Fast Algorithms for Directed Graph Partitioning Using Flows and Reweighted Eigenvalues

Lap Chi Lau[*] Kam Chuen Tung[†] Robert Wang[‡]

Abstract

We consider a new semidefinite programming relaxation for directed edge expansion, which is obtained by adding triangle inequalities to the reweighted eigenvalue formulation. Applying the matrix multiplicative weight update method on this relaxation, we derive almost linear-time algorithms to achieve $O(\sqrt{\log n})$-approximation and Cheeger-type guarantee for directed edge expansion, as well as an improved cut-matching game for directed graphs. This provides a primal-dual flow-based framework to obtain the best known algorithms for directed graph partitioning. The same approach also works for vertex expansion and for hypergraphs, providing a simple and unified approach to achieve the best known results for different expansion problems and different algorithmic techniques.

[*]Cheriton School of Computer Science, University of Waterloo. Supported by NSERC Discovery Grant.

[†]Cheriton School of Computer Science, University of Waterloo. Supported by NSERC Discovery Grant.

[‡]Cheriton School of Computer Science, University of Waterloo. Supported by NSERC Discovery Grant and Canada Graduate Scholarship.

1 Introduction

The main combinatorial quantity that we study in this work is the directed edge expansion with arbitrary vertex weights.

Definition 1.1 (π-Weighted Directed Edge Expansion)**.** *Let $G = (V, E, w)$ be a directed graph with edge weights $w : E \to \mathbb{R}^+$, equipped with vertex weights $\pi : V \to \mathbb{R}^+$. For $S \subseteq V$, let $\delta^+(S) := \{ij \in E : i \in S,\ j \notin S\}$ be the set of edges going out of S, and let $\delta^-(S) = \delta^+(\overline{S})$. Let $\pi(S) := \sum_{i \in S} \pi(i)$ be the π-weight of S. The π-weighted edge expansion of $S \subseteq V$ and of the graph G are defined as*

$$\vec{\phi}_\pi(S) := \frac{\min\{w(\delta^+(S)), w(\delta^-(S))\}}{\min\{\pi(S), \pi(\overline{S})\}} \quad \text{and} \quad \vec{\phi}_\pi(G) := \min_{\emptyset \neq S \subset V} \vec{\phi}_\pi(S).$$

This is a general problem that encompasses various expansion problems studied in the literature. The directed edge expansion problem is when $\pi(i) = 1$ for all $i \in V$, and this is equivalent (up to a factor of $\Theta(n)$ where n is the number of vertices) to the directed sparsest cut of G

$$\min_{\emptyset \neq S \subset V} \frac{\min\{w(\delta^+(S)), w(\delta^+(\overline{S}))\}}{|S| \cdot |V \backslash S|}$$

studied in [ACMM05, AK07, Kal07]. The directed edge conductance problem studied in [Yos19, LTW23] is when $\pi(i) = w(\delta^+(i)) + w(\delta^-(i))$, the weighted total degree of vertex i. Clearly, the corresponding problems in undirected graphs as studied in [ARV09, KRV06, AK07] can be reduced to Definition 1.1 by bidirecting the edges in the undirected graph. Also, the undirected vertex expansion problem studied in [FHL08, LRV13][1] and the directed vertex expansion problem studied in [LTW23] can be reduced to Definition 1.1 through a standard reduction of splitting each vertex into two. Furthermore, the corresponding problems in undirected and directed hypergraphs can be reduced to Definition 1.1 through a reduction of replacing each hyperedge by a vertex as shown in [CS18]. The main goal of this work is to design fast algorithms for approximating $\vec{\phi}_\pi$.

1.1 Previous Work

Before presenting our results, we first review previous work on approximating various graph expansion problem to provide the context of our work. We let $n := |V|$ and $m := |E|$ unless otherwise specified.

1.1.1 Undirected Graphs

The edge expansion, sparsest cut, and the edge conductance problems in undirected graphs are central problems in approximation algorithms. These problems have a rich literature with various techniques developed.

[1] To be precise, [FHL08] studies "minimum ratio vertex cuts", which is equivalent to undirected vertex expansion which we will define in Section 6. They use "vertex expansion" to refer to a different quantity incomparable to ours.

Spectral Method: Cheeger's inequality [AM85, Alo86] provides a near-linear time algorithm to return a set S with conductance $\lambda_2 \lesssim \phi(S) \lesssim \sqrt{\lambda_2}$ where λ_2 is the second smallest eigenvalue of the normalized Laplacian matrix of the graph.

Linear Programming: Leighton and Rao [LR99] gave an $O(\log n)$-approximation algorithm for sparsest cut based on linear programming. The dual problem of their linear program is to embed a complete graph into the original graph using flows.

Semidefinite Programming: Arora, Rao, and Vazirani [ARV09] gave a celebrated semidefinite programming $O(\sqrt{\log n})$-approximation algorithm for sparsest cut. They introduced novel geometric ideas in analyzing the triangle inequalities of the Goemans-Linial SDP relaxation. The dual problem of their semidefinite program (SDP) is to embed an expander graph into the original graph using flows.

Cut-Matching Game: Developing the idea of expander flows in [ARV09], Khandekar, Rao, and Vazirani [KRV06] introduced the cut-matching game as a combinatorial approach to obtain fast approximation algorithm for sparsest cut. Orecchia, Schulman, Vazirani, Vishnoi [OSVV08] improved the analysis of cut-matching game to give an $O(\log n)$-approximation algorithm for sparsest cut using $O(\log^3 n)$ undirected approximate max-flow computations. Since then, the cut-matching game has become a useful algorithmic tool on its own, with interesting applications in different problems [And10, Chu12, CE13, CL16, CGLNPS20, BGS20].

Primal-Dual Algorithms: Arora and Kale [AK07] developed a general primal-dual combinatorial approach to solve SDPs based on the matrix multiplicative weight update (MMWU) method. Using this, they gave an $\widetilde{O}(n^2)$-time $O(\sqrt{\log n})$-approximation algorithm for sparsest cut using $O(\log^2 n)$ multi-commodity flow computations. Notably, the cut-matching game in [OSVV08] can be interpreted as an instantiation of the matrix multiplicative weight update method.

Almost Linear-Time Algorithm: Sherman [She09] pushed the approach in [AK07] further to get the best of the semidefinite programming approach and the combinatorial approach. He gave an $O\big(\sqrt{\frac{1}{\epsilon}\log n}\big)$-approximation algorithm for sparsest cut using $n^{O(\epsilon)}$ approximate max-flow computations, which implies an almost linear-time $O(\sqrt{\log n})$-approximation algorithm for the problem.

1.1.2 Directed Graphs

The corresponding problems for directed graphs are not as well-understood, particularly in relation to fast algorithms.

Spectral Method: There was no known analog of Cheeger's inequality for directed graphs until recently. Lau, Tung, Wang [LTW23] defined a "spectral" quantity (using semidefinite programming) called the reweighted eigenvalue λ_2^*, and showed that there is a polynomial-time algorithm to return a set $S \subseteq V$ with $\lambda_2^* \lesssim \vec{\phi}(S) \lesssim \sqrt{\lambda_2^* \log(1/\lambda_2^*)}$. They left it as an open question to design a fast algorithm to return such a set.

Semidefinite Programming: Agarwal, Charikar, Makarychev and Makarychev [ACMM05] formulated an SDP using directed semi-metrics, and extended the analysis in [ARV09] to obtain $O(\sqrt{\log n})$-approximation algorithms for directed sparsest cut, directed balanced separator and other related

problems.

Cut-Matching Game: Louis [Lou10] defined an analog of the cut-matching game in [KRV06] for directed graphs, and used it to obtain an $O(\log^2 n)$-approximation algorithm for directed sparsest cut using $O(\log^3 n)$ max-flow computations.

Primal-Dual Algorithms: Using the matrix multiplicative weight update method on the SDP formulation in [ACMM05], Arora and Kale [AK07, Kal07] claimed an $O(\sqrt{\log n})$-approximation algorithm for directed sparsest cut with time complexity $O(n^{2+o(1)})$ plus $O(\log^3 n)$ maximum flow computations. Chan and Sun [CS18] pointed out an issue (which was acknowledged by Kale) in the trace bound in the analysis in [AK07], and consequently the number of iterations is only bounded by $\widetilde{O}(n^2)$ instead of $\widetilde{O}(1)$, and so the time complexity should be $O(n^{4+o(1)})$ plus $\widetilde{O}(n^2)$ maximum flow computations. Therefore, even with the recent breakthrough [CKLPPS22] in maximum flow computations in directed graphs, the time complexity of Arora-Kale's algorithm remains $\Omega(n^4)$ for directed sparsest cut. Moreover, unlike for undirected graphs, the connection between the cut-matching game in [Lou10] and the matrix multiplicative weight update method is not known.

1.2 Our Results

We consider a new semidefinite program for directed edge expansion based on the reweighted eigenvalue formulation. Using the MMWU method on this new SDP, we improve the algorithmic results for directed edge expansion, matching the corresponding results for undirected edge expansion.

1.2.1 Primal Formulation

We consider a new SDP relaxation for directed edge expansion in Definition 1.1. For undirected graphs, the SDP formulation in [ARV09] can be understood as the spectral formulation for second smallest Laplacian eigenvalue plus the ℓ_2^2 triangle inequalities [Tre16]. For directed graphs, our SDP formulation is to use the spectral formulation for reweighted eigenvalue in [LTW23] plus the ℓ_2^2 triangle inequalities.

Definition 1.2 (Reweighted Eigenvalue with Triangle Inequalities)**.** *Given an edge-capacitated directed graph $G = (V, E, w)$, we say that $F : E \to \mathbb{R}_{\geq 0}$ is a circulation*[2] *on G if $\sum_{j:ij\in E} F(i,j) = \sum_{j:ji\in E} F(j,i)$ for all $i \in V$. We let $\mathcal{F}(G)$ be the set of all circulations on G that also satisfy the capacity constraints $F(e) \leq w(e)$ for all $e \in E$. Given also vertex weights $\pi : V \to \mathbb{R}_{\geq 0}$, the $\lambda_\pi^{\triangle}(G)$*

[2] In [LTW23], F is called an Eulerian reweighting of G. In this paper, network flows is a unifying theme, and so we find it more suitable to call F a circulation.

program for directed edge expansion is

$$\begin{aligned}\lambda_\pi^{\triangle}(G) := \min_{v_1,\dots,v_n\in\mathbb{R}^n} \max_{F\in\mathcal{F}(G)} \quad & \sum_{i<j} \frac{1}{2}\big(F(i,j)+F(j,i)\big)\cdot\|v_i-v_j\|^2 \\ \text{subject to} \quad & \sum_{i=1}^n \pi(i)\cdot v_i = \vec{0} \\ & \sum_{i=1}^n \pi(i)\cdot\|v_i\|^2 = 1 \\ & \|v_i-v_k\|^2+\|v_k-v_j\|^2 \geq \|v_i-v_j\|^2 \qquad \forall i,j,k\in V.\end{aligned} \tag{1.1}$$

Here we use the convention that $F(i,j)=0$ if $ij\notin E$.

We note that the formulation in Definition 1.2 without the ℓ_2^2 triangle inequalities in the last line is exactly the formulation for reweighted eigenvalues in [LTW23, Proposition 3.4]. Just as the addition of triangle inequalities to the spectral formulation reduces the integrality gap of undirected edge expansion to $O(\sqrt{\log n})$ in [ARV09], we show the exact analog for directed edge expansion by using the spectral formulation for reweighted eigenvalues.

Theorem 1.3 ($O(\sqrt{\log n})$-Approximation for Directed Vertex Expansion)**.** *For any edge-capacitated directed graph $G=(V,E,w)$ with vertex weights $\pi: V\to\mathbb{R}_+$,*

$$\lambda_\pi^{\triangle}(G) \lesssim \vec{\phi}_\pi(G) \lesssim \sqrt{\log n}\cdot\lambda_\pi^{\triangle}(G).$$

The proof is a simple adaptation of that in [ARV09]. We will compare our formulation with that in [ACMM05] in Section 2.3.1, and we will compare the two dual formulations in Section 2.3.2. We note that the same approach of adding ℓ_2^2 triangle inequalities to reweighted eigenvalues provides considerably simpler formulations and proofs for undirected vertex expansion and hypergraph edge expansion than that in [FHL08] and in [LM14], while having the same integrality gap $O(\sqrt{\log n})$; see Section 6 for more details.

1.2.2 Dual Formulation

As in [ARV09], the dual program of $\lambda_\pi^{\triangle}$ in Definition 1.2 can be interpreted as embedding a directed expander flow into the original directed graph. Since our formulation in Definition 1.2 requires the flow to be a circulation, we obtain a new structural result about the existence of a circulation of high edge expansion as a dual certificate, which may be of independent interest.

Proposition 1.4 ($O(\sqrt{\log n})$ Dual Certificate)**.** *Given an edge-capacitated directed graph $G=(V,E,w)$ with vertex weights $\pi: V\to\mathbb{R}_+$, there exists a circulation $F\in\mathcal{F}$ on G satisfying edge capacity constraints with*

$$\phi_\pi(F) := \min_{\emptyset\neq S\subset V} \frac{\sum_{i\in S,j\notin S} F(i,j)}{\min\{\pi(S),\pi(\overline{S})\}} \gtrsim \frac{\vec{\phi}_\pi(G)}{\sqrt{\log n}}.$$

1.2.3 Primal-Dual Algorithms

Using the MMWU method in [AK07, Kal07] on $\lambda_\pi^\triangle$, combining with the chaining techniques in [She09], we extend Sherman's result to directed graphs.

Theorem 1.5 (Fast $O(\sqrt{\log n})$-Approximation to Directed Edge Expansion)**.** *For small enough $\epsilon > 0$, there is a randomized algorithm that, given any edge-capacitated directed graph $G = (V, E, w)$ with vertex weights $\pi : V \to \mathbb{R}_+$, uses $\widetilde{O}(n^{3\epsilon})$ directed max-flow computations to compute a cut $S \subseteq V$ with $\vec{\phi}_\pi(S) \lesssim \sqrt{\frac{\log n}{\epsilon}} \cdot \lambda_\pi^\triangle(G)$ with constant probability.*

Using the recent breakthrough [CKLPPS22] on directed maximum flow, Theorem 1.5 implies an $O(m^{1+O(\epsilon)})$-time $O(\sqrt{(\log n)/\epsilon})$-approximation algorithm for directed edge expansion. This is a significant improvement over the previous $O(n^{4+o(1)})$-time $O(\sqrt{\log n})$-approximation algorithm for directed sparsest cut by Arora and Kale [AK07, Kal07].

Since undirected vertex expansion can be reduced to directed edge expansion, this is also a significant improvement over the previous results [CK19, CS21] in fast approximation algorithms for undirected vertex expansion, where the best known result is a $O(\log^2 n)$-approximation using $O(\log^3 n)$ vertex-capacitated max-flow computations. We remark that our algorithm is simpler than Sherman's when restricted to undirected graphs, bypassing the use of multi-commodity flows[3]. See Section 2.2.3 and Section 2.3.2 for more discussions.

1.2.4 Cheeger-Type Guarantee

We show that the MMWU method can also be used to obtain a fast algorithm to output a set with the Cheeger-type guarantee in [LTW23].

Theorem 1.6 (Fast Cheeger-type Approximation)**.** *Given an edge-capacitated directed graph $G = (V, E, w)$, there is an almost linear time algorithm for approximating the directed edge conductance $\vec{\phi}(G)$ that returns a set S with $\vec{\phi}(S) \lesssim \sqrt{\vec{\phi}(G) \cdot \log \frac{1}{\vec{\phi}(G)}}$.*

This answers an open question in [LTW23] and provides a fast "spectral" algorithm for directed graph partitioning.

1.2.5 Cut-Matching Game

The cut-matching game is an interesting and useful way to construct an expander graph; see Section 2.2.4 for an introduction. Using the MMWU method, we also obtain a cut-player strategy that matches the cut-matching game result in [OSVV08] for undirected graphs.

[3] In a concurrent work, Kolmogorov [Kol23] also showed that Sherman's algorithm on undirected graphs can be simplified by bypassing the multi-commodity flow step. Their work does not generalize the algorithm to directed graphs and instead focuses on making the algorithm parallelizable. We leave as potential follow-up work whether our algorithms can also be made parallelizable.

Theorem 1.7 (Cut-Matching Game for Directed Edge Expansion)**.** *In the cut-matching game for directed graphs (see Section 2.3.3 for definition), there is a cut player strategy so that, in $O(\log^2 n)$ iterations, the union of the matchings played by the matching player is an Eulerian graph with edge expansion $\Omega(\log n)$.*

This is an improvement over the cut-matching game by Louis [Lou10], which only had an expansion lower bound of $\Omega(1)$. A corollary of Theorem 1.7 is a simple almost linear-time $O(\log n)$-approximation algorithm for directed edge expansion.

1.2.6 Unifying Framework

The reweighted eigenvalue formulations in [KLT22, LTW23] provide a unifying framework to obtain Cheeger-type inequalities for vertex expansion, directed graph expansions, and hypergraph expansions. In this study, we show that in all these cases, adding ℓ_2^2 triangle inequality constraints to the reweighted eigenvalue formulations gives $O(\sqrt{\log n})$-approximation algorithms for estimating these quantities, as well as fast algorithms for computing such approximations using expander flows and the chaining techniques [ARV09, AK07, Kal07, She09]. Our results bring the more general expansion problems closer to the basic undirected edge expansion problem, since both the formulations and the proofs are close analogs of the corresponding results for undirected edge expansion. Moreover, our proofs show that the MMWU method and the max-flow min-cut theorem can also be used to recover the Cheeger-type inequality and the cut-matching game, providing a common framework to analyze these different algorithmic techniques for graph expansion problems. Overall, we believe that our results simplify and unify the state-of-the-art of various problems and approaches studied in the literature.

2 Technical Review and Overview

Since our work revisits and extends several previous works [ARV09, KRV06, AK07, Kal07, She09, ACMM05, Lou10, LTW23], we review these previous techniques and mention some of our ideas for improvements along the way in the corresponding subsections, and we conclude with the common themes in Section 2.4.

2.1 Preliminaries

First, we introduce some notation that we will use throughout the paper. We use $\mathbb{R}_+$ to denote the set of positive real numbers and $\mathbb{R}_{\geq 0}$ to denote the set of non-negative real numbers. Given two functions $f, g : X \to \mathbb{R}_{\geq 0}$, we use $f \lesssim g$ to denote the existence of a positive constant $c > 0$, such that $f \leq c \cdot g$ always holds. We use $f \sim g$ to denote $f \lesssim g$ and $g \lesssim f$.

