-\rho)}$ for $\beta = 4\ell\sqrt{\log(2/\delta)}$ over u. By averaging, for at least $1-\delta$ u, for at least $1 - (2/\delta)(\delta/2)^{4/(1-\rho)}$ $\hat{u} \sim_\rho u$, we have $\mathbf{Pr}[\Gamma(\hat{u}) \in \mathsf{Stretch}[x, \rho\sigma - \beta, u]] \geq 1/2$. Call such pairs $(u, \hat{u})$ good.

Applying lemma 5.4.2 to A, for at least $1-\delta$ u, we have $\mathbf{Pr}[\hat{u} \in A] \geq \delta^{2/(1-\rho)}$. All together, for at least $1-2\delta$ u, with probability at least $\delta^{2/(1-\rho)} - (2/\delta)(\delta/2)^{4/(1-\rho)}$ we have both $\hat{u} \in A$ and $(u, \hat{u})$ good. In that case, $\mu_{M(D(\hat{u}))}(x, \mathsf{Stretch}[x, \rho\sigma - \beta, u] \cap \mathsf{Ball}[x; \ell]) \geq \gamma/2$. The lemma follows by noting $(2/\delta)(\delta/2)^{4/(1-\rho)} \leq \delta^{2/(1-\rho)}/2$. □

Combining the previous results, we prove the main inductive lemma.

Lemma 5.4.8. *Let M be a (σ, δ)-uniform-matching-cover of X where $\delta \leq 1/4$. Let $\ell \leq \sigma/2^7\sqrt{\log(1/\delta)}$ and $K \geq 1$. Then, one of the following must occur.*

1. *There are distribution collections $D^0, \ldots, D^K$, such that for every $k \leq K$, at least $\delta^{6k}|X|$ vertices are $(k\sigma/4, \delta^8, \delta^{59Kk}, \ell)$-covered in $M(D^k)$.*
2. *There is a distribution D^* such that at least $\delta^{6K}|X|$ vertices x are δ^{59K^2}-connected to $X \setminus B[x; \ell]$ in $M(D^*)$. Furthermore, D^* is a shuffling of $\mathcal{N}^k_{1-1/K}$ with $\mathcal{N}^{k'}_0$ for some $k \leq K$ and $k' \leq 6K$.*

Proof. For $k = 0$, every $x \in X$ is $(0, 1, 1, 0)$-covered by $D^0(u) = ()$, the empty list.

Assuming case 1 holds for some $0 \leq k < K$, let T_0 be those vertices that are $(k\sigma/4, \delta^8, \gamma, \ell)$-covered by D^k. We begin by finding a set S that is well-connected to T_0. Since at least $\delta|T_0|$

in-degree enters T_0 in M, by averaging either at least $\delta^{-1}|T_0|$ vertices have at least $\delta^2|T_0|/|X|$ out-degree into T_0 or else at least $\delta|T_0|$ vertices have at least δ^3 out-degree into T_0. In the former case, call that set T_1 and repeat, yielding sets $T_0, T_1, \ldots, T_t$ where each $y \in T_s$ has at least $\delta^2|T_{s-1}|/|X|$ out-degree into T_{s-1}. Let S be those vertices with out-degree at least δ^3 into T_t, so that $\delta|T_t| \le |S| \le \delta^{-1}|T_t|$. Let $D' = \mathcal{N}_0^t$; by construction, each $y \in T_t$ has

$$\begin{aligned}\mu_{M(D')}(y, T_0) &\ge \prod_{s=0}^{t-1} \delta^2 |T_s|/|X| \\ &\ge \delta^{2t-t(t-1)/2}(|T_0|/|X|)^t \\ &\ge \delta^{3+6kt}\end{aligned}$$

Assuming case 2 does not hold by setting $D^* = D'$, there is a set $T \subseteq T_t$ of size at least $(1-\delta^5)|T_t|$ such that each $y \in T$ has $\mu_{M(D')}(y, T_0 \cap \mathsf{Ball}[y;\ell]) \ge \delta^{3+6kt}/2 =: \gamma'$. It follows that at least $\delta^3|S| - \delta^5|T_t| \ge \delta^5|T_t|$ out-degree from S enters T.

Lemma 5.4.5 implies each $y \in T$ is $((k-1)\sigma/4, \delta^9, \gamma'', 2\ell)$-covered by $D''(u) = D', D^k(u)$ where $\gamma'' = \gamma\gamma'\delta^9$ (we replace factors of $1/4$ with δ). Setting $\rho = 1 - 1/K$, lemma 5.4.7 implies each $y \in T$ is $((k-3)\sigma/4, 1-2\delta^9, \gamma''', 2\ell)$-covered by $D'''(u) = D(\hat{u})$ for $\hat{u} \sim_{1-1/K} u$ where $\gamma''' = \delta^{18K+1}\gamma''$. Finally, since $\delta^5|T_t|/4|S| \ge \delta^7$, lemma 5.4.6 implies at least $\delta^5|T_t|/2$ vertices in S are $((k+1)\sigma/4, \delta^7, \gamma''', \infty)$-covered by $D^{k+1}(u) = u, D'''(u)$, where

$$\gamma''' = \delta^{18K+1+3+6kt}\gamma/2 \ge 2\delta^{23K+6Kt}\gamma$$

Assuming case 2 does not hold for $D^* = D^{k+1}$, at least $\delta^5|T_t|/4 \ge \delta^{6(k+1)-t}|X|$ vertices in S are $((k+1)\sigma/4, \delta^8, \delta^{23K+6Kt}\gamma, \ell)$-covered by D_{k+1}.

Observe that at step k, this argument yields $\gamma = \delta^{23Kk+6Kt_k}$, where t_k is the total number of expanding steps taken by step k. Since $t_k \le 6k$, we have $\gamma \ge \delta^{59Kk}$. Furthermore, D_k is a shuffling of $\mathcal{N}_{1-1/K}^k$ with $\mathcal{N}_0^{t_k}$. □

To complete the proof of theorem 5.2.3, recall M is a $(\sigma, \delta/4)$-uniform-matching-cover of X. Let $1 \le R \le \log(n)/\log(1/\delta))$. For $R < 7$, lemma 5.4.8 implies a typical edge in M has length $\Omega(\sigma/\sqrt{\log(n/\delta)}) = \Omega(R\sigma/\sqrt{\log n})$ since $\log(n \ge \log(1/\delta))$ by assumption. That is, setting $D = \texttt{Sample}(1, 0)$ suffices.

For $R \ge 7$, set $K = \lfloor R/7 \rfloor$ and $\ell = R\sigma/2^{12}\sqrt{\log(n)}$, so that ℓ satisfies lemma 5.4.8. Lemma 5.4.4 implies case 1 of lemma 5.4.8 can not hold for K, so case 2 must hold. That is, setting $D = \texttt{Sample}(R, 1-1/K)$ suffices.

5.4.4 Using ± 1 Coins

One might be concerned with issues of precision required for sampling Gaussians. Fortunately, it suffices to approximate them by sampling $\mathbf{w} \in \{\pm 1\}^k$ and returning $\frac{1}{\sqrt{k}}\sum_{i=1}^k \mathbf{w}_i$ for $k = O(\log n)$.

Lemma 5.4.9. *Suppose that instead of a random Gaussian u, we sample a uniform random ± 1 matrix $U \in \mathbb{R}^{d\times k}$ and set $u = U\mathbf{1}$, where $\mathbf{1} \in \mathbb{R}^k$ has $\mathbf{1}_j = 1/\sqrt{k}$ for all $j \le k$. To*

sample a ρ-correlated copy $\hat{u}$, we sample $\hat{U} \in \mathbb{R}^{d\times k}$ as a ρ-correlated copy of U (i.e., each $\hat{U}_{ij} = U_{ij}$ with probability ρ or a random ± 1 with probability $1-\rho$) and set $\hat{u} = \hat{U}\mathbf{1}$. Then, for $k = O(R^2 \log(1/\delta)) = O(\log n)$, theorem 5.2.3 still holds.

The proof of lemma 5.4.9 is straightforward. Lemmas 5.3.3, 5.4.4, 5.4.5 all still hold with similar constants even for $k = 1$ (see e.g. [1]), so the only issue is lemma 5.4.7. For the latter, lemma 5.4.2 also holds for ρ-correlated ± 1 variables, so the only change needed is in bounding $(\mathbf{v}_x - \mathbf{v}_y) \cdot (u - \hat{u})$, which is easily done for $k = O(R^2 \log(1/\delta)) = O(\log(n))$ using Bernstein's inequality(see e.g. [13]). For completeness, we include the details in appendix 5.7.

5.5 Lower-bound for the Cut-Matching Game

Khandekar, Rao, and Vazirani proposed a primal-dual framework based on the following two-player game game, which proceeds for T rounds[28]. On each round, the cut player chooses a bisection $(S^t, \overline{S^t})$ of the vertices, and the matching player responds with a perfect matching M^t pairing each $x \in S^t$ with some $y \in \overline{S^t}$ The payoff to the cut player is $h(H^T)$, where $H^t = M^1 + \cdots + M^t$. Thus on round t, the cut player aims to choose a cut so that *any* matching response M^t will increase the expansion of H^t.