2.2 Previous Works on Undirected Sparsest Cut

2.2.1 Semidefinite Program with Triangle Inequalities

The seminal work of Arora, Rao and Vazirani [ARV09] proved that the following Goemans-Linial SDP relaxation for the undirected sparsest cut problem has an integrality gap of $O(\sqrt{\log n})$.

$$\begin{aligned} \min_{v_1,\ldots,v_n\in\mathbb{R}^n} \quad & \sum_{ij\in E} \|v_i - v_j\|^2 \\ \text{subject to} \quad & \sum_{i<j} \|v_i - v_j\|^2 = 1 \\ & \|v_i - v_k\|^2 + \|v_k - v_j\|^2 \geq \|v_i - v_j\|^2 \qquad \forall i,j,k \in V. \end{aligned} \tag{2.1}$$

Note that this formulation without the triangle inequalities in the last line is equivalent to the second smallest eigenvalue of the normalized Laplacian matrix when the graph is regular (see e.g. [Tre16]).

A major contribution in [ARV09] is a structure theorem on vectors satisfying the ℓ_2^2 triangle inequalities. It asserts that, given a "well-spread" set of vectors satisfying the ℓ_2^2 triangle inequalities, there are two large subsets L and R, such that all vectors in L are far away from all vectors in R.

Definition 2.1 (Well-Spread Vectors)**.** *Let $\{v_i\}_{i=1}^n$ be a set of vectors that satisfy $\sum_{i<j} \|v_i - v_j\|^2 = n^2$. Let $B(i,\delta) := \{j \in V : \|v_j - v_i\| \leq \delta\}$ denote the closed δ-ball centered at v_i. We say that $\{v_i\}_{i=1}^n$ is well-spread if $|B(i, \frac{1}{\sqrt{10}})| \leq \frac{n}{10}$ for all $i \in V$.*

Theorem 2.2 (ℓ_2^2 Structure Theorem [ARV09, Theorem 1])**.** *Let $\{v_i\}_{i=1}^n$ be a set of vectors[4] that satisfy the ℓ_2^2 triangle inequalities and $\sum_{i,j\in V} \|v_i - v_j\|^2 = n^2$. If $\{v_i\}_{i=1}^n$ is well-spread, then there exist two sets $L, R \subseteq V$ such that $|L|, |R| \geq \Omega(n)$ and*

$$d(L,R) := \min_{i\in L, j\in R} \|v_i - v_j\|^2 \gtrsim 1/\sqrt{\log n}.$$

Moreover, there is a randomized polynomial-time algorithm that finds such sets with high probability.

The proof consists of novel geometric arguments involving measure concentration and chaining. We will use Theorem 2.2 straightforwardly to prove that the new SDP formulation in Definition 1.2 has integrality gap $O(\sqrt{\log n})$. We will also use a refined version of the chaining result by Sherman [She09] for our fast algorithm in Theorem 1.5.

2.2.2 Expander Flows

Another important contribution of [ARV09] is the concept of expander flows. The idea of using multi-commodity flow to certify edge expansion was first introduced by Leighton and Rao [LR99].

[4] In [ARV09], the vectors v_i are assumed to be of unit length. We note that the structure theorem holds without this assumption as well; see for example [Rot16] for a writeup.

Definition 2.3 (Multi-Commodity Flow and Demand Graph)**.** *Let $G = (V, E, w)$ be an edge-capacitated undirected graph. Given demands d_{ij} for each $i, j \in V$, a multicommodity flow f assigns a value $f_p \geq 0$ to each path p in G such that (i) $\sum_{p \ni e} f_p \leq w_e$ for all $e \in E$ and (ii) $\sum_{p \in \mathcal{P}_{ij}} f_p = d_{ij}$ for all $i, j \in V$, where $\mathcal{P}_{ij}$ denotes the set of paths from i to j. The demand graph D is defined on the same vertex set V, with edge set $E' = V \times V$ and the weight of each edge ij being d_{ij}.*

For an edge-capacitated undirected graph $G = (V, E, w)$, let

$$\Phi(G) := \frac{\min_{S \subseteq V : |S| \leq |V|/2} w(\delta(S))}{|S||\overline{S}|}$$

be the value of the sparsest cut of G. If there is a multi-commodity flow in G with demand graph D, then it is not difficult to check that $\Phi(G) \geq \Phi(D)$. Leighton and Rao [LR99] used linear programming with the demand graph $D = K_n$, the complete graph on n vertices, to approximate the sparsest cut of G up to an approximation ratio $O(\log n)$.

The new idea in [ARV09] was to use semidefinite programming to search for a demand graph D with a feasible multi-commodity flow on G, and to lower bound the sparsest cut of G using the second eigenvalue of the Laplacian matrix of D through Cheeger's inequality. This approach can be summarized as

$$\begin{aligned} \max_{D, f} \quad & \lambda_2(L(D)) \\ \text{subject to} \quad & f \text{ is a multi-commodity flow on } G \text{ with demand graph } D \end{aligned} \tag{2.2}$$

2.2.3 Expander Flows vs Dual Program

In fact, the above approach of lower bounding $\Phi(G)$ can be interpreted as lower bounding the objective value of the dual of the Goemans-Linial SDP in (2.1). To see this, we first express the triangle inequalities as

$$\|v_{i_1} - v_{i_2}\|^2 + \|v_{i_2} - v_{i_3}\|^2 + \cdots + \left\|v_{i_{\ell-1}} - v_{i_\ell}\right\|^2 \geq \|v_{i_1} - v_{i_\ell}\|^2 \quad \forall p = (i_1, \ldots, i_\ell) \in \mathcal{P}(K_n),$$

where $\mathcal{P}(K_n)$ denotes the set of paths in the complete graph K_n on the same vertex set V. We write the primal program in matrix form. Let U be the matrix with the i-th column being v_i for $1 \leq i \leq n$ and let $X = U^T U$. Let $L_{i,j}$ be the Laplacian of the edge ij and

$$T_p := \sum_{k=1}^{\ell-1} L_{i_k, i_{k+1}} - L_{i_1, i_l}. \tag{2.3}$$

Then the Goemans-Linial SDP in (2.1) can be written as

$$\begin{aligned} \min_{X \succcurlyeq 0} \quad & \langle L(G), X \rangle \\ \text{subject to} \quad & \langle L(K_n), X \rangle = 1 \\ & \langle T_p, X \rangle \geq 0 \quad \forall p \in \mathcal{P}(K_n). \end{aligned} \tag{2.4}$$

One can check that strong duality holds, and the dual program can be written as

$$\begin{aligned} \max_{f_p \geq 0 : p \in \mathcal{P}(K_n)} \quad & \lambda \\ \text{subject to} \quad & \lambda \cdot L(K_n) \preccurlyeq L(G) - \sum_p f_p T_p. \end{aligned}$$

Therefore, the dual program of the Goemans-Linial SDP can be succinctly written as

$$\max_f \quad \lambda_2\Big(L(G) - \sum_p f_p T_p\Big). \tag{2.5}$$

The expander flow formulation in (2.2) is weaker than the dual program.

Claim 2.4. *The objective value of (2.2) is a lower bound on the objective value of (2.5).*

Proof. Let f be a multi-commodity flow on G with demand graph D, and F be the $n \times n$ matrix with $F(i,j) = \sum_{p \ni ij} f_p$. Then, check that $\sum_p f_p T_p = L(F) - L(D)$, and hence

$$L(D) = L(F) - \sum_p f_p T_p \preccurlyeq L(G) - \sum_p f_p T_p \quad \Longrightarrow \quad \lambda_2(L(D)) \leq \lambda_2\Big(L(G) - \sum_p f_p T_p\Big),$$

where the inequality $L(F) \preccurlyeq L(G)$ is because $F(i,j) \leq w_{ij}$ for all $(i,j) \in V \times V$. □

We remark that all previous works on undirected graphs [ARV09, KRV06, AK07, Kal07, She09] use the expander flow formulation in (2.2) to approximate sparsest cut. This can be understood as the dual program in (2.5) with the additional constraint that $\sum_{p \ni ij} f_p \leq w_{ij}$ for all $i, j \in V \times V$, which in particular implies that only the path variables f_p when p is a path in G are used. Since we will discuss several variations of the program (2.1) and take their duals, we will refer to dual programs with additional capacity constraints on the f_p variables such as (2.2) as the "constrained dual programs" and the original dual programs such as (2.5) as the "unmodified dual programs."

In proving Theorem 1.5, we will use the unmodified dual program of λ_π^Δ. As we will explain later, this will allow us to design a simpler primal-dual algorithm using the MMWU method, bypassing the use of multi-commodity flow as in [AK07, Kal07, She09].

2.2.4 Cut-Matching Game

The cut-matching game was first introduced by Khandekar, Rao and Vazirani [KRV06] as a fast combinatorial method for approximating sparsest cut in undirected graphs using flows. In this game, there is a cut player and a matching player who try to build an expander from the empty graph as follows. In each round, the cut player chooses a bisection $(S, \overline{S})$ of the vertices, and the matching player chooses a perfect matching between $(S, \overline{S})$. The goal of the cut player is to minimize the number of rounds so that the union of the matchings is guaranteed to be a good expander. Khandekar, Rao and Vazirani [KRV06] gave a cut player strategy that always builds a graph with $\Omega(1)$ edge expansion in $O(\log^2 n)$ rounds. Orecchia, Schulman, Vazirani, and Vishnoi [OSVV08]

gave an improved cut player strategy that always builds a graph with $\Omega(\log n)$ edge expansion in $O(\log^2 n)$ rounds. The proofs of these results are based on ad-hoc potential functions, although in hindsight the algorithm in [OSVV08] is very similar to the one using MMWU method in [AK07].

The original motivation of the cut-matching game is to build an expander flow to approximate sparsest cut. In each round, we aim to send a flow between the cut $(S, \overline{S})$ provided by the cut player. On the one hand, if such a flow cannot be sent, then we obtain a sparse cut by the max-flow min-cut theorem and the algorithm stops. On the other hand, if such a flow can be sent, then the demand pairs routed by this flow form a perfect matching between S and $\overline{S}$. Therefore, if we successfully send such a flow in each round, then the average of the flows is a multicommodity flow in the original graph, with the demand graph being the average of the perfect matchings, which is guaranteed to be an expander by the cut-matching game. In this case, we can prove a lower bound on the sparsest cut by the expander flow formulation in (2.2), with the approximation ratio depending on the parameters in the cut-matching game. The cut player strategy in [KRV06] gave an $O(\log^2 n)$-approximation for undirected sparsest cut using $O(\log^3 n)$ max-flow computations, while the one in [OSVV08] gave an $O(\log n)$-approximation using $O(\log^3 n)$ max-flow computations.

We remark that the cut-matching game has become a useful algorithmic tool on its own, with interesting applications in other important problems such as edge-disjoint paths [And10, Chu12, CL16] and dynamic graph problems [CGLNPS20, BGS20].

2.2.5 Matrix Multiplicative Weight Update Method

Arora and Kale [AK07, Kal07] developed a general primal-dual framework to solve SDPs using the matrix multiplicative weight update method. For our purpose, it would be better to understand this method from the viewpoint of regret minimization, which is the setting in online optimization. In each iteration t, the player chooses a density matrix X_t, which represents a probability distribution over the set of unit vectors. The player then observes a feedback matrix M_t with bounded spectral norm and incurs a loss of $\langle X_t, M_t \rangle$. The objective of the player is to minimize the total loss. In hindsight, if the player had knowledge of all the feedback matrices M_t from the start, then the best strategy would be to choose the density matrix vv^T where v is a unit-length minimum eigenvector of $\sum_t M_t$, with total loss $\lambda_{\min}(\sum_t M_t)$. The regret of the player is defined as $\sum_t \langle M_t, X_t \rangle - \lambda_{\min}(\sum_t M_t)$, the difference of the player's loss to this offline loss. Arora and Kale [AK07, Kal07] analyzed the following algorithm that sets X_t to be the matrix exponential of the feedback matrices.

Algorithm 1 Matrix Multiplicative Weight Update Algorithm

Initialization: $X_0 = \frac{1}{n} I_n$, $\eta \in (0, 1)$ as a step size
For $t = 0, \ldots, T-1$

1. Observe feedback matrix M_t such that $\|M_t\| \leq \rho$. Incur a loss of $\langle M_t, X_t \rangle$.
2. Compute $X'_{t+1} := \exp(-\eta \sum_{i=0}^{t} \frac{1}{\rho} M_i)$ and update $X_{t+1} := X'_{t+1} / \operatorname{tr}(X'_{t+1})$.

The requirement that M_t has bounded spectral norm, or $\|M_t\| \leq \rho$, is to control the regret bound.

The ρ parameter is called the "width" and is the key parameter in analyzing the matrix multiplicative weight update method in many applications.

Theorem 2.5 (Regret Bound [Kal07, Theorem 10])**.** *After T iterations of Algorithm 1, let $\overline{M} := \frac{1}{T}\sum_{t=0}^{T-1} M_t$, then*

$$\lambda_{\min}(\overline{M}) \gtrsim \frac{1}{T}\sum_{t=0}^{T-1}\langle M_t, X_t\rangle - \eta\rho - \frac{\rho \log n}{\eta T}. \tag{2.6}$$

If, in addition, each M_t satisfies $M_t \succcurlyeq 0$, then we have the stronger bound that

$$\lambda_{\min}(\overline{M}) \gtrsim \frac{1}{T}\sum_{t=0}^{T-1}\langle M_t, X_t\rangle(1-\eta) - \frac{\rho \log n}{\eta T}. \tag{2.7}$$

Theorem 2.5 is a key result that we will use to design fast algorithms.

2.2.6 Primal-Dual Algorithms for Sparsest Cut

Arora and Kale [AK07] uses the regret bound in Theorem 2.5 to design a primal-dual algorithm for approximating the sparsest cut problem. The setup is to either certify that the optimal value is at least $\Omega(\alpha)$ by building an expander flow solution to (2.2), or to find a cut of sparsity at most $\sqrt{\log n}\cdot\alpha$. In each iteration, the algorithm uses the density matrix X_t given by the matrix multiplicative weight update algorithm as a candidate primal solution to (2.4). To build a dual solution to (2.2), the idea is to use the regret minimization framework to reduce to the simpler task of finding a multi-commodity flow f_t whose demand graph D_t satisfies $\langle L(D_t), X_t\rangle \geq \alpha$. If such a multi-commodity flow with demand graph D_t can be found in each iteration t for $O(\log n)$ iterations, then the regret bound in Theorem 2.5 would imply that $\lambda_2(L(\frac{1}{T}\sum_t D_t)) \gtrsim \alpha$, and thus the average of the flows f_t is an expander flow solution to (2.2) with objective value at least $\Omega(\alpha)$.

The remaining task is that, given a density matrix X_t, either to find a multi-commodity flow f_t whose demand graph D_t satisfies $\langle L(D_t), X_t\rangle \geq \alpha$ and $\|L(D_t)\| \leq \rho$, or to find a cut with sparsity at most $O(\sqrt{\log n}\cdot\alpha)$. This task is usually called implementing the "oracle" for the MMWU method. To do so, consider the Gram decomposition $v_1, \ldots, v_n$ of X and note that $\langle L(D_t), X_t\rangle = \sum_{i,j} D_t(i,j)\,\|v_i - v_j\|^2$. To ensure that the width ρ is small, the algorithm only searches for demand graphs with bounded maximum degree. To ensure that the inner product $\langle L(D_t), X_t\rangle$ is large, the algorithm only routes flow between pairs of vertices (i,j) with $\|v_i - v_j\| = \Omega(1)$. If such a multi-commodity flow can be sent, then the oracle succeeds and the primal-dual algorithm proceeds to the next iteration. If not, using the dual solution to the multi-commodity flow problem, along with the geometric chaining arguments used in [ARV09], they showed how to find a cut with sparsity at most $O(\sqrt{\log n}\cdot\alpha)$ (see [AK07, Lemma 6.6 and Theorem 6.7]). The time complexity of their algorithm is $\widetilde{O}(n^2)$, where the bottleneck is in the multi-commodity flow computation in the implementation of the oracle.

To achieve $O(\log n)$-approximation, there is a much easier way to implement the oracle using only max-flow computations. The algorithm is to project the vectors $v_1, \ldots, v_n$ along a random direction,

and set up a single-commodity flow between the $\Omega(n)$ vertices with the lowest projection values and the $\Omega(n)$ vertices with the highest projection values. This algorithm is very similar to the cut-matching game in [OSVV08] that uses matrix exponentials to define a cut-player strategy.

2.2.7 Almost Linear-Time Primal-Dual Algorithm

Sherman [She09] pushed the approach in [AK07] further to almost get the best of the semidefinite programming approach ($O(\sqrt{\log n})$-approximation) and the combinatorial cut-matching game approach (near linear-time algorithms).

The approach in [She09] is to use an inner multiplicative weight update algorithm to compute the multicommodity flow in the oracle implementation, rather than doing it in a black-box manner as in [AK07]. Specifically, each iteration of this inner multiplicative weight update algorithm consists of chaining together matchings corresponding to flow paths of single-commodity flows. The single-commodity flows are set up using the random projection method as in the $O(\log n)$-approximation in [AK07], but the random directions for these flows are correlated and the distribution of the random directions is explicit and can be sampled efficiently. The main contribution of [She09] was to show that, after chaining together $\Theta(\sqrt{\log n})$ of these correlated random matchings, one can find not just one (as in [ARV09]), but many flow paths between pairs (i, j) such that $\|v_i - v_j\|$ is $\Omega(1)$. Using this chaining method as a subroutine, one can either find a good multicommodity flow whose demand graph satisfies $\sum_{i,j} D(i,j) \|v_i - v_j\|^2 \geq \alpha$ in $O(n^{1+\epsilon})$ time by running the inner multiplicative weight update algorithm, or find some direction along which the single commodity flow cannot be sent and an associated min-cut S with $\phi(S) \lesssim \alpha \cdot \sqrt{\frac{\log n}{\epsilon}}$.

Sherman's algorithm and its analysis are rather technical and we will provide more details in Section 4.3.2. We will use his main chaining result as a black-box in our algorithm for Theorem 1.5.

2.3 Previous Works on Directed Sparsest Cut

2.3.1 Directed Semi-Metric for Directed Sparsest Cut

Agarwal, Charikar, Macharychev and Macharychev [ACMM05] introduced an SDP for approximating directed sparsest cut using a directed semi-metric. The idea was to introduce an extra vector v_0 to the embedding, and to define the semi-metric as $d(i,j) := \|v_i - v_j\|^2 - \|v_i - v_0\|^2 + \|v_j - v_0\|^2 \geq 0$. The program is formulated as follows:

$$\begin{aligned} \min_{v: V \cup \{0\} \to \mathbb{R}^n} \quad & \sum_{ij \in E} w(i,j) \left(\|v_i - v_j\|^2 - \|v_i - v_0\|^2 + \|v_j - v_0\|^2 \right) \\ \text{subject to} \quad & \|v_i - v_j\|^2 + \|v_j - v_k\|^2 \geq \|v_i - v_k\|^2 \qquad \forall i, j, k \in V \cup \{0\} \\ & \sum_{i \in V} \pi(i) \cdot v_i = \vec{0} \\ & \sum_{i \in V} \pi(i) \cdot \|v_i\|^2 = 1. \end{aligned} \tag{2.8}$$

The $\lambda_\pi^\triangle$ program that we introduce in Definition 1.2 is less constrained than this program. We can see this by taking the linear programming dual of the inner maximization problem with respect to the $F(i,j)$ variables (see [LTW23, Lemma 3.21 and Lemma 3.22]):

$$\max_{F\in\mathcal{F}(G)} \sum_{i<j} \big(F(i,j)+F(j,i)\big)\cdot\|v_i-v_j\|^2 = \min_{r:V\to\mathbb{R}} \sum_{ij\in E} w(i,j)\cdot\max\Big\{0,\|v_i-v_j\|^2-r(i)+r(j)\Big\}.$$

Thus, we see that every feasible solution to (2.8) corresponds to a feasible solution to the $\lambda_\pi^\triangle(G)$ program with the same objective value by taking $r(i)=\|v_i-v_0\|^2$. The reason we present $\lambda_\pi^\triangle$ throughout the paper in the min-max form is that all our analyses make use of this min-max formulation of the problem, as it can be naturally captured by flows.

2.3.2 Primal-Dual Algorithm for Directed Sparsest Cut

Arora and Kale [AK07, Kal07] used the matrix multiplicative update method on the SDP (2.8) in [ACMM05] to obtain a primal-dual $O(\sqrt{\log n})$-approximation algorithm for directed sparsest cut.

One important difference with the algorithm for undirected sparsest cut is that they used the unmodified dual program of (2.8), which can be expressed as $\max_f \lambda_2\big(\vec{L}(D)-\sum_p f_p T_p\big)$, where D is the demand graph of a flow on G (see Section 2.2.3 for a discussion about these two dual programs). Recall from our previous discussion that simply using $\vec{L}(D)$ instead of $\vec{L}(D)-\sum_p f_p T_p$ as in the undirected case (i.e. using the constrained instead of unmodified dual program) would correspond to only enforcing ℓ_2^2 triangle inequalities along paths in the directed graph. Since paths in the directed graph are restricted by the orientation of the edges, it seems arbitrarily restrictive to only enforce triangle inequalities along directed paths.

Using the dual program $\max_f \lambda_2\big(\vec{L}(D)-\sum_p f_p T_p\big)$, there remains an important difference between the primal-dual algorithm here with that for the undirected sparsest cut. Unlike in the algorithm for undirected sparsest cut, the algorithm for directed sparsest cut does not involve the use of multicommodity flows. Instead, it tries to find a single-commodity flow f with demand graph D that pushes a lot of flow between pairs of vertices (i,j) such that $\|v_i-v_j\|$ is large, and to then use $\vec{L}(D)$ as the feedback matrix. If it fails to do so, then it finds many paths that violate the ℓ_2^2 triangle inequality, and it then uses $-y\sum_p T_p$ as the feedback matrix, where the sum is over the violating paths p and y is an appropriate scaling factor. The procedure for finding violating paths is implemented in time $O(n^{2+o(1)})$ using a special data structure about dynamic decremental spanners. This is the bottleneck and thus the runtime per matrix multiplicative weight update iteration is $O(n^{2+o(1)})$.

The original claim in [Kal07] was that $O(\log n)$ iterations suffice, but Chan and Sun [CS18] found that their analysis should only yield the weaker bound of $\widetilde{O}(n^2)$ iterations and thus a total runtime of $O(n^{4+o(1)})$. This is because of a technical issue in bounding the trace of feasible solutions in the primal program (see footnotes 1 and 2 in [CS18], with Kale's acknowledgement). Chan and Sun simplified their approach and obtained an $O(n^4)$-algorithm with the same approximation ratio $O(\sqrt{\log n})$, that also works for directed hypergraphs.

As mentioned in Section 2.2.2, we will use the unmodified dual program of $\lambda_\pi^\triangle$ similar to how Arora-Kale's uses the unmodified dual program of (2.8). We also use their "flows or violating paths" oracle

for this dual program, thus bypassing the multicommodity flow computation in [AK07, She09]. We observe that Sherman's chaining result can be used to find many violating paths efficiently, without using any special data structures. This gives us an almost linear-time $O(\sqrt{\log n})$-approximation algorithm for directed sparsest cut, which also simplifies the corresponding algorithm for undirected sparsest cut.

We end this subsection with the following technical remark about the primal-dual algorithm for directed sparsest cut using the SDP in [ACMM05].

Remark 2.6. *Because of the directed semi-metric with the special vector v_0, Arora and Kale needed to work with a non-PSD Laplacian $\vec{L}(G)$ with vertex set $V \cup \{0\}$ and with both positive and negative edge weights (specifically, edges (i,j) and $(j,0)$ have weight 1 while edge $(i,0)$ has weight -1). the Laplacian of the demand graph of a flow is used as a feedback matrix in each iteration. However, the newly introduced vertex 0 has large degree in any demand graph, thus making it difficult to bound the spectral norm of the feedback matrix, i.e. the width of the oracle. To address this, Arora and Kale duplicated the vertex 0 into n copies, and considered a graph on $2n$ vertices in order to have a better bound on the width.*

One advantage of our formulation $\lambda_\pi^\triangle$ in Definition 1.2 is that it is defined on the original graph, and this simplifies the primal-dual algorithm and the analysis for Theorem 1.5 considerably.

2.3.3 Cut-Matching Game for Directed Graphs

Louis [Lou10] developed a cut-matching game for directed graphs, where the matching player plays a directed matching, which is defined as an Eulerian graph where each vertex has indegree and outdegree exactly one. He analyzed a cut-player strategy that is similar to the one in [KRV06] and proved that, in $O(\log^2 n)$ iterations, the union of the directed matchings is an Eulerian graph with edge expansion $\Omega(1)$.

For undirected graphs, the matrix multiplicative update method can be used to give an improved cut-player strategy [AK07, OSVV08]. For directed graphs, however, the primal-dual algorithm is more complicated because of the directed semi-metric formulation as discussed in Remark 2.6, and it does not directly translate to a cut-matching game. Using the simpler $\lambda_\pi^\triangle$ formulation in Definition 1.2, which also has a natural correspondence with Eulerian subgraphs, we obtain an improved cut-player strategy as stated in Theorem 1.7 using the matrix multiplicative weight update method on $\lambda_\pi^\triangle$.

2.3.4 Reweighted Eigenvalues for Directed Graphs

Lau, Tung, and Wang [LTW23] defined the reweighted eigenvalue for directed edge expansion and use it to prove a Cheeger-type inequality for directed graphs. Given a directed graph $G = (V, E, w)$ with edge weights $w : E \to \mathbb{R}_+$, the maximum reweighted second eigenvalue problem seeks to find a circulation F satisfying edge capacity constraints (see Definition 1.2) such that the second smallest eigenvalue of the symmetric Laplacian of F is maximized.

Definition 2.7 (Maximum Reweighted Second Eigenvalue for π-Weighted Edge Expansion)**.** *Given an edge-capacitated directed graph $G = (V, E, w)$ and vertex weights $\pi : V \to R_+$, define the maximum reweighted second eigenvalue as*

$$\lambda_2^*(G) := \max_{F \in \mathcal{F}(G)} \lambda_2\left(\Pi^{-1/2}\left(D_F - \frac{F + F^\top}{2}\right)\Pi^{-1/2}\right)$$

where $\Pi = \mathrm{diag}(\pi)$, F is the $n \times n$ adjacency matrix of the circulation, and D_F is the diagonal degree matrix of $(F + F^T)/2$ with $D_F(i,i) = \sum_{j \in V} \frac{1}{2}(F(i,j) + F(j,i))$ for $1 \le i \le n$.

Using the semidefinite programming formulation for the second eigenvalue and von-Neumann min-max theorem, $\lambda_2^*(G)$ can be rewritten as the form in Definition 1.2 without the triangle inequalities.

The directed edge conductance $\vec{\phi}$ studied in [LTW23] is a special case of the directed edge expansion $\vec{\phi}_\pi$ in Definition 1.1 when $\pi(i) = w(\delta^+(i)) + w(\delta^-(i))$ for all $i \in V$. The directed Cheeger inequality in [LTW23] states that

$$\lambda_2^*(G) \lesssim \vec{\phi}(G) \lesssim \sqrt{\lambda_2^*(G) \cdot \log \frac{1}{\vec{\phi}(G)}}. \lesssim \sqrt{\lambda_2^*(G) \cdot \log \frac{1}{\lambda_2^*(G)}}. \tag{2.9}$$

In Theorem 1.6, we provide an almost linear-time algorithm to return a set S with $\vec{\phi}(S) \le \sqrt{\lambda_2^*(G) \cdot \log \frac{1}{\lambda_2^*(G)}}$. The idea is to use the regret minimization framework to construct an optimal circulation iteratively, and the observation is that this converges quickly when $\lambda_2^*(G)$ is large. This combines with our almost linear-time $O(\sqrt{\log n})$-approximation algorithm in Theorem 1.5 gives Theorem 1.6.

2.4 Our Techniques

We have already discussed the ideas of our main results in Section 1.2 in the corresponding subsections above when we reviewed the previous techniques. Here we highlight two common themes in our techniques.

One common theme is called the "metric rounding lemma" that we prove in Section 3.1, which is to use the max-flow min-cut theorem to find a sparse cut in a geometric embedding of the graph. All the algorithms in this paper use this lemma to find sparse cuts, including the almost linear-time $O(\sqrt{\log n})$-approximation in Theorem 1.5, the improved cut-matching game in Theorem 1.7, and interestingly even the Cheeger-type result whose original proof in [LTW23] is based on a threshold rounding algorithm.

Another common theme is the matrix multiplicative weight update method developed in [AK07]. All the algorithms in this paper use this method to construct the dual objects, including the expander flows in the $O(\sqrt{\log n})$-approximation in Theorem 1.5 and the cut-matching game in Theorem 1.7, as well as the circulation in reweighted eigenvalues in Theorem 1.6 and in the dual certificate in Proposition 1.4. The cut-matching game was considered original when it was introduced, but now we see that it can be derived systematically from the matrix multiplicative weight update method.

An important element in all our results is the reweighted eigenvalue formulation from [LTW23]. We believe that it is the right formulation, as it allows us to extend all known results for undirected graphs to directed graphs, in a way that is consistent with the formulations and the proofs for undirected graphs. As we discuss in Section 6, our technique of adding ℓ_2^2 triangle inequalities to reweighted eigenvalue formulations can be extended to directed vertex expansion and hypergraph edge expansion as well, providing a unifying method to extend the results for undirected graphs to more general settings.

2.5 Organization

In Section 3, we present the metric rounding lemma, and use it to prove Theorem 1.3 and to provide an alternative proof of the directed Cheeger inequality. In Section 4, we extend Sherman's result to directed graphs and prove Theorem 1.5. In Section 5, we also use the matrix multiplicative weight update method to compute reweighted eigenvalues, proving Theorem 1.6 and to design cut-matching game, proving Theorem 1.7. Finally, in Section 6, we outline how these results can be extended easily to vertex expansion and to hypergraphs.

3 Rounding Algorithms

In this section, we first present the metric rounding lemma in Section 3.1. Then, we will use it to prove that $\lambda_\pi^{\triangle}$ in Definition 1.2 has integrality gap $O(\sqrt{\log n})$ in Section 3.2, and also to provide an alternative proof of the Cheeger-type inequality in [LTW23] in Section 3.3.

3.1 Metric Rounding Lemma

The following metric rounding lemma will be used to find sparse cuts in all algorithms in this paper.

Lemma 3.1 (Metric Rounding Lemma)**.** *Let $G = (V, E, w)$ be an edge-capacitated directed graph. Let $d(\cdot,\cdot)$ be a metric on V, and let $\pi : V \to \mathbb{R}^+$ be an arbitrary weight function over V. Suppose we are given disjoint vertex subsets $L, R \subseteq V$ as input to the algorithm. Let $r := \pi(R)/\pi(L)$ and $r' := \max\{1, r\}$. Then there is an algorithm using $O(\log n)$ maximum flow computations to output a set S with*

$$\vec{\phi}_\pi(S) \lesssim \frac{r' \cdot \max_{F \in \mathcal{F}(G)} \sum_{i,j \in V} F(i,j) \cdot d(i,j)}{\sum_{i \in R} \pi(i) \cdot d(i, L)}.$$

Our proof of the lemma is constructive. Algorithm 2, Bidirectional Max-Flow, finds a maximum flow $\vec{f}$ from L to R and also a flow $\overleftarrow{f}$ from R to L with a prescribed target amount of flow. If either of the flow is not "saturating", then we find a sparse cut S using the max-flow min-cut theorem. Otherwise, we combine $\vec{f}$ and $\overleftarrow{f}$ to form a circulation F, which helps upper bound the expansion of the graph through the flow parameter β.

In the case where the flows $\vec{f}$ and $\overleftarrow{f}$ are saturated, we upper bound the flow value parameter β.

Algorithm 2 Bidirectional Max-Flow

Input: Graph G, semi-metric $d(\cdot,\cdot)$, vertex weights $\pi : V \to \mathbb{R}^+$ as given in Lemma 3.1; $L, R \subseteq V$ such that $L \cap R = \emptyset$, flow value parameter $\beta \in \mathbb{R}^+$, and congestion parameter $\kappa \in \mathbb{R}^+$

1. Let $r := \pi(R)/\pi(L)$. Construct flow network $\vec{G}$ from G as follows: add vertices s and t to G. Connect s to each vertex $i \in L$ with an arc (s, i) of capacity $r \cdot \beta \cdot \pi(i)$. Connect each vertex $j \in R$ to t with an arc (j, t) of capacity $\beta \cdot \pi(j)$. Multiply the capacities of the edges in G by κ.

2. Construct $\overleftarrow{G}$ in the same way as $\vec{G}$, but with arcs directed from L to s and from t to R instead.

3. Compute s-t maximum flow $\vec{f}$ on $\vec{G}$ and t-s maximum flow $\overleftarrow{f}$ on $\overleftarrow{G}$. If one of $\vec{f}$ or $\overleftarrow{f}$ does not saturate all source and sink edges (i.e. if maximum flow value is less than $\beta \cdot \pi(R)$), output the minimum cut S associated with the non-saturating flow. Otherwise, output the circulation $F = \frac{1}{2}(\vec{f} + \overleftarrow{f})$.

Lemma 3.2 (Saturated Case)**.** *Suppose $d(\cdot,\cdot)$ is a metric and Algorithm 2 outputs a circulation F. Then,*

$$\beta \leq \frac{\sum_{ij \in E} F(i,j) \cdot d(i,j)}{\sum_{i \in R} \pi(i) \cdot d(i, L)},$$

where $F(i,j) = \frac{1}{2} \sum_{p \ni (i,j)} \left(\vec{f}(p) + \overleftarrow{f}(p)\right)$ defines the flow graph of the circulation returned in step 3.

Proof. Each flow can be decomposed into a set of (weighted) flow paths from source to sink. For each $j \in R$, let $\vec{\mathcal{P}}(j)$ be the set of s-t flow paths in $\vec{f}$ entering t through j, and let $\overleftarrow{\mathcal{P}}(j)$ be the set of t-s flow paths in $\overleftarrow{f}$ leaving t through j. For a particular flow path $p = (s, i_1, i_2, \ldots, i_k, t)$ or $p = (t, i_k, i_{k-1}, \ldots, i_1, s)$, let $d(p) = \sum_{\ell=1}^{k-1} d(i_\ell, i_{\ell+1})$ be its length. Note that for any path $p \in \vec{\mathcal{P}}(j)$ or $p \in \overleftarrow{\mathcal{P}}(j)$, by triangle inequality,

$$d(p) = \sum_{\ell=1}^{k-1} d(i_\ell, i_{\ell+1}) \geq d(i_1, i_k) \geq d(j, L),$$

where the last inequality is because $i_k = j$ and $i_1 \in L$. Then,

$$\begin{aligned}
\sum_{p \in \vec{f}} \vec{f}(p) \cdot d(p) + \sum_{p \in \overleftarrow{f}} \overleftarrow{f}(p) \cdot d(p) &= \sum_{j \in R} \left[\sum_{p \in \vec{\mathcal{P}}(j)} \vec{f}(p) \cdot d(p) + \sum_{p' \in \overleftarrow{\mathcal{P}}(j)} \overleftarrow{f}(p') \cdot d(p') \right] \\
&\geq \sum_{j \in R} d(j, L) \left[\sum_{p \in \vec{\mathcal{P}}(j)} \vec{f}(p) + \sum_{p' \in \overleftarrow{\mathcal{P}}(j)} \overleftarrow{f}(p') \right] \\
&= 2\beta \sum_{j \in R} \pi(j) \cdot d(j, L),
\end{aligned}$$

where the last equality is due to both $\vec{f}$ and $\overleftarrow{f}$ being saturating. Thus, we have

$$2\sum_{ij\in E} F(i,j)\cdot d(i,j) = \sum_{p\in\vec{f}} \vec{f}(p)\cdot d(p) + \sum_{p\in\overleftarrow{f}} \overleftarrow{f}(p)\cdot d(p) \geq 2\beta \sum_{i\in R} \pi(i)\cdot d(i,L).$$

Rearranging gives the desired result. □

On the other hand, if either of the flows $\vec{f}$ or $\overleftarrow{f}$ is unsaturated, we extract from it a cut with bounded expansion. This is a slight extension of [KRV06, Lemma 3.7] to the π-weighted and vertex-capacitated settings, and so we include a proof here.

Lemma 3.3 (Unsaturated Case)**.** *Suppose Algorithm 2 outputs a cut S. Then $\vec{\phi}_\pi(S) \leq \beta r'/\kappa$, where $r' := \max\{1, r\}$.*

Proof. Suppose $\vec{f}$ is the non-saturating flow; the other case is similar (we would look at $\delta^-(S)$ for S defined below). We obtain from it a cut, which is a set of edges whose removal would make it impossible to go from s to t. Let $S \subseteq V$ be the set of vertices reachable from s after removing the cut edges. Let $V_s \subseteq L$ be the set of vertices connected by a cut edge from s, and let $V_t \subseteq R$ be the set of vertices connected by a cut edge to t. We claim that

$$w(\delta_G^+(S)) \leq \frac{\beta}{\kappa}\big(\pi(R) - r\cdot\pi(V_s) - \pi(V_t)\big), \quad \pi(S) \geq \pi(L) - \pi(V_s), \quad \text{and } \pi(V-S) \geq \pi(R) - \pi(V_t).$$

The first inequality comes from the fact that $\{si \mid i \in V_s\} \cup \delta_G^+(S) \cup \{jt \mid j \in V_t\}$ is the minimum cut obtained, with total weight equal to $r\cdot\beta\cdot\pi(V_s) + \beta\cdot\pi(V_t) + \kappa\cdot w(\delta_G^+(S))$ by our construction of $\vec{G}$, which is at most $\beta\cdot\pi(R)$. The second and third inequalities follow from the facts that $L\setminus V_s \subseteq S$ and $R\setminus V_t \subseteq V - S$. Since $\pi(R) = r\cdot\pi(L)$, it follows that

$$\vec{\phi}_\pi(S) = \frac{w(\delta^+(S))}{\min\{\pi(S), \pi(V-S)\}} \leq \frac{\beta}{\kappa}\max\left\{\frac{\pi(R)-\pi(V_t)}{\pi(R)-\pi(V_t)}, \frac{r(\pi(L)-\pi(V_s))}{\pi(L)-\pi(V_s)}\right\} = \frac{\beta\cdot r'}{\kappa}.$$

□

Now we are ready to prove the metric rounding lemma.