To see the connection to SPARSEST CUT, suppose the cut player has a strategy that guarantees $h(H^T) \geq T/\kappa$, and consider a matching player that plays as follows. When given a bisection $(S, \overline{S})$, the matching player connects a source s to all $x \in S$ with edges of unit capacity and a sink t to all $y \in \overline{S}$. A simple lemma similar to lemma 5.3.4 implies that if the min-cut is at most $n/2$, then it has expansion at most one. Otherwise, the added edges are saturated, and assuming all edges have integral capacities, the flow can be pseudo-decomposed into a matching; the matching player responds with that matching. Then, after T rounds, we have either found a cut of expansion one or else routed H^T in G with congestion T. Assuming the cut-player forced $h(H^T) \geq T/\kappa$, scaling down by T yields a feasible flow routing a graph of expansion $1/\kappa$, yielding a κ-approximation.

The following theorems appear in [40].

Theorem 5.5.1 ([40])**.** *The cut player has an (efficient) strategy to ensure,*

$$\exp\left(-\lambda_2(\mathcal{L}_{H^t})\right) \leq n \exp\left(\frac{-t}{O(\log n)}\right)$$

In particular, after $T = O(\log^2(n))$, the cut player can ensure $\lambda_2(\mathcal{L}_{H^T}) \geq \Omega(\log n)$, yielding an $O(\log n)$ factor approximation using $O(\log^2 n)$ max-flows.

Theorem 5.5.2 ([40])**.** *The matching player can ensure*

$$h(H^t) \leq O\left(\frac{1}{\sqrt{\log n}}\right) \cdot t$$

We prove the following.

Theorem 5.5.3. *The matching player can ensure*

$$\lambda_2(\mathcal{L}_{H^t}) \leq O\left(\frac{\log\log n}{\log n}\right) \cdot t$$

Theorem 5.5.3 does not entirely eliminate the possibility of achieving a better approximation in the cut-matching game, and indeed it is known among experts that there *exists* an (inefficient) strategy for the cut-player to ensure $\exp(-h(H^t)) \leq n\exp(-t/O(\sqrt{\log(n)})$[39]. However, theorem 5.5.3 says that doing so will require certifying expansion via something stronger than $\lambda_2(\mathcal{L}_{H^t})$. For example, one could route another expander flow H' in H^T and certify $h(H^T) \geq \lambda_2(\mathcal{L}_{H'})/2$. Such an approach seems somewhat awkward though, as any such flow might as well have been routed in G directly.

In theorem 5.5.2, the matching player arbitrarily identifies the vertices of G with a hypercube, and tries to keep the dimension cuts sparse. In particular, it is shown that for any bisection $(S, \overline{S})$, there must always exist a matching that raises the expansion of the average dimension cut by at most $O(1/\sqrt{d})$.

To prove theorem 5.5.3, we identify the vertices of G arbitrarily with a dense set of points $\mathbf{v}_1, \ldots, \mathbf{v}_n \in \mathbb{R}^d$ on the sphere S^{d-1}, where $d = \Omega(\log(n)/\log\log(n))$. Letting $\mathbf{w}_1, \ldots, \mathbf{w}_d \in \mathbb{R}^n$ be the column vectors of the $n \times d$ matrix with row vectors $\{\mathbf{v}_x\}$, we show that for any bisection $(S, \overline{S})$ there must be a matching that raises the average Rayleigh quotient $\frac{\mathbf{w}_i^T \mathcal{L}_H \mathbf{w}_i}{\mathbf{w}_i^T \mathbf{w}_i}$ by at most $O(1/d)$.

The following lemma is an easy generalization of one in [40].

Lemma 5.5.4. *Let $\mathbf{v}_1, \ldots, \mathbf{v}^n \in R^d$, and let $\mathbf{w}_1, \ldots, \mathbf{w}_d \in \mathbb{R}^n$ be defined by $\mathbf{w}_{i,x} = \mathbf{v}_{x,i}$. Define,*

$$\psi(t) = \frac{1}{d}\sum_{i=1}^{d} \frac{\mathbf{w}_i^T \mathcal{L}_{H^t} \mathbf{w}_i}{\mathbf{w}_i^T \mathbf{w}_i}$$

If all $\|\mathbf{w}_i\|^2 \geq L > 0$, then,

$$\psi(t) - \psi(t-1) \leq \frac{1}{dL} \sum_{xy \in M^t} \|\mathbf{v}_x - \mathbf{v}_y\|^2$$

Proof.

$$\begin{aligned}
\psi(t) - \psi(t-1) &= \frac{1}{d}\sum_{i=1}^{d} \frac{\mathbf{w}_i^T \mathcal{L}_{M^t} \mathbf{w}_i}{\mathbf{w}_i^T \mathbf{w}_i} \\
&\leq \frac{1}{dL}\sum_{i=1}^{d} \sum_{xy \in M^t} (\mathbf{w}_{i,x} - \mathbf{w}_{i,y})^2 \\
&= \frac{1}{dL} \sum_{xy \in M^t} \|\mathbf{v}_x - \mathbf{v}_y\|^2 \quad \square
\end{aligned}$$