Proof of Lemma 3.1. In Algorithm 2, choose $\kappa = 2r'$. Let α be such that the algorithm outputs a circular flow f when $\beta = \alpha$ and outputs a cut S when $\beta = 2\alpha$. When a cut S is output at $\beta = 2\alpha$, by Lemma 3.3 (unsaturated case), the vertex or edge expansion of S is at most $\beta\cdot r'/\kappa = \alpha$. When a circulation F is output at $\beta = \alpha$, then by construction $F' = F/\kappa$ is a circulation satisfying the edge or vertex capacity constraints of G, i.e. $F' \in \mathcal{F}(G)$. Therefore, by Lemma 3.2 (saturated case),

$$\vec{\phi}_\pi(S) \leq \alpha \leq \kappa\cdot\frac{\sum_{ij\in E} F'(i,j)\cdot d(i,j)}{\sum_{i\in R}\pi(i)\cdot d(i,L)} \leq 2r'\cdot\max_{F\in\mathcal{F}(G)}\frac{\sum_{ij\in E} F(i,j)\cdot d(i,j)}{\sum_{i\in R}\pi(i)\cdot d(i,L)}$$

Finally, note that we can find α using binary search on the range $[\Omega(1/\mathrm{poly}(n)), O(\mathrm{poly}(n))]$. Therefore, we only need to invoke Algorithm 2 $O(\log n)$ times, leading to a total of $O(\log n)$ maximum flow computations. □

3.2 Rounding Algorithm for Semidefinite Programming Solution

In this subsection, we prove Theorem 1.3 that the integrality gap of $\lambda_\pi^\triangle(G)$ is $O(\sqrt{\log n})$. The proof is by applying the metric rounding lemma on the two sets provided by the structure theorem of Arora, Rao, and Vazirani (see Theorem 2.2).

We note that by adding triangle inequalities in the reweighted eigenvalues in [KLT22, LTW23], essentially the same proof implies $O(\sqrt{\log n})$-approximation algorithms for undirected and directed vertex expansions, and undirected and directed hypergraph expansions (See Section 6 for more details). These approximation guarantees are all known previously, but with different formulations and proof techniques. In particular, the SDP relaxation for vertex expansion obtained through our approach is considerably simpler than that obtained by Feige, Hajiaghayi, and Lee [FHL08]. This demonstrates that our approach of using reweighted eigenvalues and triangle inequalities provides a simple and unifying way to recover all these results.

The proof that $\lambda_\pi^\triangle(G)$ is indeed an SDP relaxation of directed edge expansion can be found in Appendix A.

Proposition 3.4 (Easy Direction)**.** *For any edge-capacitated directed graph $G = (V, E, w)$ with vertex weights $\pi : V \to \mathbb{R}_+$, it holds that $\lambda_\pi^\triangle(G) \le 2\vec{\phi}_\pi(G)$.*

We will use the structure theorem in [ARV09] for the proof of $\vec{\phi}_\pi(G) \lesssim \sqrt{\log n} \cdot \lambda_\pi^\triangle(G)$. Since we consider π-weighted directed edge expansion, we need the following weighted version of the structure theorem. The proof of the weighted version is a straightforward reduction to the unweighted version in Theorem 2.2 and is deferred to Appendix A (see [ACMM05, Algorithm 1] for a similar weighted structure theorem and reduction).

Lemma 3.5 (π-Weighted Structure Theorem)**.** *Let $G = (V, E, w)$ be an edge-capacitated directed graph with vertex weights $\pi : V \to \mathbb{R}_+$ and $\pi(V) = 1$. Let $\{v_i\}_{i=1}^n$ be a set of embedding vectors satisfying ℓ_2^2 triangle inequalities and $\sum_{i,j\in V} \pi(i)\cdot\pi(j)\cdot\|v_i - v_j\|^2 = 1$. The embedding $\{v_i\}_{i=1}^n$ is said to be well-spread if $\pi(B(i, 1/\sqrt{10})) \le 1/10$ for all $i \in V$. If $\{v_i\}_{i=1}^n$ is well-spread, then there exist two subsets $L, R \subseteq V$ with $\pi(L), \pi(R) \ge \Omega(1)$ and*

$$d(L, R) := \min_{i\in L, j\in R} \|v_i - v_j\|^2 \gtrsim 1/\sqrt{\log n}.$$

Moreover, there is a randomized polynomial-time algorithm that finds such sets with high probability.

With the ℓ_2^2 triangle inequalities, the function $d(i, j) := \|v_i - v_j\|^2$ is a metric. We will apply the metric rounding lemma to find a sparse cut, with the observations that the numerator term $\max_{F\in\mathcal{F}(G)} \sum_{(i,j)\in E} F(i, j) \cdot d(i, j)$ in Lemma 3.1 is exactly the inner maximization problem of $\lambda_\pi^\triangle(G)$, and the denominator term in Lemma 3.1 is large using the two subsets L, R provided by the structure theorem.

Theorem 3.6 (Hard Direction)**.** *Let $G = (V, E, w)$ be an edge-capacitated directed graph with vertex weights $\pi : V \to \mathbb{R}_+$. There is a polynomial-time algorithm which, with high probability, finds a set $S \subseteq V$ with $\vec{\phi}_\pi(S) \lesssim \lambda_\pi^\triangle(G) \cdot \sqrt{\log n}$.*

Proof. Let $\{v_i\}_{i=1}^n$ be an optimal solution to the $\lambda_\pi^\triangle(G)$ program. Let $d(i,j) = \|v_i - v_j\|^2$, which is a metric by the ℓ_2^2 triangle inequalities in $\lambda_\pi^\triangle(G)$. By Lemma 3.1, given two subsets L and R, there is a subset $S \subseteq V$ with

$$\vec{\phi}_\pi(S) \lesssim \frac{r' \cdot \max_{F \in \mathcal{F}(G)} \sum_{(i,j)\in E} F(i,j) \cdot \|v_i - v_j\|^2}{\sum_{i\in R} \pi(i) \cdot d(i,L)} = \frac{2r' \cdot \lambda_\pi^\triangle(G)}{\sum_{i\in R} \pi(i) \cdot d(i,L)}. \tag{3.1}$$

There are two cases to consider: the "well-spread" case and the "large core" case. The difference in these two cases lies in the different choices of L and R to apply the metric rounding bound in (3.1). In either case, we assume without loss of generality that $\pi(V) = 1$. Also, by a straightforward calculation that we will show in Appendix A, the two normalization constraints in $\lambda_\pi^\triangle(G)$ in Definition 1.2 imply the following condition.

Fact 3.7. *If* $\sum_{i\in V} \pi(i) \cdot v_i = \vec{0}$ *and* $\sum_{i\in V} \pi(i) \cdot \|v_i\|^2 = 1$, *then* $\sum_{i,j\in V} \pi(i) \cdot \pi(j) \cdot \|v_i - v_j\|^2 = 2$.

Suppose the vectors $\{v_i\}_{i=1}^n$ are well-spread. Since $\sum_{i,j\in V} \pi(i) \cdot \pi(j) \cdot \left\| \frac{1}{\sqrt{2}} v_i - \frac{1}{\sqrt{2}} v_j \right\|^2 = 1$, we can apply Lemma 3.5 to obtain two subsets $L, R \subseteq V$ with $\pi(L), \pi(R) \geq \Omega(1)$ and $d(L,R) \gtrsim 1/\sqrt{\log n}$ in randomized polynomial time. This implies that the denominator in (3.1) is

$$\sum_{i\in R} \pi(i) \cdot d(i,L) \gtrsim \pi(R) \cdot \frac{1}{\sqrt{\log n}} \gtrsim \frac{1}{\sqrt{\log n}},$$

and thus we get from the metric rounding bound a set S with $\vec{\phi}_\pi(S) \lesssim \sqrt{\log n} \cdot \lambda_\pi^\triangle(G)$ as $r' = \max\{1, \pi(R)/\pi(L)\} = O(1)$.

Otherwise, we are in the large core case, where there is a vertex $i^* \in V$ with $\pi(B(i^*, 1/\sqrt{10})) > 1/10$. In this case, we set $L := B(i^*, 1/\sqrt{10})$ and $R := V \setminus L$ with $r' = \max\{1, \pi(R)/\pi(L)\} = O(1)$, and use the following lemma to lower bound the denominator in (3.1).

Lemma 3.8 (Total Distance to Core)**.** *Let $d : V \times V \to \mathbb{R}_{\geq 0}$ be a semi-metric (i.e. satisfying all axioms of metric except possibly the triangle inequality). Let $s \geq 1$ so that $d(\cdot,\cdot)$ satisfies an s-relaxed triangle inequality: $d(i,j) \leq s \cdot (d(i,k) + d(k,j))$ for all $i,j,k \in V$. Let $\pi : V \to \mathbb{R}_+$ be a weight function with $\pi(V) = 1$ and suppose $d(\cdot,\cdot)$ satisfies $\sum_{i,j\in V} \pi(i) \cdot \pi(j) \cdot d(i,j) = 2$. Let $L \subseteq V$ be a subset with diameter* $\operatorname{diam}(L) := \max_{i,j\in L} d(i,j)$. *Then*

$$\sum_{i\notin L} \pi(i) \cdot d(i,L) \geq \frac{1}{s^2} - \frac{1}{2} \operatorname{diam}(L)$$

Applying Lemma 3.8 with $s = 1$ and $\operatorname{diam}(L) \leq 2 \cdot 1/10 = 1/5$, it follows that

$$\sum_{i\in R} \pi(i) \cdot d(i,L) = \sum_{i\notin L} \pi(i) \cdot d(i,L) \geq 1 - \frac{1}{10} = \frac{9}{10},$$

and thus we get from the metric rounding bound in (3.1) a set S with $\vec{\phi}_\pi(S) \lesssim \lambda_\pi^\triangle(G)$.

The proof of Lemma 3.8 is in Appendix A, which was already done in previous works ([ARV09], [AK07]) for the uniform case. □

Theorem 1.3 follows immediately from Theorem 3.6 and Proposition 3.4.

3.3 Rounding Algorithm for Spectral Solution

In this subsection, we provide an alternative proof of the Cheeger-type inequality for directed graphs in (2.9) using the metric rounding lemma, where the original proof in [LTW23] is by a refined "threshold rounding" algorithm. This proof will be used in the proof of Theorem 1.6 in Section 5.1.1, as the threshold rounding algorithm in [LTW23] requires a linear programming duality step which is not clear how to be implemented in almost linear time.

We note that essentially the same proof works for the ordinary Cheeger's inequality [AM85, Alo86], as well as the Cheeger-type inequalities for directed vertex expansion and hypergraph edge conductance in [LTW23] (see Section 6). This illustrates the max-flow min-cut theorem in the proof of the metric rounding lemma as a unifying method to find sparse cuts in different settings.

Recall from Section 2.3.4 that $\vec{\phi}(G)$ denotes the directed edge conductance, which is the special case of directed edge expansion in Definition 1.1 when $\pi(i) = d_w(i) := w(\delta^+(i)) + w(\delta^-(i))$ is the total degree of i. We will focus on the proof of the "hard direction" of (2.9) that

$$\vec{\phi}(G) \lesssim \sqrt{\lambda_2^*(G) \cdot \log\left(1/\vec{\phi}(G)\right)}.$$

Also recall from Section 2.3.4 that $\lambda_2^*(G)$ can be written as the SDP in Definition 1.2 without the triangle inequalities. In [LTW23], the first step of the proof of the hard direction is to relate the $\lambda_2^*(G)$ program to the following "one-dimensional ℓ_1 program", which was done by using Gaussian projection and applying Cauchy-Schwarz inequality.

Lemma 3.9 (One-Dimensional ℓ_1 Program [LTW23, Definition 3.19])**.** *Given an edge-capacitated directed graph $G = (V, E, w)$ with vertex weights $d_w : i \mapsto \sum_{e:e\ni i} w(e)$, let*

$$\begin{aligned} \eta_e(G) := \min_{v:V\to\mathbb{R}} \max_{F\in\mathcal{F}(G)} \quad & \frac{1}{2}\sum_{ij\in E} F(i,j)\cdot|v(i)-v(j)| \\ \text{subject to} \quad & \sum_{i\in V} d_w(i)\cdot v(i) = 0 \\ & \sum_{i\in V} d_w(i)\cdot|v(i)| = 1. \end{aligned}$$

Then, it holds that

$$\eta_e(G) \lesssim \sqrt{\lambda_2^*(G) \cdot \log\left(1/\vec{\phi}(G)\right)}.$$

The second step in [LTW23] is to use a refined threshold rounding algorithm to prove that $\vec{\phi}(G) \lesssim \eta_e(G)$, thus proving the hard direction. Here we will use the metric rounding lemma to prove that $\vec{\phi}(G) \lesssim \eta_e(G)$. The reasons that we can apply the metric rounding lemma to the one-dimensional ℓ_1 program, but not to the n-dimensional ℓ_2^2 program, are as follows: (i) $d_1(i,j) := |v(i) - v(j)|$ is a metric while $d_2(i,j) := \|v(i) - v(j)\|^2$ needs not be; (ii) It is natural and straightforward to define sets L and R for a solution to the one-dimensional ℓ_1 program, but not for the n-dimensional ℓ_2^2 program. Therefore, we may view the Gaussian projection and Cauchy-Schwarz steps as reducing to 1-dimension and "metrifying" the objective, so that we can apply metric rounding.

3.3.1 Proof of the Second Step

We aim to prove that $\vec{\phi}(G) \lesssim \eta_e(G)$ using the metric rounding lemma. Let $v(1), v(2), \ldots, v(n) \in \mathbb{R}$ be an optimal solution to the $\eta_e(G)$ program. Set $d(i,j) := |v(i) - v(j)|$ which is a metric. Let $L := \{i \in V : v(i) \le 0\}$ and $R := \{j \in V : v(j) > 0\}$. We assume without loss of generality that $r := \mu(R)/\mu(L) \le 1$ so that $r' := \max\{1, r\} = 1$. From the definitions of L and R and the constraints on $v(i)$, one can verify that

$$d(i, L) \ge |v(i)| \quad \forall i \in R \quad \text{and} \quad \sum_{i \in R} d_w(i) \cdot |v(i)| = \frac{1}{2} \sum_{i \in V} d_w(i) \cdot |v(i)| = \frac{1}{2}.$$

Therefore, applying Lemma 3.1, it follows that

$$\begin{aligned} \vec{\phi}(G) = \vec{\phi}_{d_w}(G) &\lesssim \frac{r' \cdot \max_{F \in \mathcal{F}(G)} \sum_{ij \in E} F(i,j) \cdot d(i,j)}{\sum_{i \in R} d_w(i) \cdot d(i, L)} \\ &\le \frac{\max_{F \in \mathcal{F}(G)} \sum_{ij \in E} F(i,j) \cdot |v(i) - v(j)|}{\sum_{i \in R} d_w(i) \cdot |v(i)|} \\ &= 2 \max_{F \in \mathcal{F}(G)} \sum_{ij \in E} F(i,j) \cdot |v(i) - v(j)| \\ &= 4 \cdot \eta_e(G). \end{aligned}$$

This completes the proof of the hard direction of (2.9).

4 Almost Linear-Time Primal-Dual $O(\sqrt{\log n})$-Approximation

The main goal of this section is to prove Theorem 1.5. First, we will derive the dual program of $\lambda_\pi^\triangle(G)$ in Section 4.1. Then, in Section 4.2, we describe the primal-dual algorithm using the matrix multiplicative weight update method assuming a black-box algorithm for the oracle exists. In Section 4.3, we present the geometric results in [ARV09, AK07, She09] for the design of the oracle, and implement the oracle in the easy "large core" case. Then, in Section 4.3.2, we implement the oracle in the more difficult "well spread" case, in which we use Sherman's chaining theorem to find many paths that violate the triangle inequality. We conclude with the proofs of Theorem 1.5 and Proposition 1.4 in Section 4.5.

4.1 Dual Program of $\lambda_\pi^\triangle$

We construct the dual program of $\lambda_\pi^\triangle(G)$ in a similar way as in Section 2.2.3 for the dual program of the Goemans-Linial relaxation in (2.1).

We first write the primal program $\lambda_\pi^\triangle(G)$ in Definition 1.2 in matrix form. Let V be the matrix with the i-th column being v_i for $1 \le i \le n$ and let $X = V^T V$. Let $L_{i,j}$ be the Laplacian of an undirected edge ij. For a matrix A that is not necessarily symmetric, define the symmetric Laplacian of A as $L_{\text{sym}}(A) = \frac{1}{2} \sum_{i,j} (A(i,j) + A(j,i)) \cdot L_{i,j}$. As in Section 2.2.3, we express the triangle inequalities

redundantly as inequalities along paths in K_n, and let $T_p := \sum_{k=1}^{\ell-1} L_{i_k,i_{k+1}} - L_{i_1,i_l}$ for a path $p = (i_1, \ldots, i_\ell)$. Let Π be the diagonal matrix with $\Pi(i,i) = \pi(i)$ for $1 \le i \le n$. Then, check that $\lambda_\pi^{\triangle}(G)$ in Definition 1.2 can be written as

$$\begin{aligned} \lambda_\pi^{\triangle}(G) \quad = \quad & \min_{X \succcurlyeq 0} \max_{F \in \mathcal{F}(G)} \quad \langle L_{\mathrm{sym}}(F), X \rangle \\ & \text{subject to} \quad \langle \Pi \mathbb{1}\mathbb{1}^\top \Pi, X \rangle = 0 \\ & \qquad\qquad\quad \langle \Pi, X \rangle = 1 \\ & \qquad\qquad\quad \langle T_p, X \rangle \ge 0 \quad \forall p \in \mathcal{P}(K_n). \end{aligned}$$

To derive the dual of $\lambda_\pi^{\triangle}(G)$, we apply von Neumann's minimax theorem to switch the order of the min and the max, and then take the SDP dual of the inner minimization program to obtain

$$\begin{aligned} & \max_{F \in \mathcal{F}(G)} \quad \max_{\substack{\lambda, x \in \mathbb{R} \\ y_p \ge 0 : p \in \mathcal{P}(K_n)}} \quad \lambda \\ & \text{subject to} \quad \sum_p y_p T_p + \lambda \Pi + x \Pi \mathbb{1}\mathbb{1}^\top \Pi \preccurlyeq L_{\mathrm{sym}}(F). \end{aligned}$$

The dual constraint can be rewritten as

$$\lambda I + x \Pi^{\frac{1}{2}} \mathbb{1}\mathbb{1}^\top \Pi^{\frac{1}{2}} \preccurlyeq \Pi^{-\frac{1}{2}} \left(L_{\mathrm{sym}}(F) - \sum_p y_p T_p \right) \Pi^{-\frac{1}{2}}.$$

Note that the vector $\Pi^{\frac{1}{2}} \mathbb{1}$ is in the null space of the right hand side, as $\mathbb{1}$ is in the nullspace of any Laplacian matrix. Therefore, for the dual constraint to hold, an optimal dual solution must set $x = -\lambda/\pi(V)$, so as to make the component $\Pi^{\frac{1}{2}} \mathbb{1}$ to be zero on the left hand side. Therefore, the dual program of $\lambda_\pi^{\triangle}(G)$ can be written succinctly as

$$\max_{F \in \mathcal{F}(G)} \quad \max_{y_p \ge 0 : p \in \mathcal{P}(K_n)} \lambda_2 \left(\Pi^{-\frac{1}{2}} \left(L_{\mathrm{sym}}(F) - \sum_p y_p T_p \right) \Pi^{-\frac{1}{2}} \right). \tag{4.1}$$

4.1.1 Dual Program as Expander Flow

For our primal-dual algorithm, we further rewrite the dual program in (4.1) to a form that is consistent with the expander flow formulation in (2.2), by considering the demand graph of the circulation F.