If $\mathbf{w}_1, \ldots, \mathbf{w}_d$ are as in lemma 5.5.4 and all orthogonal to the all-1s vector, then $\lambda_2(\mathcal{L}_{H^t}) \leq \psi(t)$; the orthogonality condition is equivalent to $\sum_x \mathbf{v}_x = 0$. Having fixed such an embedding, when presented with a bisection $(S, \overline{S})$, the matching player aims to match points so as to minimize the average distance between matched points. The analysis of [40] shows that for the hypercube embedding $\{-1, 1\}^d$, one can obtain $\psi(t) - \psi(t-1) \leq O(1/\sqrt{d})$. The analysis is not constructive; rather, they use the vertex isoperimetry of the hypercube to establish an upper bound on the value of the matching problem's dual LP, and then conclude a matching achieving that bound exists by strong duality. Their argument also depends on the fact that for the hypercube embedding, the squared distances $\|\mathbf{v}_x - \mathbf{v}_y\|^2$ form a metric.

In fact, the metric assumption is not needed, and there is also no need to apply LP duality. We give a simple proof that large vertex isoperimetry of the embedding implies the simple greedy strategy of iteratively matching closest points works.

Lemma 5.5.5. *Let $\mathbf{v}_1, \ldots, \mathbf{v}_n \in \mathbb{R}^d$ be a set of points such that, for any $S \subseteq \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ with $|S| \leq n/2$, $|\mathsf{Ball}[S; \sqrt{r}]| \geq (1 + \Omega(1))|S|$, Then, the greedy strategy produces M with,*

$$\sum_{xy \in M} \|\mathbf{v}_x - \mathbf{v}_y\|^2 \leq O(nr)$$

Proof. Starting with $S, \overline{S}$, we pick $x \in S, y \in \overline{S}$ minimizing $\|\mathbf{v}_x - \mathbf{v}_y|^2$, match them, and then remove them. Repeated application of the the isoperimetric condition implies that, for all S with $|S| \leq n/2$, $|\mathsf{Ball}[S; t\sqrt{r}]| \geq \min\{1 + n/2, (1 + \Omega(1))^t |S|\}$. It follows that if two sets A, B have size s, there must be $x \in A$, $y \in B$ with $\|\mathbf{v}_x - \mathbf{v}_y\| \leq 2t\sqrt{r}$ for $t = \lceil \log_{(1+\Omega(1))}(n/2s) \rceil + 1 \leq 2 + O(1) \log(n/2s)$. Then, the total cost of the greedy solution is at most,

$$\begin{aligned}
\sum_{xy \in M} \|\mathbf{v}_x - \mathbf{v}_y\|^2 &\leq O\left(\sum_{s=1}^{n/2} (1 + \log(n/2s))^2 \cdot r \right) \\
&\leq O\left(n + \int_0^{n/2} \log^2(n/2s)\ \mathrm{d}s \right) r \\
&\leq O(nr) \quad \square
\end{aligned}$$

For the case of theorem 5.5.2, let $\mathbf{v}_x \in \{-1/\sqrt{d}, 1/\sqrt{d}\}^d$ be the hypercube embedding and take $L = n/d$ in lemma 5.5.4. The vertex isoperimetry of the hypercube implies $r = O(1/\sqrt{d})$ in lemma 5.5.5, yielding a strategy to ensure $\psi(t) \leq O(nr/dL) \cdot t = O(1/\sqrt{d}) \cdot t$.

To prove theorem 5.5.3, we choose $\mathbf{v}_x$ as per the following lemma, and take $L = \Omega(n/d)$, $r = O(1/d)$, yielding a strategy to ensure $\psi(t) \leq O(nr/dL) \cdot t = O(1/d) \cdot t$.

Lemma 5.5.6. *For every d, there exists a set of $n = O(\sqrt{d})^d$ points $\mathbf{v}_1, \ldots, \mathbf{v}_n \in S^{d-1}$ such that $\sum_{i=1}^n \mathbf{v}_i = 0$, every $i \leq d$ has $\sum_{x=1}^n \mathbf{v}_{x,i}^2 = \Omega(n/d)$, and for every $S \subseteq [n]$ with $|S| \leq n/2$, $|\mathsf{Ball}[S; O(1/\sqrt{d})]| \geq (1 + \Omega(1))|S|$.*

The proof of lemma 5.5.6 is a straightforward application of a construction of Feige and Schechtman[23], which we include in appendix 5.8

5.6 Proof of Lemma 5.4.2

For $f : \mathbb{R}^d \to \mathbb{R}_{\geq 0}$, let $\|f\|_p = \mathbf{E}[f^p]^{1/p}$, where the expectation is over the multivariate standard normal distribution. For $x \in \mathbb{R}^d$, we write $y \sim_\rho x$ for a ρ-correlated copy of u. The *Ornstein-Uhlenback operator* is defined by,

$$T_\rho f(x) = \mathbf{E}_{y \sim_\rho x}[f(y)]$$

Theorem 5.6.1 (Borell[12])**.** *Let $f : \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ and $-\infty < q \leq p \leq 1$. If $0 \leq \rho^2 \leq (1-p)/(1-q)$, then*

$$\|T_\rho f\|_q \geq \|f\|_p \qquad \text{for } 0 \leq \rho^2 \leq (1-p)/(1-q)$$

By a change of variables, lemma 5.4.2 is equivalent to,

$$\mathbf{Pr}_u[\mathbf{Pr}_{\hat{u}}[\hat{u} \in A] < \tau] < \frac{\tau^{1-\rho}}{\delta}$$

Let f indicate A, and set $p = 1-\rho, q = 1 - 1/\rho$. Note $q < 0 < p \leq 1$ satisfy theorem 5.6.1, so

$$\|T_\rho f\|_q \geq \|f\|_p = \delta^{1/p}$$

Then, $\mathbf{Pr}_{\hat{u} \sim_\rho u}[\hat{u} \in A] = T_\rho f(u)$, and we have,

$$\begin{aligned}
\mathbf{Pr}[T_\rho f < \tau] &= \mathbf{Pr}[(T_\rho f)^q > \tau^q] \\
&< \|(T_\rho f)\|_q^q \tau^{-q} \\
&\leq \delta^{q/p} \tau^{-q} \\
&= \left(\frac{\tau^{1-\rho}}{\delta}\right)^{1/\rho}
\end{aligned}$$

For $\tau^{1-\rho}/\delta \leq 1$, raising the last line to ρ can't decrease its value. In the other case, the result is trivial.

5.7 Proof of Lemma 5.4.9

Lemma 5.3.3 only uses the fact that for a vector $\mathbf{v}$ and standard normal u, $(u \cdot \mathbf{v})^2 \geq \Omega(\|\mathbf{v}\|^2)$ with probability $\Omega(1)$. That property still holds.

Lemma 5.7.1. *Let $U \in \mathbb{R}^{d \times k}$ be a uniform random ± 1 matrix, and let $\mathbf{v} \in \mathbb{R}^d$ be a vector. Then,*

$$\mathbf{Pr}\left[(\mathbf{v} \cdot U\mathbf{1})^2 \geq \|\mathbf{v}\|^2/4\right] \geq 1/5$$

Proof. It suffices to consider a unit vector $\mathbf{v}$. Let $Z = \mathbf{v} \cdot U\mathbf{1}$. Then,

$$\mathbf{E}[Z^2] = \mathbf{E}\left[\left(\sum_{i \leq d, j \leq k} \mathbf{v}_i \frac{U_{ij}}{\sqrt{k}}\right)^2\right] = \sum_{i_1, i_2, j_1, j_2} \mathbf{v}_{i_1}\mathbf{v}_{i_2} \frac{\mathbf{E}\left[U_{i_1 j_1} U_{i_2 j_2}\right]}{k} = \sum_{i,j} \mathbf{v}_i^2/k = \|\mathbf{v}\|^2 = 1$$

$$\mathbf{E}[Z^4] = \sum_{i_1,\ldots,i_4,j_1,\ldots,j_4} \mathbf{v}_{i_1}\cdots\mathbf{v}_{i_4}\frac{\mathbf{E}\left[U_{i_1j_1}\cdots U_{i_4j_4}\right]}{k^2} \leq 3\sum_{i_1,j_1,i_2,j_2}\mathbf{v}_{i_1}^2\mathbf{v}_{i_2}^2/k^2 = 3\|\mathbf{v}\|^4 = 3$$

Then, for any $t \leq 1/\lambda$, we have,

$$\mathbf{Pr}\left[Z^2 < \lambda\right] \leq \mathbf{Pr}\left[\left(1-tZ^2\right)^2 > (1-t\lambda)^2\right] < \frac{\mathbf{E}\left[\left(1-tZ^2\right)^2\right]}{(1-t\lambda)^2} = \frac{1-2t\mathbf{E}[Z^2]+t^2\mathbf{E}[Z^4]}{(1-t\lambda)^2}$$

Taking $t = \lambda = 1/4$ yields,

$$\mathbf{Pr}\left[Z^2 < 1/4\right] < \frac{1-1/2+3/16}{(15/16)^2} < 1/5$$

□

The remaining lemmas require a Gaussian-like bound on stretch; for that, we'll use the following theorem.

Theorem 5.7.2 (Bernstein's Inequality)**.** *Let $X_1,\ldots,X_n$ be independent random variables with $\mathbf{E}[X_i] = 0$ and $X_i \leq 1$. Let $\sigma^2 = \sum_{i=1}^n \mathbf{E}[X_i^2]$. Then, for any $t > 0$,*

$$\mathbf{Pr}\left[\sum_{i=1}^n X_i > t\sigma\right] \leq \exp\left(\frac{-t^2}{2+t/3\sigma}\right)$$

The next lemma says that if k is large enough, we can obtain Gaussian-like bounds on stretch. Note that when $\rho = 0$ much better bounds are possible, in that even $k = 1$ works (see [1]).