We say $f = \{f_p\}_{p \in \mathcal{P}(G)}$ is a flow path decomposition of F if $F(e) = \sum_{p \ni e} f_p$ for all $e \in E$. The demand graph D of f is defined such that $D(i,j) = \sum_{p \in \mathcal{P}_G(i,j)} f_p$ for all $i, j \in V$, where $\mathcal{P}_G(i,j)$ denotes the set of directed paths from i to j in G. Note that the demand graph D of a flow path decomposition of a circulation F is Eulerian. We will use the following formulation of the dual program of $\lambda_\pi^{\triangle}$.

Lemma 4.1 (Dual Program of $\lambda_\pi^\triangle$)**.** *The dual program of $\lambda_\pi^\triangle(G)$ can be written as*

$$\max_{F\in\mathcal{F}(G)} \quad \max_{y_p\geq 0: p\in\mathcal{P}(K_n)} \quad \lambda_2\left(\Pi^{-1/2}\left(L_{\mathrm{sym}}(D)-\sum_p y_p T_p\right)\Pi^{-1/2}\right)$$
$$\text{subject to} \quad D \text{ is the demand graph of a flow-path decomposition of } F.$$

Note that a circulation $F\in\mathcal{F}(G)$ can have many different flow path decompositions. The trivial flow path decomposition is simply to have a path $p=(i,j)$ of length two for each edge ij, with the demand graph $D=F$. Alternatively, we can decompose F into weighted directed cycles and each cycle is expressed as the union of two paths, where each path is assigned a flow value equal to the weight of the cycle. In our primal-dual algorithm, we will build a circulation F using a demand graph D of low maximum degree so as to bound the width of the oracle, and this is the reason for the formulation Lemma 4.1.

Proof of Lemma 4.1. We show that the dual program in the statement is equivalent to that in (4.1). One direction is easy. Given a solution to (4.1), we can use the trivial flow decomposition of F to obtain a solution to the dual program in the statement.

For the other direction, given a solution to the dual program in the statement, we consider a flow-path decomposition $f=\{f_p\}_{p\in\mathcal{P}(G)}$ of $\frac{1}{2}F$ with demand graph $\frac{1}{2}D$. For any flow path $p\in\mathcal{P}(G)$, we write $T_p=L_p-L_{e(p)}$ where L_p is the Laplacian of the undirected path p and $L_{e(p)}$ is the Laplacian of the edge connecting two endpoints of the path. As $e(p)$ is simply an edge in the demand graph, it follows that

$$\sum_p f_p T_p=\sum_p f_p(L_p-L_{e(p)})=\sum_{ij}\frac{1}{2}\big(F(i,j)+F(j,i)\big)-\sum_{ij}\frac{1}{2}\big(D(i,j)+D(j,i)\big)=L_{\mathrm{sym}}(F)-L_{\mathrm{sym}}(D),$$

where the second last equality follows from the definition of the flow-path decomposition and the definition of the demand graph. Therefore,

$$L_{\mathrm{sym}}(D)-\sum_p y_p T_p \;=\; L_{\mathrm{sym}}(F)-\sum_p f_p T_p-\sum_p y_p T_p,$$

which is a solution to (4.1) with the same objective value, where the value of the dual variable for each path p is f_p+y_p. □

4.1.2 Intuition of the Dual Program

Since the dual program in Lemma 4.1 is slightly different from the expander flow formulation in (2.2) used in all previous works for undirected sparsest cut, we would like to provide some intuition about the term $-\sum_p y_p T_p$ in the objective function and how it will be used to simplify Sherman's algorithm for undirected sparsest cut.

We may interpret each $-T_p$ as a "shortcut cycle" C_p, where the edges in p have weight -1 and the edge connecting the two endpoints have weight 1. Since a shortcut cycle has only one positive

edge, any cut across the cycle has non-positive weight. Thus, adding a shortcut cycle to a graph does not increase the value of directed edge expansion. A nice way to understand that the dual program is a lower bound on the directed edge expansion is as follows:

$$\vec{\phi}_\pi(G) \gtrsim \phi_\pi(D) \geq \phi_\pi\left(D + \sum_p y_p C_p\right) \gtrsim \lambda_2\left(\Pi^{-\frac{1}{2}}\left(L_{\mathrm{sym}}(D) - \sum_p y_p T_p\right)\Pi^{-\frac{1}{2}}\right), \tag{4.2}$$

where the first inequality is by the flow argument because D is the demand graph of a circulation $F \in \mathcal{F}(G)$ (which is Eulerian and so can be considered as an undirected graph), the second inequality is by the discussion above that adding shortcut cycles doesn't increase the value of directed edge expansion, and the third inequality is by the easy direction of $\lambda_\pi^\triangle$ in Proposition 3.4.

Why would adding shortcut cycles help in obtaining a stronger lower bound? There are graphs where the easy direction of Cheeger's inequality is not tight, such that $\phi \approx \lambda_2^2$ rather than $\phi \approx \lambda_2$. The prototypical example is a long path p, where every edge in the path is short in its spectral embedding, which heavily violates the ℓ_2^2 triangle inequality. So, intuitively, given an embedding of the vertices, we would like to add shortcut cycles along the paths that heavily violate ℓ_2^2 triangle inequalities, so as to increase the objective of this embedding in the hope to improve the lower bound provided by the second eigenvalue, while not decreasing the objective value of sparsest cut of D by much. Thus, the dual program of $\lambda_\pi^\triangle(G)$ can be intuitively understood as finding the best way to add these shortcut cycles to prove the strongest spectral lower bound. This interpretation is also consistent with the primal program $\lambda_\pi^\triangle(G)$ in which we add triangle inequalities to the spectral program. In our primal-dual algorithm for Theorem 1.5 that we will present in the next subsection, we will indeed add shortcut cycles for paths that heavily violate the ℓ_2^2 triangle inequalities in the embedding.

4.2 Regret Minimization for Approximating Directed Edge Expansion

As in the work by Arora and Kale [AK07] described in Section 2.2.6 and Section 2.3.2, we use the regret bound in Theorem 2.5 to design a primal-dual algorithm for approximating directed edge expansion. The setup is to either certify that the optimal value to $\lambda_\pi^\triangle(G)$ is at least $\Omega(1/\kappa)$ by constructing a solution to the dual program in Lemma 4.1, or to find a cut of expansion at most $O(\sqrt{\log n}/\kappa)$ for some parameter κ. Doing binary search on κ will give us a $O(\sqrt{\log n})$-approximation algorithm.

In each iteration, the algorithm uses the density matrix X_t given by the matrix multiplicative weight update algorithm as a candidate primal solution to $\lambda_\pi^\triangle(G)$. To build a dual solution to Lemma 4.1, in each iteration t, the oracle tries to either

1. find a circulation f with demand graph D such that $\left\langle \Pi^{-\frac{1}{2}} L_{\mathrm{sym}}(D)\Pi^{-\frac{1}{2}}, X_t \right\rangle$ is large (i.e. send a lot of flow between vertices that are far apart in the geometric embedding defined by X_t) and $\|L_{\mathrm{sym}}(D)\|$ is small (i.e. the demand graph has small maximum degree), and set the feedback matrix $M_t := \Pi^{-\frac{1}{2}} L_{\mathrm{sym}}(D)\Pi^{-\frac{1}{2}}$, or

2. find paths $p_1, \dots, p_k$ and weights $y_1, \dots, y_k$ such that $-\left\langle \Pi^{-\frac{1}{2}}\left(\sum_i y_i T_{p_i}\right)\Pi^{-\frac{1}{2}}, X_t \right\rangle$ is large (i.e. paths along which the triangle inequality is violated heavily) and $\left\|\Pi^{-\frac{1}{2}}\left(\sum_i y_i T_{p_i}\right)\Pi^{-\frac{1}{2}}\right\|$ is small (i.e. the union of these paths found have small total degree) and set the feedback matrix $M_t := \Pi^{-\frac{1}{2}}\left(\sum_i y_i T_{p_i}\right)\Pi^{-\frac{1}{2}}$.

If the oracle succeeds for $T = O(\rho^2 \log n)$ iterations, where $\rho \geq \max_{t \leq T} \|M_t\|$, then the regret bound in Theorem 2.5 would imply that $\lambda_2(\frac{1}{T}\sum_{i=1}^{T} M_t)$ is large, and thus we found a solution to the dual program in Lemma 4.1 with large objective value. Otherwise, if the oracle fails to find the above objects in some iteration, then the oracle must return a sparse cut S. Above is the high level description of the algorithm, while below is the precise description of the algorithm.

Algorithm 3 Regret Minimization for Directed Sparsest Cut

Input: An edge-capacitated directed graph $G = (V, E, w)$ with vertex weights π such that $\pi(V) = 1$; step size $\eta \in (0, 1)$, width bound $\rho \in \mathbb{R}_+$, congestion parameter $\kappa \in \mathbb{R}_+$, and approximation factor $\alpha \in \mathbb{R}_+$.

Output: Either a sparse cut S, or a solution $\overline{M}$ to the dual program in Lemma 4.1.

Initialization: $X_0 = \frac{1}{n-1}(I - \Pi^{\frac{1}{2}}\mathbb{1}\mathbb{1}^\top\Pi^{\frac{1}{2}})$.

For $t = 0$ to $T - 1$:

1. Given $X_t \succcurlyeq 0$ such that $\mathrm{tr}(X_t) = 1$ and $X_t \perp \Pi^{\frac{1}{2}}\mathbb{1}$, let $Y_t := \Pi^{-\frac{1}{2}} X_t \Pi^{-\frac{1}{2}}$ and $v_1, \dots, v_n$ be the Gram decomposition of Y_t.

2. **(Oracle)** Do one of the following:

   (a) Find a circulation f on G with congestion κ and demand graph D such that $\langle L_{\mathrm{sym}}(D), Y_t\rangle \geq 1$ and $L_{\mathrm{sym}}(D) \preccurlyeq \rho \cdot \Pi$. If this succeeds, set $M_t := \Pi^{-\frac{1}{2}} L_{\mathrm{sym}}(D) \Pi^{-\frac{1}{2}}$.

   (b) Find paths $p_1, \dots, p_k$ in K_n and weights $y_1, \dots, y_k \geq 0$ such that $\langle \sum_i y_i T_{p_i}, Y_t\rangle \leq -1$ and that $-\rho \cdot \Pi \preccurlyeq \sum_i y_i T_{p_i} \preccurlyeq \rho \cdot \Pi$. If this succeeds, set $M_t := -\Pi^{-\frac{1}{2}}(\sum_i y_i T_{p_i})\Pi^{-\frac{1}{2}}$.

   (c) If both cases (a) and (b) fail, then we say that Oracle fails. In this case, find a cut $S \subseteq V$ such that $\vec{\phi}_\pi(S) = O(\alpha/\kappa)$. Return S and terminate the algorithm.

3. If Oracle succeeds, update $X'_{t+1} := \exp\left(-\frac{\eta}{\rho}\sum_{i=0}^{t} M_i\right)$. Let X_{t+1} be obtained from X'_{t+1} by projecting it onto the space orthogonal to $\Pi^{\frac{1}{2}}\mathbb{1}$ and scaling it to have trace 1.

Return the average feedback matrix $\overline{M} := \frac{1}{T}\sum_{t=0}^{T-1} M_t$.

We analyze Algorithm 3 assuming that there is a black-box algorithm for Oracle.

Lemma 4.2 (Regret Minimization Algorithm)**.** *Suppose there is a black-box algorithm for Oracle. Set $\eta = \Theta(1/\rho)$. After $T = \Theta(\rho^2 \log n)$ iterations, Algorithm 3 either certifies that $\vec{\phi}_\pi(G) \geq \Omega(1/\kappa)$ or finds a cut $S \subseteq V$ with $\vec{\phi}_\pi(S) \leq O(\alpha/\kappa)$.*

Proof. First, suppose Oracle succeeds for $T = \Theta(\rho^2 \log n)$ iterations. By applying the general regret

bound (2.6) in Theorem 2.5 restricting to the subspace orthogonal to $\Pi^{\frac{1}{2}}\mathbf{1}$, it follows that

$$\lambda_2(\overline{M}) \geq \frac{1}{T}\sum_{t=0}^{T-1}\langle M_t, X_t\rangle - \eta\rho - \frac{\rho\log n}{\eta T} = \frac{1}{T}\sum_{t=0}^{T-1}\langle \Pi^{\frac{1}{2}} M_t \Pi^{\frac{1}{2}}, Y_t\rangle - \eta\rho - \frac{\rho\log n}{\eta T} \geq 1 - \eta\rho - \frac{\rho\log n}{\eta T},$$

where the last inequality follows from the fact that cases (a) and (b) in Oracle both imply that $\langle \Pi^{\frac{1}{2}} M_t \Pi^{\frac{1}{2}}, Y_t\rangle \geq 1$. By choosing suitable implicit constants in the $\Theta(\cdot)$ for T and η,

$$\lambda_2(\overline{M}) \geq 1 - \eta\rho - \frac{\rho\log n}{\eta T} \geq 1 - \frac{1}{4} - \frac{1}{4} \geq \frac{1}{2}.$$

Note that the average feedback matrix $\overline{M}$ is a Laplacian of the form $\Pi^{-\frac{1}{2}}(L_{\mathrm{sym}}(D) - \sum_p y_p T_p)\Pi^{-\frac{1}{2}}$, where D is the demand graph of a circulation f with congestion κ (as f is the average of circulations each with congestion κ). Therefore, by scaling down f, D, and all y_p by a factor of κ, we obtain a solution to the dual program of $\lambda_\pi^{\triangle}(G)$ in Lemma 4.1 with objective value $\Omega(1/\kappa)$, and this certifies that $\vec{\phi}_\pi(G) \gtrsim 1/\kappa$.

On the other hand, if Oracle fails at some iteration, then it outputs a cut S with $\vec{\phi}_\pi(S) \leq O(\alpha/\kappa)$. □

In Lemma 4.2, we have set the values of T and η in relation to the width bound ρ, to obtain the desired approximation guarantee $O(\alpha)$. The undetermined parameters in the algorithm are ρ and α. We would like to set them to be as small as possible, so as to minimize both the runtime (as the number of iterations T will be minimized) and the approximation ratio of the algorithm, while the Oracle can still be implemented efficiently. This is the goal in Section 4.3 and Section 4.4.

4.3 Geometric Results for Implementation of Oracle

To implement the Oracle in Algorithm 3, we need the results proved in [ARV09, AK07, Kal07, She09] about geometric embeddings.

Let $v_1, \ldots, v_n$ be the Gram decomposition of Y_t in step (1) of Algorithm 3. Note that the trace condition in step (1) implies that $\sum_i \pi(i)\cdot\|v_i\|^2 = 1$, and the null-space condition in step (1) implies that $\sum_i \pi(i)\cdot v_i = 0$. It follows from Fact 3.7 that $\sum_{i<j}\pi(i)\cdot\pi(j)\cdot\|v_i - v_j\|^2 = 1$. As in the SDP rounding result in Section 3.2, we consider the following two cases of the geometric embedding.

Proposition 4.3 (Dichotomy of Embeddings)**.** *Let $v_1, \ldots, v_n$ be vectors in $\mathbb{R}^n$ satisfying the condition that $\sum_{i<j}\pi(i)\cdot\pi(j)\cdot\|v_i - v_j\|^2 = 1$. One of the following two cases must hold:*

(i) Large Core*: There exists a vector v such that $\pi(B(v, \frac{1}{2\sqrt{10}})) \geq 1/4$.*

(ii) Well Spread*: There is a vector w such that if we apply the transformation $u_i := c(v_i - w)$ for $1 \leq i \leq n$ for some constant $c > 0$, then there exists a subset U of vectors with (i) $\pi(U) \gtrsim 1$, (ii) $\|u_i\| \leq 1$ for all $i \in U$, and (iii) $\sum_{i,j\in U}\pi(i)\cdot\pi(j)\cdot\|u_i - u_j\|^2 \gtrsim 1$.*

Note that a version of Proposition 4.3 for uniform vertex weights was already proved in [Kal07]. The weighted case follows by a simple reduction which we will defer to the appendix.

4.3.1 Large Core Case

This is the easy case where we can implement the oracle to either return a circulation in step 2(a) or a sparse cut in step 2(c) of Algorithm 3, using a result in Section 3.1 for metric rounding proved by the max-flow min-cut theorem.

Lemma 4.4 (Oracle in Large Core Case)**.** *In the large core case in Proposition 4.3, there is an algorithm that, using two max-flow computations, implements Oracle in Algorithm 3 so that it either computes a cut $S \subseteq V$ with $\vec{\phi}_\pi(S) \le O(1/\kappa)$ or obtains a circulation f whose demand graph D satisfies $\langle L_{\mathrm{sym}}(D), Y_t \rangle \ge 1$ and $L_{\mathrm{sym}}(D) \preccurlyeq O(1) \cdot \Pi$.*

Proof. Let v_j be a vector with $\pi\big(B\big(v_j, \frac{1}{2\sqrt{10}}\big)\big) \ge \frac{1}{4}$. By triangle inequality, for any $v_i \in B\big(v_j, \frac{1}{2\sqrt{10}}\big)$, it holds that $\pi\big(B\big(v_i, \frac{1}{\sqrt{10}}\big)\big) \ge \frac{1}{4}$. So, by random sampling, we can find in $O(n \log n)$ time a vector v_{i^*} such that $\pi\big(B\big(v_{i^*}, \frac{1}{\sqrt{10}}\big)\big) \ge \frac{1}{4}$ with high probability.