Lemma 5.7.3. *Let $U \in \mathbb{R}^{d\times k}$ be an arbitrary ± 1 matrix, and let $\hat{U} \sim_\rho U$ be a ρ-correlated copy of U. Then, for any vector $\mathbf{v}$, and any $0 < t \leq \sqrt{k(1-\rho^2)}$,*

$$\mathbf{Pr}\left[\mathbf{v}\cdot\hat{U}\mathbf{1} > \rho(\mathbf{v}\cdot U\mathbf{1}) + t\sqrt{1-\rho^2}\|\mathbf{v}\|\right] \leq e^{-t^2/3}$$

$$\mathbf{Pr}\left[\mathbf{v}\cdot\hat{U}\mathbf{1} < \rho(\mathbf{v}\cdot U\mathbf{1}) - t\sqrt{1-\rho^2}\|\mathbf{v}\|\right] \leq e^{-t^2/3}$$

Proof. It suffices to consider a unit vector $\mathbf{v}$. For each $i \leq d$, $j \leq k$, let $Z_{ij} = \mathbf{v}_i(\hat{u}_{ij} - \rho U_{ij})/2$, so that we have $\mathbf{E}[Z_{ij}] = 0$, $|Z_{ij}| \leq 1$, and $\mathbf{E}[Z_{ij}^2] = (1-\rho^2)\mathbf{v}_i^2/4$. Note that,

$$\mathbf{v}\cdot\hat{U}\mathbf{1} = \rho(\mathbf{v}\cdot\hat{U}\mathbf{1}) + \frac{2}{\sqrt{k}}\sum_{i\leq d, j\leq k} Z_{ij}$$

Applying theorem 5.7.2 with $\sigma^2 = k(1-\rho^2)\|\mathbf{v}\|^2/4 = k(1-\rho^2)/4$, we have

$$\mathbf{Pr}\left[\sum_{i,j} Z_{ij} > t\sigma\right] \leq \exp\left(\frac{-t^2}{2+t/3\sigma}\right) \leq e^{-t^2/3}$$

proving the first part. The second part follows by applying the same argument to $-Z_{ij}$. □

For lemmas 5.4.4 and 5.4.5, we use $\rho = 0$ and $t = O(\sqrt{\log(1/\delta)})$, so $k = O(\log(1/\delta))$ suffices. For lemma 5.4.7, we use $\rho = 1 - 1/K$ and $t = O(\sqrt{K\log(1/\delta)})$, so $k = O(K^2\log(1/\delta))$ suffices. Also, lemma 5.4.2 holds for the uniform measure on the hypercube, as Borell's theorem also holds for $f : \{-1,+1\}^n \to \mathbb{R}_{\geq 0}$.

5.8 Proof of Lemma 5.5.6

Lemma 5.8.1 (Feige, Schechtman [23])**.** *For each $0 < \gamma < \pi/2$, the sphere S^{d-1} can be partitioned into $n = (O(1)/\gamma)^d$ equal volume cells, each of diameter at most γ.*

Apply lemma 5.8.1 with $\gamma = 1/\sqrt{d}$, yielding cells $C_1, \ldots, C_n$ with $n = \exp(O(d \log d))$. Let V be a set of n arbitrary points, each from a distinct cell; for convenience, let us choose V so that $V \cap (-V) = \emptyset$.

Claim 5.8.2. *If $A \subseteq S^{d-1}$ has $\mu(A) \geq \alpha$, then $|\mathsf{Ball}(A;\gamma) \cap V| \geq \alpha n$; if $A \subseteq V$ has $|A| \geq \alpha n$, then $\mu(\mathsf{Ball}(A;\gamma)) \geq \alpha$.*

Proof. For the first direction, if $\mu(A) \geq \alpha$, A intersects at least αn of the cells, so $\mathsf{Ball}[A;\gamma]$ contains at least αn cells, and hence at least αn elements of V. For the second, if $A \subseteq V$ has size at least αn, then $\mathsf{Ball}[A;\gamma]$ contains at least αn cells, so $\mu(\mathsf{Ball}[A;\gamma]) \geq \alpha$. □

Claim 5.8.3. *For every $i \leq d$, $\sum_{\mathbf{v} \in V} \mathbf{v}_i^2 \geq \Omega(n/d)$.*

Proof. Let $A = \{x \in S^{d-1} : x_i \geq \sqrt{2/d}\}$. By bounds on the measure of spherical caps (see e.g. [34]), $\mu(A) \geq 1/12$. Using claim 5.8.2, $|\mathsf{Ball}(A;\gamma) \cap V| \geq (1/12)n$. Then, since $x_i \geq \sqrt{2/d} - \gamma \geq \sqrt{1/8d}$ for all $x \in \mathsf{Ball}(A;\gamma)$, we have $\sum_{\mathbf{v} \in V} \mathbf{v}_i^2 \geq (1/12)n(1/8d) \geq \Omega(n/d)$. □