After finding such a vector v_{i^*}, we run Algorithm 2 (Bidirectional Max-Flow) with $L := B\big(v_{i^*}, \frac{1}{\sqrt{10}}\big)$, $R := \overline{L}$, and κ the same as given in Algorithm 3. In order to choose the flow value parameter β appropriately, we would first lower bound the total distance to L. Applying Lemma 3.8 with the set L as chosen and the semi-metric $d(i,j) := \|v_i - v_j\|^2$, which satisfies the s-relaxed triangle inequality for $s = 2$, it follows that

$$\sum_{j \in R} \pi(j) \cdot d(j, L) = \sum_{j \notin L} \pi(j) \cdot d(j, L) \ge \frac{1}{4} - \frac{1}{10} = \frac{3}{20}.$$

Apply Algorithm 2 with $\beta = 20/3$. On the one hand, if the algorithm returns a circulation f, then its demand graph D satisfies

$$\langle L_{\mathrm{sym}}(D), Y_t \rangle = \sum_{i \in L, j \in R} \frac{1}{2}\big(D(i,j) + D(j,i)\big) \|v_i - v_j\|^2 \ge \sum_{j \in R} \beta \cdot \pi(j) \cdot d(j, L) \ge 1, \tag{4.3}$$

where the first inequality follows from the fact that each vertex $j \in R$ has capacity $\beta \cdot \pi(j)$ and the flow saturates all such capacities. The capacities also imply that the normalized Laplacian of the demand graph satisfies $\Pi^{-1/2} L_{\mathrm{sym}}(D) \Pi^{-1/2} \preccurlyeq 2\beta \cdot I$, or equivalently $L_{\mathrm{sym}}(D) \preccurlyeq 2\beta \cdot \Pi$.

On the other hand, if the algorithm returns a cut $S \subseteq V$, then by Lemma 3.3 (unsaturated case) we have $\vec{\phi}_\pi(S) \le r\beta/\kappa = 20/\kappa$, since $r := \pi(R)/\pi(L) \le 3$. □

To summarize, in the large core case, there is an efficient oracle that achieves approximation factor $\alpha = O(1)$ and width bound $\rho = O(1)$.

4.3.2 Well Spread Case

The well spread case is much more involved, for which we need the correlated chaining theorem of Sherman [She09]. In this subsection, we present the background for the correlated chaining theorem, and we defer the implementation of the oracle in the well spread case to the next subsection.

The idea of chaining matchings was the main ingredient that led to the $O(\sqrt{\log n})$ approximation result of [ARV09], and was also used in [AK07, Kal07] to compute expander flows to solve the dual program. The main idea was to show that if for many directions, there is a large matching between embedding vertices that are well-separated along that direction but close to each other in the overall embedding, then $O(\sqrt{\log n})$ such matchings can be chained together to form a path that violates the ℓ_2^2 triangle inequality. In [She09], this was improved so that instead of finding one such violating path, we can find *many* such paths *efficiently* with good probability through a simple sampling process.

To handle the arbitrary vertex weights $\pi : V \to \mathbb{R}_+$, we slightly modify Sherman's definitions and results and make use of a version of his main theorem for fractional matchings instead of integral matchings.

Definition 4.5 (π-Fractional Matching)**.** *Let V be a vertex set with weights $\pi : V \to \mathbb{R}^+$. We say that M is a π-fractional matching if M is a weighted directed subgraph of K_n with edge weights $M(i,j) \in \mathbb{R}_{\geq 0}$ for $i, j \in V$, satisfying the property that each vertex $i \in V$ has either only incoming edges or only outgoing edges and has degree at most $\pi(i)$. The total weight of M is denoted by $w(M) := \sum_{i,j} M(i,j)$.*

Definition 4.6 (Fractional Matching Cover)**.** *A (σ, δ)-matching cover is a function assigning a π-fractional matchings $\mathcal{M}_u$ to each vector $u \in \mathbb{R}^n$ satisfying the following properties:*

(i) $\forall (i,j) \in \text{supp}(\mathcal{M}_u)$, $\langle v_j - v_i, u \rangle \geq \sigma$;

(ii) $\mathcal{M}_u(i,j) = \mathcal{M}_{-u}(j,i)$ *for all* $u \in \mathbb{R}^n$;

(iii) $\mathbb{E}_u[w(\mathcal{M}_u)] \geq \delta \cdot \pi(V)$ *where* $u \sim \mathcal{N}(0, I)$.

We define formally what it means to "chain together" fractional matchings.

Definition 4.7 (Chained Matchings)**.** *Let $\mathcal{M}$ be a fractional matching cover. Given vectors $u_1, \ldots, u_\ell \in \mathbb{R}^n$, we define $\mathcal{M}(u_1, \ldots, u_\ell)$ to keep track of the paths that result from chaining together matchings $\mathcal{M}_{u_1}, \ldots, \mathcal{M}_{u_\ell}$. Define $\mathcal{M}(u_1, \ldots, u_\ell) := (\mathcal{M}_{u_1,\ldots,u_\ell}, \mathcal{P}_{u_1,\ldots,u_\ell} = \{f_p, p\}_{p \in \mathcal{P}(K_n)})$, where each $p \in \mathcal{P}_{u_1,\ldots,u_\ell}$ is a weighted path of length $\ell + 1$ with weight f_p and $\mathcal{M}_{u_1,\ldots,u_\ell}$ is the graph with $\mathcal{M}_{u_1,\ldots,u_\ell}(i,j) = \sum_{p \in \mathcal{P}_{u_1,\ldots,u_\ell} \cap \mathcal{P}_{K_n}(i,j)} f_p$ being the total weight on paths in $\mathcal{P}_{u_1,\ldots u_{\ell-1}}$ going from vertices i to j. The paths and weights in $\mathcal{P}_{u_1,\ldots,u_l}$ are defined recursively in the following algorithm.*

Construction of $\mathcal{P}_{u_1,\ldots,u_\ell}$

- *If $\ell = 1$, then $\mathcal{P}_{u_1} = \{\mathcal{M}_{u_1}(i,j), (i,j) \mid \mathcal{M}_{u_1}(i,j) > 0\}$. That is, the paths are simply the edges in $\mathcal{M}_{u_1}$ with the corresponding weights.*
- *If $\ell > 1$, then for each $q \in \mathcal{P}_{u_1,\ldots,u_{\ell-1}}$ where $q \in \mathcal{P}(i,j)$, run the following loop.*
  1. *While $f_q > 0$ and there exists $j' \in V$ with $\mathcal{M}_{u_\ell}(j,j') > 0$, let p be the path obtained by extending q by j' and add p to $\mathcal{P}_{u_1,\ldots,u_\ell}$ with weight $f_p = \min\{\mathcal{M}_{u_\ell}(j,j'), f_q\}$.*
  2. *Decrement both f_q and $\mathcal{M}_{u_\ell}(j,j')$ by $\min\{\mathcal{M}_{u_\ell}(j,j'), f_q\}$.*

The following simple claim will be used in the runtime analysis of the oracle.

Claim 4.8. *Suppose that each matching $\mathcal{M}_u$ has at most m edges. Then $\mathcal{P}_{u_1,\ldots,u_\ell}$ has at most $m\ell$ paths and can be constructed in $O(m\ell^2)$ time given oracle access to $\mathcal{M}_{u_1},\ldots,\mathcal{M}_{u_\ell}$.*

Proof. Clearly, the claim holds true for $\ell = 1$. Now assume by induction that the claim holds for $\ell - 1$. It suffices to bound the number of times we run the while loop in which we add a new path p to $\mathcal{P}_{u_1,\ldots,u_\ell}$. Since in each iteration of the while loop, we either remove a path from $\mathcal{P}_{u_1,\ldots,u_{\ell-1}}$ or an edge from $\mathcal{M}_{u_\ell}$, it can run for at most $m(\ell-1)+m = m\ell$ iterations. Thus $\mathcal{P}_{u_1,\ldots,u_\ell}$ has at most $m\ell$ paths. □

Note that when π is uniform, then the definition of π-fractional matching cover is the same as the matching cover from [She09, Definition 5.2.1], in which all edges have weight 0 or 1. Now we present the main theorem that we will use to implement the oracle in Algorithm 3 in the well spread case.

Theorem 4.9 (Sherman's Chaining Theorem)**.** *For any small enough constant l, there is a $k = O(\sqrt{l \log n})$ and an efficiently sample-able distribution $\mathcal{D}$ over vectors $(u_1,\ldots,u_k) \subseteq \mathbb{R}^d$ with the following property: if $\mathcal{M}$ is a $(\Omega(1),\Omega(1))$-fractional matching cover for the set of embedded vertices V, then the expected total weight of paths in $\mathcal{M}(u_1,\ldots,u_k)$ between vertices i,j with $\|v_i - v_j\|^2 \geq l$ is at least $e^{-O(k^2)} \cdot \pi(V)$ when $(u_1,\ldots,u_k)$ is sampled from $\mathcal{D}$.*

The uniform π version of this theorem was proved in [She09, Theorem 5.2.3]. The π-weighted version follows from a simple reduction to the uniform case, which we will defer to Appendix B.

4.4 Fast Implementation of Oracle for Well-Spread Case

With Sherman's chaining theorem, we are ready to implement the oracle in Algorithm 3 in the well spread case in this subsection, with approximation ratio $O\left(\sqrt{\frac{\log n}{\epsilon}}\right)$ and width bound $\widetilde{O}\left(\frac{n^\epsilon}{\epsilon^{3/2}}\right)$.

Proposition 4.10 (Oracle in Well Spread Case)**.** *Let $\epsilon > 0$ be a small enough constant. In the well-spread case in Proposition 4.3, there is a randomized implementation of Oracle in Algorithm 3 that, with high probability, using $\widetilde{O}(n^\epsilon)$ max-flow computations, either outputs a feedback matrix M_t with $\langle M_t, X_t\rangle \geq 1$ and $\|M_t\| \leq \widetilde{O}\left(\frac{n^\epsilon}{\epsilon^{3/2}}\right)$, or returns a cut $S \subseteq V$ with $\vec{\phi}_\pi(S) \leq O\left(\frac{1}{\kappa}\sqrt{\frac{\log n}{\epsilon}}\right)$.*

4.4.1 Overview

The basic subroutine, as in [AK07, Kal07, She09], is the Project Max-Flow algorithm (Algorithm 4), where we project the vectors along a random direction and set up a bi-directional flow problem between two subsets L and R that are far apart in the projection. If such a bi-directional flow cannot be sent, then we will show that any min-cut is a sparse cut by Lemma 3.3, and so Algorithm 3 can terminate in step 2(c). If such a bi-directional can be sent, with the additional property that many flow paths are between vertices that are far apart in the embedding such that

$$\langle L_{\text{sym}}(D), Y_t\rangle = \frac{1}{2}\sum_{i\in L, j\in R}\big(D(i,j)+D(j,i)\big)\cdot\|v_i - v_j\|^2 \geq 1,$$

then we will show that the oracle succeeds in finding a circulation in step 2(a) of Algorithm 3, and so the algorithm can proceed to the next iteration.

The new observation is that if for many random directions, such a bi-directional flow can be sent but its demand graph does not satisfy $\langle L_{\text{sym}}(D), Y_t \rangle \geq 1$, then we can construct a fractional matching cover and use Sherman's chaining theorem to find many paths that violate the triangle inequality heavily. Thus, the oracle succeeds in finding many violating paths in step 2(b) of Algorithm 3, and so the algorithm can proceed to the next iteration. So, as long as such a bi-directional flow can be sent, then either step 2(a) or 2(b) succeeds in giving a good feedback matrix for the regret minimization algorithm.

This is the main difference with previous algorithms in [AK07, She09], where a multi-commodity flow computation is needed to guarantee a condition similar to $\langle L_{\text{sym}}(D), Y_t \rangle \geq 1$ for the oracle to succeed. We remark that using violating paths as feedback is only possible because of the stronger unmodified dual program in Lemma 4.1, but not in the usual expander flow formulation corresponding to the constrained dual program as in in (2.2).

4.4.2 Project Max-Flow Algorithm

In the well spread case in Proposition 4.3, we will only focus on the vectors in the subset U, with $\pi(U) \gtrsim \pi(V)$ and $\|v_i\| \leq 1$ for $i \in U$ and $\sum_{i,j \in U} \pi(i) \cdot \pi(j) \cdot \|v_i - v_j\|^2 \gtrsim 1$.

Algorithm 4 Project Max-Flow (G, u, c, β, κ)

Input: An edge-capacitated directed graph $G = (V, E, w)$ with vertex weights $\pi : V \to \mathbb{R}_+$, an embedding $v_1, \ldots, v_{|U|} \in \mathbb{R}^n$ of the vertices in U, vector $u \in \mathbb{R}^n$, small constant c, congestion parameter κ, and flow value parameter β.

1. Order the vertices $i \in U$ by the values of $\langle u, v_i \rangle$. Let L be the l smallest vertices in this ordering, where l is the smallest integer such that $\pi(L) \geq c \cdot \pi(V)$. Let R be the r largest vertices in this ordering, where r is the smallest integer such that $\pi(R) \geq c \cdot \pi(V)$.

2. Compute a bidirectional max-flow using Algorithm 2 on (L, R, β, κ) to obtain either a cut $S \subseteq V$ or a circulation f in G with congestion κ.

The following lemma shows that with constant probability over the random direction u, the sets L and R will be well-separated along the direction u.

Lemma 4.11 (Good Direction)**.** *Let $v_1, \ldots, v_{|U|}$ be a set of vectors that satisfies (i) $\pi(U) \gtrsim 1$, (ii) $\|v_i\| \leq 1$ for all $i \in U$ and (iii) $\sum_{i,j \in U} \pi(i) \cdot \pi(j) \cdot \|v_i - v_j\|^2 \gtrsim 1$. Then there exist positive constants γ and σ and c such that if we sample random $u \sim N(0, I)$, then with probability at least γ the sets L, R in step (1) of Algorithm 4 satisfy the condition that $\langle u, v_i - v_j \rangle \geq \sigma$ for all $i \in L, j \in R$. We say that such vectors u are good vectors.*

The proof is a simple reduction to the uniform π case proven in [Kal07, Lemma14] and [She09, Lemma 5.3.3], and so we defer to the Appendix. We remark that Lemma 4.11 is the only place in the proof of Proposition 4.10 that we use the assumption that the vectors are well spread.

For each good vector $u \in \mathbb{R}^n$, the sets L, R in step (1) of Algorithm 4 are disjoint, and so the bi-directional max-flow in Algorithm 2 is well-defined. Therefore, exactly one the following three cases must happen.

- **A:** Algorithm 4 returns a circulation f with demand graph D such that $\langle L_{\text{sym}}(D), Y_t \rangle \geq 1$.
- **B:** Algorithm 4 returns a circulation f with demand graph D such that $\langle L_{\text{sym}}(D), Y_t \rangle < 1$.
- **C:** Algorithm 4 returns a cut $S \subseteq V$.

If we are in case C for some good vector u, then we show that the primal-dual Algorithm 3 can terminate with approximation ratio $O(\beta)$. (In the proof of Proposition 4.10 that we will present later, we will set $\beta = O(\sqrt{\log n/\epsilon})$.)

Claim 4.12 (Case C)**.** *If Algorithm 4 returns a cut S for some good vector u, then $\vec{\phi}_\pi(S) = O(\beta/\kappa)$.*

Proof. Since u is good, the sets L, R are disjoint and $\pi(L), \pi(R) \geq c \cdot \pi(V)$ for some (small) constant c. Then, by Lemma 3.3 (unsaturated case), Algorithm 2 will return a set S with $\vec{\phi}_\pi(S) \leq \beta r'/\kappa \lesssim \beta/\kappa$ as $r' = \max\{1, \pi(R)/\pi(L)\} \leq 1/c$. □

If we are in case A for some good vector u, then we show that the oracle succeeds in step 2(a) of Algorithm 3 with width $\rho = O(\beta)$.

Claim 4.13 (Case A)**.** *If Algorithm 4 returns a circulation f with demand graph D for some good vector u such that $\langle L_{\text{sym}}(D), Y_t \rangle \geq 1$, then the feedback matrix $M_t := \Pi^{-\frac{1}{2}} L_{\text{sym}}(D) \Pi^{-\frac{1}{2}}$ in step 2(a) of Algorithm 3 satisfies $\|M_t\| \lesssim \beta$.*

Proof. Since u is good, the sets L, R are disjoint and $\pi(L), \pi(R) \geq c \cdot \pi(V)$ for some (small) constant c. In the bi-directional max-flow problem in Algorithm 2, each vertex i in $L \cup R$ has degree at most $r' \cdot \beta \cdot \pi(i) \lesssim \beta \cdot \pi(i)$ as $r' = \max\{1, \pi(R)/\pi(L)\} \leq 1/c$. This implies that the demand graph satisfies $L_{\text{sym}}(D) \preccurlyeq O(\beta \cdot \Pi)$, and thus $M_t = \Pi^{-\frac{1}{2}} L_{\text{sym}}(D) \Pi^{-\frac{1}{2}} \preccurlyeq O(\beta I)$. □

4.4.3 Finding Many Violating Paths

If we are in case B for some good vector u, then we show how to construct a large matching of flow paths between pairs of vertices with small embedding distance using the following algorithm.

The reason that we ignored the original direction of the paths in Algorithm 5 is that we are trying to find paths in K_n, rather than in G, that violate the triangle inequality. Thus, it is fine if the resulting violating paths from chaining together the matchings do not correspond to paths in G.

Lemma 4.14 (Case B)**.** *If Algorithm 4 returns a circulation f with demand graph D for some good vector u such that $\langle L_{\text{sym}}(D), Y_t \rangle < 1$, then Algorithm 5 returns a fractional matching $\mathcal{M}_u$ with $w(\mathcal{M}_u) \gtrsim c^2$, where each edge ij in $\mathcal{M}_u$ satisfies $\langle v_j - v_i, u \rangle \geq \sigma$ and $\|v_j - v_i\|^2 \lesssim \frac{1}{\beta c}$. Moreover, there is a randomized algorithm to compute $\mathcal{M}_u$ in expected time $O(m \log n)$.*

Algorithm 5 Matching(u)

Input: s-t flow $\vec{f}$ and t-s flow $\overleftarrow{f}$ obtained from Step 2 of Algorithm 4, with parameters (G, u, c, β, κ).

1. Decompose the two flows into at most m flow paths between sets L and R. Ignore the original direction of the paths and reorient every path from L to R. In particular, the paths we get are $(p_r, i_r, j_r, f_{p_r})_{r=1}^{k}$ where $k \le 2m$. For each $r \in [k]$, p_r is a path from $i_r \in L$ to $j_r \in R$, with weight f_{p_r}.
2. Discard any path p_r with $\langle v_{j_r} - v_{i_r}, u\rangle < \sigma$ or $\|v_{j_r} - v_{i_r}\|^2 > \frac{4}{\beta c}$.
3. Define $\mathcal{M}'_u$ so that $\mathcal{M}'_u(i,j)$ is the sum of the weights of all remaining paths from i to j. Return
$$\mathcal{M}_u := \frac{1}{\beta \cdot \max\left\{1, \pi(R)/\pi(L)\right\}} \mathcal{M}'_u.$$

Proof. Since Algorithm 4 returns a circulation f, both flows $\vec{f}$ and $\overleftarrow{f}$ from Algorithm 2 are saturating. This implies that the flow value for $\vec{f}$ and $\overleftarrow{f}$ is at least $\sum_{j\in R} \beta\cdot\pi(j) = \beta\cdot\pi(R) \ge \beta\cdot c\cdot\pi(V) = \beta\cdot c$, and thus
$$\sum_{i\in L, j\in R} \big(D(i,j) + D(j,i)\big) \ge \beta \cdot c$$
where D is the demand graph of the circulation $f = \frac{1}{2}(\vec{f} + \overleftarrow{f})$.