Claim 5.8.4. *For every $A \subseteq V$ with $|A| \leq n/2$, $|\mathsf{Ball}[A; O(1/\sqrt{d})] \cap X| \geq (1 + 1/12)|A|$.*

Proof. Let $A \subseteq V$ have $|A| \leq n/2$, and set $A_1 = \mathsf{Ball}[A;\gamma], A_2 = \mathsf{Ball}[A_1; 4/\sqrt{d}], A_3 = \mathsf{Ball}[A_2;\gamma]$; the goal is to show $|A_3 \cap V| \geq (1 + 1/12)|A|$. Note claim 5.8.2 ensures $\mu(A_1) \geq |A|/n$. If $\mu(A_1) \geq (1 + 1/12)/2$, then $\mu(A_2) \geq (1 + 1/12)|A|/n$. Otherwise, by the isoperimetric inequality on the sphere (see e.g. [34]), $\mu(A_2) \geq (1 + 1/12)\mu(A_1) \geq (1 + 1/12)|A|/n$. By by claim 5.8.2, $|A_3 \cap V| \geq (1 + 1/12)|A|$. □

Now to prove lemma 5.5.6, we let $V' = V \cup -V$. Clearly $\sum_{\mathbf{v} \in V'} \mathbf{v} = 0$, and claim 5.8.3 still applies to V', so it remains only to argue claim 5.8.4 still holds for V'. Let $A \subseteq V'$ have $|A| \leq |V'|/2 = n$. Let $A = A_+ \cup A_-$ where $A_+ \subseteq V$ and $A_- \subseteq -V$, and suppose $|A_+| \leq |A_-|$ (in the other case, an analogous argument applies). We consider two cases. First, if $|A_+| \leq |A_-|/2$, then $\mu(\mathsf{Ball}[A_-;\gamma]) \geq |A_-|/n$, implying $|\mathsf{Ball}[A_-; 2\gamma] \cap V| \geq |A_-|/n$. Therefore, $|\mathsf{Ball}[A; 2\gamma] \cap V'| \geq 2|A_-| \geq (4/3)|A|$. Otherwise, $|A_+| \leq n/2$ and $|A_+| \geq |A|/3$, so claim 5.8.4 implies $|\mathsf{Ball}[A_+; O(1/\sqrt{d})] \cap V| \geq (1 + 1/12)|A_+|$. Therefore, $|\mathsf{Ball}[A; O(1/\sqrt{d})] \cap V'| \geq (1 + 1/12)|A_+| + |A_-| \geq (1 + 1/36)|A|$.

Bibliography

[1] Dimitris Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *J. Comput. Syst. Sci.*, 66(4):671–687, 2003.

[2] Ravindra K Ahuja, Andrew V Goldberg, James B Orlin, and Robert E Tarjan. Finding minimum-cost flows by double scaling. *Mathematical programming*, 53(1-3):243–266, 1992.

[3] Noga Alon and V. D. Milman. λ_1, isoperimetric inequalities for graphs, and superconcentrators. *J. Comb. Theory, Ser. B*, 38(1):73–88, 1985.

[4] Sanjeev Arora, Elad Hazan, and Satyen Kale. $O(\sqrt{\log n})$ approximation to SPARSEST CUT in $\tilde{O}(n^2)$ time. In *FOCS '04: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, pages 238–247, Washington, DC, USA, 2004. IEEE Computer Society.

[5] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta algorithm and applications. Technical report, Princeton University, 2005.

[6] Sanjeev Arora and Satyen Kale. A combinatorial, primal-dual approach to semidefinite programs. In *STOC '07: Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 227–236, New York, NY, USA, 2007. ACM.

[7] Sanjeev Arora, Satish Rao, and Umesh Vazirani. Expander flows, geometric embeddings and graph partitioning. In *STOC '04: Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 222–231, New York, NY, USA, 2004. ACM.

[8] Josh Barnes and Piet Hut. A hierarchical o (n log n) force-calculation algorithm. *nature*, 324(6096):446–449, 1986.

[9] Amir Beck. On the convergence of alternating minimization for convex programming with applications to iteratively reweighted least squares and decomposition schemes. *SIAM Journal on Optimization*, 25(1):185–209, 2015.

[10] András A. Benczúr and David R. Karger. Approximating s-t minimum cuts in $O(n^2)$ time. In *STOC '96: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 47–55, New York, NY, USA, 1996. ACM.

[11] András A. Benczúr and David R. Karger. Randomized approximation schemes for cuts and flows in capacitated graphs. *CoRR*, cs.DS/0207078, 2002.

[12] Christer Borell. Positivity improving operators and hypercontractivity. *Mathematische Zeitschrift*, 180:225–234, 1982.

[13] Stéphane Boucheron, Gábor Lugosi, and Olivier Bousquet. Concentration inequalities. In *Advanced Lectures on Machine Learning*, pages 208–240. Springer, 2003.

[14] J. Bourgain. On lipschitz embedding of finite metric spaces in hilbert space. *Israel Journal of Mathematics*, 52(1):46–52, 1985.

[15] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[16] Hui Han Chin, Aleksander Madry, Gary L Miller, and Richard Peng. Runtime guarantees for regression problems. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 269–282. ACM, 2013.

[17] Paul Christiano, Jonathan A. Kelner, Aleksander Madry, Daniel A. Spielman, and Shang-Hua Teng. Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs. In Lance Fortnow and Salil P. Vadhan, editors, *STOC*, pages 273–282. ACM, 2011.

[18] Samuel I Daitch and Daniel A Spielman. Faster approximate lossy generalized flow via interior point algorithms. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 451–460. ACM, 2008.

[19] Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of johnson and lindenstrauss. *Random Structures & Algorithms*, 22(1):60–65, 2003.

[20] Stephen Demko. Condition numbers of rectangular systems and bounds for generalized inverses. *Linear Algebra and its Applications*, 78:199–206, 1986.

[21] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 448–455. ACM, 2003.

[22] Lisa K. Fleischer. Approximating fractional multicommodity flow independent of the number of commodities. *SIAM Journal on Discrete Mathematics*, 13:505–520, 2000.

[23] Yoav Freund and Robert E Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29(1):79–103, 1999.

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

[25] William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

[26] Jonathan A Kelner, Yin Tat Lee, Lorenzo Orecchia, and Aaron Sidford. An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 217–226. Society for Industrial and Applied Mathematics, 2014.

[27] Jonathan A Kelner, Gary L Miller, and Richard Peng. Faster approximate multicommodity flow using quadratically coupled flows. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 1–18. ACM, 2012.

[28] Rohit Khandekar, Satish Rao, and Umesh Vazirani. Graph partitioning using single commodity flows. In *STOC '06: Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 385–390, New York, NY, USA, 2006. ACM.

[29] GM Korpelevich. Extragradient method for finding saddle points and other problems. *Matekon*, 13(4):35–49, 1977.

[30] Ioannis Koutis, Gary L. Miller, and Richard Peng. A fast solver for a class of linear systems. *Commun. ACM*, 55(10):99–107, 2012.

[31] James R. Lee. On distance scales, embeddings, and efficient relaxations of the cut cone. In *SODA '05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 92–101, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics.

[32] Tom Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM*, 46(6):787–832, 1999.

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

[34] Jiri Matousek. *Lectures on Discrete Geometry*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[35] Elchanan Mossel, Oded Regev, Jeffrey E. Steif, and Benny Sudakov. Non-interactive correlation distillation, inhomogeneous markov chains, and the reverse bonami-beckner inequality. *Israel Journal of Mathematics*, 154, 2006.

[36] Arkadi Nemirovski. Prox-method with rate of convergence o (1/t) for variational inequalities with lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM Journal on Optimization*, 15(1):229–251, 2004.

[37] Yu Nesterov. Smooth minimization of non-smooth functions. *Mathematical programming*, 103(1):127–152, 2005.

[38] Yurii Nesterov. Dual extrapolation and its applications to solving variational inequalities and related problems. *Mathematical Programming*, 109(2-3):319–344, 2007.

[39] Lorenzo Orecchia. personal communication, 2009.

[40] Lorenzo Orecchia, Leonard J. Schulman, Umesh V. Vazirani, and Nisheeth K. Vishnoi. On partitioning graphs via single commodity flows. In *STOC '08: Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 461–470, New York, NY, USA, 2008. ACM.

[41] Richard Peng. Approximate undirected maximum flows in o (m polylog (n)) time. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1862–1867. SIAM, 2016.

[42] Serge A. Plotkin, David B. Shmoys, and Eva Tardos. Fast approximation algorithms for fractional packing and covering problems. *Mathematics of Operations Research*, 20:257–301, 1995.

[43] Ralph Tyrell Rockafellar. *Convex analysis*. Princeton university press, 2015.

[44] Jonah Sherman. Nearly maximum flows in nearly linear time. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 263–269. IEEE, 2013.

[45] Jonah Sherman. Generalized preconditioning and network flow. In *Proceedings of the Twenty-Eigth Annual ACM-SIAM Symposium on Discrete Algorithms*, page to appear. Society for Industrial and Applied Mathematics, 2017.

[46] Daniel D. Sleator and Robert Endre Tarjan. A data structure for dynamic trees. *J. Comput. Syst. Sci.*, 26(3):362–391, 1983.

[47] Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. *CoRR*, abs/cs/0607105, 2006.