Next, we bound the total weight of the flow paths that we discard in step (2) of Algorithm 5. Since u is good (see Lemma 4.11), all flow paths are between vectors i, j such that $\langle v_j - v_i, u\rangle \ge \sigma$, so no paths will be discarded this way. Our assumption implies that
$$1 > \langle L_{\mathrm{sym}}(D), Y_t\rangle = \frac{1}{2} \sum_{i\in L, j\in R} \big(D(i,j) + D(j,i)\big) \cdot \|v_i - v_j\|^2,$$
and thus an average flow path is between pairs i, j with $\|v_i - v_j\|^2 < \frac{2}{\beta c}$. By Markov's inequality, at least half of the flow of f is on flow paths between i, j with $\|v_i - v_j\|^2 \le \frac{4}{\beta c}$. Therefore, we discard at most half of the flow of f in Step (2) of Algorithm 5, and hence the weight of $\mathcal{M}'_u$ is at least $\beta c/2$.

By the construction of the bidirectional flow in Algorithm 2, each source and sink vertex has degree at most $\beta \cdot \pi(i) \cdot \max\{1, \pi(R)/\pi(L)\} \le \beta \cdot \pi(i)/c$. So, scaling $\mathcal{M}'_u$ down by this factor gives a fractional matching $\mathcal{M}_u$ as defined in Definition 4.5, with $w(\mathcal{M}_u)$ at least $c^2/2$.

Finally, we bound the runtime of the algorithm. The only non-trivial step is step 1, in which we must decompose a fractional single-commodity flow into integral flow paths. The following theorem due to Lau and Kwok show that this can be done in almost linear time on expectation

Theorem 4.15. *([LRS13, Theorem 5]) Given a fractional $s-t$ flow $\vec{f}$, there is a randomized algorithm that, in $O(m \log n)$ expected time, returns a flow path decomposition $(p_r, \vec{f}_{p_r})_{r=1}^{k}$ where $k \le m$ and each p_r is a path from s to t with flow value $\vec{f}_{p_r}$ along the path.*

□

It follows that if case B happens often enough, then we can construct a fractional matching cover as defined in Definition 4.6.

Lemma 4.16. *Suppose that conditioned on u being a good vector, the probability that we are in case B is at least $1/2$. Then $\mathcal{M} = \{\mathcal{M}_u\}_{u\in\mathbb{R}^n}$ is a (σ,δ)-matching cover with $\sigma,\delta = \Omega(1)$.*

Proof. Clearly, conditions (i) and (ii) in Definition 4.6 are met. As long as u is a good vector, $w(\mathcal{M}_u) \gtrsim c^2$ by Lemma 4.14. As a random vector is a good vector with probability at least γ by Lemma 4.11, we conclude that $\mathcal{M} = \{\mathcal{M}_u\}_{u\in\mathbb{R}^n}$ is a (σ,δ)-matching cover with $\sigma = \Omega(1)$ and $\delta = \gamma\cdot c^2 = \Omega(1)$. □

We apply Sherman's chaining theorem on the matching cover to construct many violating paths for step 2(b) in Algorithm 3.

Lemma 4.17 (Violating Paths)**.** *Given the $(\Omega(1),\Omega(1))$-matching cover $\mathcal{M}$ in Lemma 4.16, by setting $\beta = O\Big(\sqrt{\frac{\log n}{\epsilon}}\Big)$, there is a randomized algorithm using $O(\sqrt{\epsilon\log n})$ max-flow computations to find paths $p_1,\dots,p_s$ with weight $f_{p_1},\dots,f_{p_s}$, so that the feedback matrix $M_t := -\Pi^{-\frac12}\big(\sum_{r=1}^s f_{p_r}\cdot T_{p_r}\big)\Pi^{-\frac12}$ satisfies $\langle M_t, X_t\rangle \ge 1$ and $\|M_t\| = \widetilde{O}(n^\epsilon/\epsilon^{3/2})$, with success probability $\Omega(n^{-\epsilon})$.*

Proof. We apply Sherman's Theorem 4.9 on $\mathcal{M}$ with $l = \Theta(\epsilon)$ and $k = \Theta(\sqrt{\epsilon\log n})$ to obtain an efficiently sample-able distribution $\mathcal{D}$ over $(u_1,\dots,u_k)$ so that the expected total weight of paths in $\mathcal{M}(u_1,\dots,u_k)$ between i,j with $\|v_i - v_j\|^2 \ge l = \Theta(\epsilon)$ is at least $e^{-O(k^2)}\cdot\pi(V) = O(n^{-\epsilon})$. Since the total weight is at most 1, by a reverse application of Markov's inequality, we will find vectors $u_1,\dots,u_k$ where the weight of such good paths in $\mathcal{M}(u_1,\dots,u_k)$ is at least $\frac12 n^{-\epsilon}$, with probability at least $n^{-\epsilon}$. With such $u_1,\dots,u_k$, by Claim 4.8, we can find paths $p_1,\dots,p_s$ with $\sum_{r=1}^s f_{p_r} \ge \frac12 n^{-\epsilon}$ in $O(mk^2) = \widetilde{O}(m)$ time, such that for $1\le r\le s$ the path $p_r = (v_{i_1},\dots,v_{i_{k+1}})$ satisfies

$$\sum_{j=1}^k \big\|v_{i_j} - v_{i_{j+1}}\big\|^2 \lesssim \frac{k}{\beta\cdot c} \lesssim \frac{\sqrt{\epsilon\log n}}{\beta\cdot c} \quad \text{but} \quad \big\|v_{i_1} - v_{i_{k+1}}\big\|^2 \ge l \gtrsim \epsilon,$$

where the first inequality is by the property that each edge ij in each fractional matching has $\|v_i - v_j\|^2 \le \frac{4}{\beta c}$ in Lemma 4.14. Thus, by choosing $\beta = \Theta\Big(\frac{\sqrt{\epsilon\log n}}{c\cdot l}\Big) = \Theta\Big(\sqrt{\frac{\log n}{\epsilon}}\Big)$ with the appropriate constant, the paths violate triangle inequality so that

$$\langle T_{p_r}, Y_t\rangle = \sum_{j=1}^k \big\|v_{i_j} - v_{i_{j+1}}\big\|^2 - \big\|v_{i_1} - v_{i_{k+1}}\big\|^2 \lesssim -\epsilon \quad\Longrightarrow\quad \Big\langle \sum_{r=1}^s f_{p_r} T_{p_r}, Y_t\Big\rangle \lesssim -\epsilon\cdot n^{-\epsilon}$$

Setting $y = \Theta(n^\epsilon/\epsilon)$ with the appropriate implicit constant and the feedback matrix $M_t := -\Pi^{-\frac12}(y\sum_{r=1}^s f_{p_r}T_{p_r})\Pi^{-\frac12}$, we ensure that $\langle M_t, X_t\rangle = \langle -y\sum_{r=1}^s f_{p_r}T_{p_r}, Y_t\rangle \ge 1$.

Finally, we bound $\|M_t\|$ to bound the width of the oracle. Note that the edges in these violating paths form a subgraph of the graph $\mathcal{M}_{u_1}\cup\cdots\cup\mathcal{M}_{u_k}$. In each of these graphs, the total degree of

vertex i is at most $2\beta \cdot \pi(i) \le 2\beta$, and so the total degree of each vertex i in the union is at most $2\beta \cdot k$. Thus, $\|M_t\| \le 2y \cdot \beta \cdot k = \widetilde{O}(n^\epsilon/\epsilon^{3/2})$. □

4.4.4 Proof of Proposition 4.10

We are ready to put together the results in this subsection to finish the proof of Proposition 4.10. Set $\beta = O\left(\sqrt{\frac{\log n}{\epsilon}}\right)$. Recall that, by Lemma 4.11, there is a positive constant γ such that $u \sim N(0, I)$ is a good vector with probability at least γ.

Suppose that when conditioned on u being a good vector, we are in case C of Algorithm 4 with probability at least $\frac{1}{4}$. This means that, with probability at least $\frac{\gamma}{4} = \Omega(1)$, a set S with $\vec{\phi}_\pi(S) = O\left(\frac{\beta}{\kappa}\right) = O\left(\frac{1}{\kappa}\sqrt{\frac{\log n}{\epsilon}}\right)$ will be returned by Claim 4.12. Therefore, after $O(\log n)$ independent samples of u, such a sparse cut will be returned with high probability, and so Algorithm 3 can be terminated.

Similarly, suppose that when conditioned on u being a good vector, we are in case A of Algorithm 4 with probability at least $1/4$. This means that, with probability $\Omega(1)$, a feedback matrix $M_t := \Pi^{-\frac{1}{2}} L_{\mathrm{sym}}(D) \Pi^{-\frac{1}{2}}$ from a circulation f with demand graph D can be returned with $\langle M_t, X_t \rangle \ge 1$ and $\|M_t\| \lesssim \beta \lesssim \sqrt{\frac{\log n}{\epsilon}}$ by Claim 4.13. Therefore, after $O(\log n)$ independent samples of u, such a circulation will be returned with high probability, and so Algorithm 3 can proceed to the next iteration.

Otherwise, suppose that when conditioned on u being a good vector, we are in case B of Algorithm 4 with probability at least $1/2$. By Lemma 4.17, we can use Sherman's result to chain together $O(\sqrt{\epsilon \log n})$ such flows to find violating paths $p_1, \ldots, p_s$ so that the feedback matrix $M_t := \Pi^{-\frac{1}{2}} (\sum_r^2 f_{p_r} \cdot T_{p_r}) \Pi^{-\frac{1}{2}}$ satisfies $\langle M_t, X_t \rangle \ge 1$ and $\|M_t\| = \widetilde{O}(n^\epsilon/\epsilon^{3/2})$, with probability at least $\Omega(n^{-\epsilon})$. Therefore, after $\widetilde{O}(n^\epsilon)$ chaining attempts using a total of $\widetilde{O}(n^\epsilon)$ max-flow computations, such violating paths will be returned with high probability, and Algorithm 3 can proceed to the next iteration.

These covers all the cases. The width and the runtime of the oracle are dominated by the step of finding violating paths.

4.5 Main Result and Corollary

In this subsection, we prove Theorem 1.5 and Proposition 1.4.

Proof of Theorem 1.5. By Lemma 4.2, if there is an oracle with width ρ and approximation factor α, then the regret minimization Algorithm 3 either certifies that $\vec{\phi}_\pi(G) \ge \Omega(1/\kappa)$ or finds a cut $S \subseteq V$ with $\vec{\phi}_\pi(S) \le O(\alpha/\kappa)$ for a given κ in $O(\rho^2 \log n)$ iterations. Combining the oracle in the large core case in Lemma 4.4 and the oracle in the well spread case in Proposition 4.10, we obtain an oracle with width $\rho = O(n^\epsilon/\epsilon^{3/2})$ and approximation ratio $\alpha = O(\sqrt{\log n/\epsilon})$. Therefore, by doing binary search on κ, we can obtain a $O(\sqrt{\log n/\epsilon})$-approximation algorithm by running a total of $\widetilde{O}(n^{2\epsilon})$ iterations of MMWU in Algorithm 3.

Now, we will bound the runtime of each iteration. By Proposition 4.10, each iteration requires $\widetilde{O}(n^\epsilon)$ max-flow computations. After each max-flow computation, we need to perform a flow-path decomposition either in Algorithm 5 or by computing the edges of the demand graph D_t, which can be implemented in expected $O(m \log n)$ time by Theorem 4.15. In addition, each iteration requires the use of a matrix exponential, whose computation is too long. Thus, instead of computing Y_t, we will approximately compute its Gram decomposition using the following lemma, whose proof we will defer to Appendix B:

Lemma 4.18 (Matrix Exponential Computation)**.** *Let $v_1, \dots v_n$ be the Gram decomposition of the matrix Y_t in step 1 of Algorithm 3. There is a randomized algorithm that, in $\widetilde{O}(\rho m/\delta^2)$ time, computes vectors $\hat{v}_1, \dots \hat{v}_n \in \mathbb{R}^d$ for $d = O(\log n/\delta^2)$ such that with probability at least $1 - n^{-1}$,*

$$\|\hat{v}_i = \hat{v}_j\|^2 \in (1 \pm \delta) \|v_i - v_j\|^2 \pm n^{-\Omega(1)} \quad \forall i, j \in V$$

In particular, Lemma 4.18 implies that if L is any Laplacian matrix (possibly with negative edge-weights) satisfying $\langle L, \sum_i \hat{v}_i \hat{v}_i^\top \rangle \geq 1$, then $\langle L, Y_t \rangle \geq 1 - 2\delta$. Since all our feedback matrices are always of the form $\Pi^{-1/2} L \Pi^{-1/2}$ for some Laplacian L, it suffices to use $\hat{v}_1 \dots \hat{v}_n$ as the embedding vectors at step 1 of the algorithm in order to ensure that $\langle M_t, X_t \rangle \geq 1 - 2\delta$ for every iteration t even though we never have to explicitly compute X_t. Since $\rho = \widetilde{O}(n^\epsilon)$, the overall runtime of each iteration is dominated by the runtime of $\widetilde{O}(n^\epsilon)$ maxflow computations. □

An interesting corollary is about a dual certificate using circulations in Proposition 1.4.

Proof of Proposition 1.4. Apply Lemma 4.2 with $\kappa \gtrsim \sqrt{\frac{\log n}{\epsilon}}/\vec{\phi}_\pi(G)$ for a suitable implicit constant, Algorithm 3 will always outputs a dual solution with value $\Omega(1/\kappa)$ rather than a cut as there is no cut S with $\vec{\phi}_\pi(S) \leq O\left(\frac{1}{\kappa}\sqrt{\frac{\log n}{\epsilon}}\right) \leq \frac{1}{2}\vec{\phi}_\pi(G)$. Therefore, we can find a circulation F with demand graph D, and weights y_p over shortcut cycles, such that

$$\phi_\pi(F) \geq \lambda_2\left(\Pi^{-\frac{1}{2}}\left(L_{\mathrm{sym}}(D) - \sum_p y_p T_p\right)\Pi^{-\frac{1}{2}}\right) \gtrsim \frac{\vec{\phi}_\pi(G)}{\sqrt{\log n}},$$

where the first inequality is by (4.2). □

5 Primal-Dual Algorithms for Reweighted Eigenvalues and Cut-Matching Game

In this section, we show that the regret minimization framework can also be used to compute reweighted eigenvalues in Section 5.1 and to derive cut-matching game in Section 5.2.

5.1 Reweighted Eigenvalues

In this subsection, we use the regret minimization framework to compute the reweighted eigenvalue defined in [LTW23]. The main result is that there is a primal-dual algorithm to compute $\lambda_2^*(G)$

in $O(\log n/\lambda_2^*(G))$ iterations, with each iteration taking almost linear time. This combined with Theorem 1.5 will prove Theorem 1.6.

The reweighted eigenvalue was used in [LTW23] to approximate the directed edge conductance $\vec{\phi}(G)$, which is a special case of the π-weighted directed edge expansion $\vec{\phi}_\pi(G)$ when $\pi(i) = w(\delta^+(i)) + w(\delta^-(i))$, the total degree of vertex i. The result in this subsection only applies to this special case. To avoid confusion, we use the notation $d_w(i) := w(\delta^+(i)) + w(\delta^-(i))$ to denote the total degree of vertex i instead of using $\pi(i)$, and $D_w := \mathrm{diag}(d_w)$ to denote the diagonal total-degree matrix instead of using Π.

From Definition 2.7, the reweighted eigenvalue is formulated as

$$\lambda_2^*(G) := \max_{F \in \mathcal{F}(G)} \lambda_2\left(D_w^{-\frac{1}{2}}\left(L_{\mathrm{sym}}(F)\right)D_w^{-\frac{1}{2}}\right). \tag{5.1}$$

To construct a circulation F that maximizes the objective value, we can use the regret minimization framework as in Section 2.3.2 and Section 4.2. This framework reduces the above maximization problem to the simpler task of finding a circulation $F_t \in \mathcal{F}(G)$ that maximizes $\langle F_t, X_t\rangle$ where X_t is the density matrix in the matrix multiplicative update method in the t-th iteration, which can be found using a min-cost flow computation. Then, the regret bound in Theorem 2.5 can be used to prove that the average circulation $\frac{1}{T}\sum_{t=1}^T F_t$ will be an approximate maximizer to $\lambda_2^*(G)$.

Alternatively, using the min-max formulation from [LTW23][Proposition 3.4] where

$$\begin{aligned} \lambda_2^*(G) = \min_{v_1,\ldots,v_n \in \mathbb{R}^n} \max_{F \in \mathcal{F}(G)} \quad & \sum_{i<j} \frac{1}{2}\big(F(i,j) + F(j,i)\big) \cdot \|v_i - v_j\|^2 \\ \text{subject to} \quad & \sum_{i=1}^n d_w(i) \cdot v_i = \vec{0} \\ & \sum_{i=1}^n d_w(i) \cdot \|v_i\|^2 = 1, \end{aligned} \tag{5.2}$$

we can also interpret the following algorithm as a natural way to play a minimax game between a primal "embedding" player and a dual "circulation" player.

We bound the number of iterations to obtain a good approximate solution.

Theorem 5.1 (Regret Minimization for Reweighted Eigenvalue)**.** *Let $0 < \eta < 1/2$. The solution $\overline{F}$ returned by Algorithm 6 satisfies*

$$\lambda_2\left(D_w^{-\frac{1}{2}} L_{\mathrm{sym}}(\overline{F}) D_w^{-\frac{1}{2}}\right) \geq (1-2\eta)\cdot\lambda_2^*(G) \quad \text{after} \quad T = \frac{\log n}{\eta^2 \lambda_2^*(G)} \quad \text{iterations.}$$

Moreover, each iteration can be implemented using one min-cost flow computation.

Proof. The main step is to lower bound the inner product $\langle M_t, X_t\rangle$ in each iteration. The observation is that $v_1, \ldots, v_n$ form a feasible solution to the $\lambda_2^*(G)$ program as stated in (5.2). To see this, we just need to check that $v_1, \ldots, v_n$ satisfies the constraints in (5.2). Since $Y_t D_w \mathbb{1} = \vec{0}$ and

Algorithm 6 Regret Minimization Algorithm for Reweighted Eigenvalue

Input: A directed graph $G = (V, E, w)$ and step size $\eta \in (0, 1)$.

Initialization: $X_0 = \frac{1}{n-1}(I - D_w^{\frac{1}{2}} \mathbb{1}\mathbb{1}^\top D_w^{\frac{1}{2}})$.

For $t = 0$ to $T - 1$:

1. Given $X_t \succcurlyeq 0$ such that $\mathrm{tr}(X_t) = 1$ and $X_t \perp D_w^{\frac{1}{2}}\mathbb{1}$, let $Y_t := D_w^{-\frac{1}{2}} X_t D_w^{-\frac{1}{2}}$ and $v_1, \ldots, v_n$ be the Gram decomposition of Y_t.
2. (Dual Player) Compute circulation $F_t := \arg\max_{F \in \mathcal{F}(G)} \sum_{i<j} \frac{1}{2}(F(i,j) + F(j,i)) \cdot \|v_i - v_j\|^2$ and set the feedback matrix $M_t := D_w^{-\frac{1}{2}} L_{\mathrm{sym}}(F_t) D_w^{-\frac{1}{2}}$.
3. (Primal Player) Update $X'_{t+1} := \exp\Big(-\frac{\eta}{\rho}\sum_{i=0}^{t} M_i\Big)$. Let X_{t+1} be obtained from X'_{t+1} by projecting it onto the space orthogonal to $D_w^{\frac{1}{2}}\mathbb{1}$ and scaling it to have trace 1.

Output $\overline{F} = \frac{1}{T}\sum_{t=0}^{T-1} F_t$.

$Y_t = \sum_i v_i v_i^T$, we have $\sum_i d_w(i) \cdot v_i = \vec{0}$. Also, we have $\mathrm{tr}(X_t) = \sum_i d_w(i) \cdot \|v_i\|^2 = 1$. Therefore, by the definition of F_t in step (2) of Algorithm 6 and $\lambda_2^*(G)$ in (5.2),

$$\langle M_t, X_t\rangle = \langle L_{\mathrm{sym}}(F_t), Y_t\rangle = \max_{F\in\mathcal{F}(G)} \sum_{i<j} \frac{1}{2}\big(F(i,j) + F(j,i)\big) \cdot \|v_i - v_j\|^2 \geq \lambda_2^*(G). \tag{5.3}$$

Note that the width[5] is $\|M_t\| \leq 1$ because each vertex i has degree at most $d_w(i)$ in $F_t \in \mathcal{F}(G)$. As each $M_t \succcurlyeq 0$, by applying the regret bound (2.7) in Theorem 2.5 restricting to the subspace orthogonal to $D_w^{\frac{1}{2}}\mathbb{1}$, it follows that

$$\lambda_2\Big(D_w^{-\frac{1}{2}} L_{\mathrm{sym}}(\overline{F}) D_w^{-\frac{1}{2}}\Big) \geq \frac{1}{T}\sum_{t=0}^{T-1}\langle M_t, X_t\rangle \cdot (1-\eta) - \frac{\log n}{\eta T} \geq (1-\eta)\cdot\lambda_2^*(G) - \frac{\log n}{\eta T} \geq (1-2\eta)\cdot\lambda_2^*(G), \tag{5.4}$$

where the last inequality is by our choice of T. Finally, note that the maximization problem in step (2) of Algorithm 6 can be solved using one min-cost flow computation, which can be implemented in $m^{1+o(1)}$ time by [CKLPPS22] (see Section B.3). □

5.1.1 Fast Algorithm for Cheeger-Type Guarantee

Note that Algorithm 6 is fast when $\lambda_2^*(G)$ is large. On the other hand, when $\lambda_2^*(G)$ is small, then $\vec{\phi}(G)$ is also small by the directed Cheeger inequality in (2.9), and thus the $O(\sqrt{\log n})$-approximation in Theorem 1.5 is better than the directed Cheeger guarantee. So, we can combine Theorem 5.1 and Theorem 1.5 to prove Theorem 1.6.

[5] This is the reason that this theorem does not hold for general π.

Proof of Theorem 1.6. First, we apply Algorithm 3 and Lemma 4.2 with $1/\kappa := 1/\log^{1.5} n$ to either certify $\vec{\phi}(G) \gtrsim 1/\log^{1.5} n$ or to find a set S with $\vec{\phi}(S) \lesssim 1/\log n$. In the latter case, we know that the set S_1 with $\vec{\phi}(S_1) \lesssim \sqrt{\log n} \cdot \vec{\phi}(G)$ returned by Theorem 1.5 has smaller directed edge conductance than the guarantee by the directed Cheeger inequality in (2.9), and so we are done.

In the former case, we compute a set S_2 of directed edge conductance $\vec{\phi}(S_2) \lesssim \sqrt{\phi(G) \log 1/\vec{\phi}(G)}$ and return S_2. Since $\vec{\phi}(G) \geq 1/\log^{1.5} n$ in this case, the directed Cheeger inequality in (2.9) implies that $\lambda_2^*(G) \geq \vec{\phi}^2(G)/\log(1/\vec{\phi}(G)) \geq 1/\log^4 n$. By setting $\eta = 1/4$, we can get a 1/2-approximation of $\lambda_2^*(G)$ in $O(\log^5 n)$ iterations using Algorithm 6.

We show how to compute S_2 from the computations done in Algorithm 6 and the flow-based rounding algorithm for the directed Cheeger inequality in Section 3.3. Let $\lambda := \min_{1 \leq t \leq T} \langle L_{\mathrm{sym}}(F_t), Y_t \rangle$ and $u_1, \ldots, u_n$ be the Gram decomposition of a Y_t that achieves this minimum. Since $u_1, \ldots, u_n$ is a solution to (5.2) with objective value λ, using the Gaussian projection and the metric rounding step in Section 3.3, we can obtain a set S_2 with $\vec{\phi}(S_2) \lesssim \sqrt{\lambda \log(1/\vec{\phi}(G))}$.

It remains to argue that $\lambda \lesssim \vec{\phi}(G)$ to prove the approximation guarantee. By (5.4) in the proof of Theorem 5.1, we have

$$\lambda_2\left(D_w^{-\frac{1}{2}} L_{\mathrm{sym}}(\overline{F}) D_w^{-\frac{1}{2}}\right) \geq \lambda(1-\eta) - \lambda_2^*(G) \cdot \eta \geq \lambda(1-2\eta) = \frac{1}{2}\lambda,$$

where the second inequality is because $\lambda \geq \lambda_2^*(G)$ by (5.3) and the last equality is because $\eta = 1/4$. This implies that

$$\lambda \leq 2\lambda_2\left(D_w^{-\frac{1}{2}} L_{\mathrm{sym}}(\overline{F}) D_w^{-\frac{1}{2}}\right) \leq 2\lambda_2^*(G) \leq 4\vec{\phi}(G),$$

where the second inequality is due to $\overline{F} \in \mathcal{F}(G)$ and (5.1), and the last inequality is by the easy direction in (2.9). This proves that $\vec{\phi}(S_2) \lesssim \sqrt{\phi(G) \log 1/\vec{\phi}(G)}$.

Finally, we bound the time complexity of the algorithm. Computing S_1 takes $\widetilde{O}(m^{1+\epsilon})$ for an arbitrarily small constant ϵ using the fast max-flow algorithm in [CKLPPS22]. In the case that we also need to compute S_2, it takes $O(m^{1+o(1)})$ time to compute a 1/2-approximation of $\lambda_2^*(G)$, where the bottleneck is the min-cost flow computations in step (2) of Algorithm 6. Note that once again, the matrix exponential can be computed in $\widetilde{O}(m)$ time each iteration by Lemma 4.18. Finally, the metric rounding step also takes $O(m^{1+o(1)})$ time, as it also requires $O(\log n)$ max-flow computations in Lemma 3.1. □

5.2 Cut-Matching Game

Louis [Lou10] considered the following cut-matching game for directed graphs. In each round, the cut player chooses a bisection $(S, \overline{S})$ of the vertices, and the matching player chooses a directed perfect matching between $(S, \overline{S})$, which is defined as an Eulerian graph where each vertex has indegree and outdegree exactly one. Louis proved that there is a cut-player strategy such that the union of the directed perfect matchings has edge expansion $\Omega(1)$ in $O(\log^2 n)$ iterations. Note that the edge expansion in [Lou10] is the special case of Definition 1.1 when $\pi(i) = 1$ for all $i \in V$.

In this subsection, we use the matrix multiplicative weight update method in Algorithm 3 to derive an improved cut-player strategy and prove Theorem 1.7. We also extend the cut-matching game to the more general setting of π-weighted directed edge expansion in Definition 1.1, for which the bipartition returned by the cut player may not be balanced.

Algorithm 7 Cut Player Strategy

1. Let $D_1, \ldots, D_{t-1}$ be the directed perfect matchings played so far. Let $M_i = \Pi^{-\frac{1}{2}} L_{\mathrm{sym}}(D_i) \Pi^{-\frac{1}{2}}$. Compute X_t from $M_1, \ldots, M_{i-1}$ using step (3) of Algorithm 3.
2. Let $v_1, \ldots, v_n$ be the Gram decomposition of $Y_t := \Pi^{-\frac{1}{2}} X_t \Pi^{-\frac{1}{2}}$ as in step (1) of Algorithm 3.
3. If there is a vertex i with $\pi(i) \geq \frac{1}{4}\pi(V)$, then output the bipartition $L = \{i\}$ and $R = V \backslash \{i\}$.
4. Otherwise, let $u \sim \mathcal{N}(0, I)$ be a random vector. Let $y = \mathrm{median}\big(\{\langle u, v_i \rangle : i \in V\}\big)$ where the median is with respect to π. Output the bipartition $L = \{i : \langle u, v_i \rangle \leq y\}$ and $R = \overline{L}$.

For general π-weighted directed edge expansion, the requirement of the matching player is to output a directed fractional perfect matching defined as follows.

Algorithm 8 Matching Player Requirement

Given a bipartition L, R from the cut player, the matching player must play a directed fractional perfect matching D, which is defined as a weighted Eulerian subgraph where each $i \in L$ has indegree and outdegree exactly $\pi(R)\pi(i)/\pi(L)$ and each $j \in R$ has indegree and outdegree exactly $\pi(j)$.

Note that when π is uniform, then the cut player will always return a bisection, and the matching player will always return a directed (fractional) perfect matching where each vertex has indegree and outdegree one, and so this is a proper generalization of Louis' cut-matching game.

The plan is to analyze the cut-player strategy using the regret bound in Theorem 2.5 as follows.

$$\vec{\phi}_\pi(\overline{D}) \geq \lambda_2\Big(\Pi^{-\frac{1}{2}} L_{\mathrm{sym}}(\overline{D}) \Pi^{-\frac{1}{2}}\Big) \geq \frac{1}{T}\sum_{t=0}^{T} \langle L_{\mathrm{sym}}(D_t), Y_t \rangle \cdot (1-\eta) - \frac{\rho \log n}{\eta T} \gtrsim \frac{1}{\log n},$$

where $\overline{D} := \frac{1}{T}\sum_{t=1}^{T} D_t$. The first inequality is by the easy direction of $\lambda_\pi^{\triangle}$ in Proposition 3.4, the second inequality is by the regret bound in Theorem 2.5, and the third inequality is what we would like to achieve in the following.

The key quantity that we would like to lower bound is $W_t := \langle L_{\mathrm{sym}}(D_t), Y_t \rangle$, which is a random variable with respect to the filtration $\mathcal{F}_t$ that is what happened up to round t of the algorithm. At each round t, we would like to lower bound $\mathbb{E}_t[W_t]$ where $\mathbb{E}_t[\cdot] = \mathbb{E}[\cdot|\mathcal{F}_{t-1}]$. To lower bound $\mathbb{E}_t[W_t]$, we will use the following basic property of Gaussians (see e.g. [Van16, Example 3.4]):

Fact 5.2 (Gaussian Concentration)**.** *Let X be a Gaussian random variable with mean μ and variance σ^2. Then,*

$$\Pr[|X - \mu| > t\sigma] \leq 2\exp(-t^2/2).$$

Claim 5.3 (Expectation)**.** $\mathbb{E}_t[W_t] \gtrsim \frac{1}{\log n}$ *for any* t *in step (4) of Algorithm 7.*

Proof. The proof is based on the fact that, with high probability, a random Gaussian projection $v_i \mapsto \langle u, v_i\rangle$ will preserve the squared distances between all the vectors within a factor of $\log n$. Let $x \in \mathbb{R}^n$ be a random vector defined by $x(i) = \langle v_i, u\rangle$ where $u \sim \mathcal{N}(0, I)$. Then, $\mathbb{E}_t[|x(i) - x(j)|^2] = \mathbb{E}_t|\langle v_i - v_j, u\rangle|^2 = \|v_i - v_j\|^2$, and by Fact 5.2,

$$\Pr\left[\left|x(i) - x(j)\right|^2 \gtrsim \log n \cdot \|v_i - v_j\|^2\right] \le n^{-3}.$$

Let $\mathcal{E}$ be the event that $|x(i) - x(j)|^2 \lesssim \log n \cdot \|v_i - v_j\|^2$ for all pairs $i, j \in V$. By union bound, $\Pr[\mathcal{E}] \ge 1 - n^{-1}$. Therefore,

$$\begin{aligned}
\mathbb{E}_t[W_t] &\ge \mathbb{E}_t\big[\langle L_{\mathrm{sym}}(D_t), Y_t\rangle \;\big|\; \mathcal{E}\big] \cdot \Pr[\mathcal{E}] \\
&\ge \mathbb{E}_t\left[\sum_{i\in L, j\in R} \frac{1}{2}\big(D(i,j) + D(j,i)\big) \cdot \|v_i - v_j\|^2 \;\middle|\; \mathcal{E}\right] \cdot (1 - n^{-1}) \\
&\gtrsim \frac{1}{\log n} \cdot \mathbb{E}_t\left[\sum_{i\in L, j\in R} \big(D(i,j) + D(j,i)\big) \cdot |x(j) - x(j)|^2\right].
\end{aligned}$$

Let y be the median in step (4) such that $x(i) \le y \le x(j)$ for all $i \in L$ and $j \in R$. Let $r = \pi(R)/\pi(L)$. Since $\max_i \pi(i) \le \frac{1}{4}\pi(V)$ in step (4), it follows that $\frac{1}{2}\pi(V) \le \pi(L) \le \frac{3}{4}\pi(V)$ and thus $\frac{1}{3} \le r \le 1$. Continuing,

$$\begin{aligned}
&\mathbb{E}_t\left[\sum_{i\in L, j\in R} \big(D(i,j) + D(j,i)\big) \cdot |x(j) - x(j)|^2\right] \\
&\ge \mathbb{E}_t\left[\sum_{i\in L, j\in R} \big(D(i,j) + D(j,i)\big) \cdot \Big(\big(x(i) - y\big)^2 + \big(x(j) - y\big)^2\Big)\right] \\
&= 2 \cdot \mathbb{E}_t\left[\sum_{i\in L} r \cdot \pi(i) \cdot \big(x(i) - y\big)^2 + \sum_{j\in R} \pi(j) \cdot \big(x(j) - y\big)^2\right] \\
&\ge \frac{2}{3} \cdot \mathbb{E}_t\left[\sum_i \pi(i) \cdot \big(x(i)^2 - 2y \cdot x(i)\big)\right] \\
&= \frac{2}{3}\sum_i \pi(i) \cdot \|v_i\|^2 = \frac{2}{3},
\end{aligned}$$

where the first equality is because $\sum_{j\in R} D(i,j) = r \cdot \pi(i)$ for $i \in L$ and $\sum_{i\in L} D(i,j) = \pi(j)$ by the matching player requirement, the last inequality is because $r \ge \frac{1}{3}$ in step (4) of Algorithm 7, the second last equality is because $\sum_i \pi(i) \cdot x(i) = \langle u, \sum_i \pi(i) \cdot v_i\rangle = 0$ and $\mathbb{E}_t[x(i)^2] = \mathbb{E}_t[\langle u, v_i\rangle^2] = \mathbb{E}_t[v_i^T(uu^T)v_i] = \|v_i\|^2$ as $u \sim \mathcal{N}(0, I)$, and the last equality is because $1 = \mathrm{tr}(X_t) = \mathrm{tr}(\Pi^{\frac{1}{2}} Y_t \Pi^{\frac{1}{2}}) = \sum_i \pi(i) \cdot \|v_i\|^2$. □

To show that with good probability, $\sum_t W_t$ does not deviate much from its conditional expectation, we will apply Azuma's inequality, which we state as follows:

Theorem 5.4 (Azuma's Inequality [Sin22, Theorem 18.3])**.** *Let $X_0, \dots, X_T$ be a Martingale such that $|X_t - X_{t-1}| \le c_t \ \forall t \in [T]$. Then we have*

$$\Pr[|X_T - X_0| \ge \delta] \le \exp\left(-\frac{\delta^2}{2\sum_{t=1}^T c_t^2}\right).$$

Claim 5.5 (Concentration)**.** *In step (4) of Algorithm 7, for any constant $\delta > 0$,*

$$\Pr\left[\sum_{t=0}^T W_t \ge \sum_{t=0}^T \mathbb{E}_t[W_t] - \frac{\delta \cdot T}{\log n}\right] \ge 1 - \exp\left(-\Omega\left(\frac{\delta^2 \cdot T}{\log^2 n}\right)\right).$$

Proof. Let $Z_t = \sum_{i=1}^t (W_i - \mathbb{E}_i[W_i])$. Then Z_t is a martingale with respect to the filtration $\mathcal{F}_t$ with $Z_0 = 0$. Moreover,

$$|Z_t - Z_{t-1}| = \left|W_t - \mathbb{E}_t[W_t]\right| \le 2|\langle L_{\text{sym}}(D_t), Y_t\rangle| \le 2\langle \Pi, Y_t\rangle = 2,$$

where the last inequality is because $L_{\text{sym}}(D_t)$ is a Laplacian where the degree of vertex i is at most $\pi(i)$ in step (4) of Algorithm 7 and thus $L_{\text{sym}}(D_t) \preccurlyeq \Pi$, and the last equality is because $1 = \text{tr}(X_t) = \langle \Pi, Y_t\rangle$. Using Theorem 5.4, we can bound our Martingale Z_t as follows:

$$\Pr\left[|Z_T| > \delta \cdot \frac{T}{\log n}\right] \le \exp\left(-\Omega\left(\frac{\delta^2 T}{\log^2 n}\right)\right).$$

Note that $|Z_T| \le \delta T/\log n$ implies that $\sum_t W_t \ge \sum_t \mathbb{E}_t[W_t] - \frac{\delta \cdot T}{\log n}$. □

We are ready to prove Theorem 1.7 with these claims.

5.2.1 Proof of Theorem 1.7

Apply the regret bound in Theorem 2.5 with feedback matrices $M_i = \Pi^{-\frac{1}{2}} L_{\text{sym}} \Pi^{-\frac{1}{2}}$ and let $\overline{D} := \frac{1}{T}\sum_{t=1}^T D_t$, it follows that

$$\vec{\phi}_\pi(\overline{D}) \ge \lambda_2\left(\Pi^{-\frac{1}{2}} L_{\text{sym}}(\overline{D}) \Pi^{-\frac{1}{2}}\right) \ge \frac{1}{T}\sum_{t=0}^T \langle L_{\text{sym}}(D_t), Y_t\rangle \cdot (1-\eta) - \frac{\rho \log n}{\eta T}, \tag{5.5}$$

where the first inequality is by the easy direction of the SDP rounding in Proposition 3.4. The main step is to lower bound $\sum_t W_t = \sum_t \langle L_{\text{sym}}(D_t), Y_t\rangle$.

First, we consider the special case in step (3) of Algorithm 7, when there is a vertex i with $\pi(i) \ge \frac{1}{4}\pi(V)$. If this holds then we are in the large core case of Proposition 4.3. We can apply the same argument as (4.3) in Lemma 4.4 to show that $\langle L_{sym}(D_t), Y_t\rangle \ge \Omega(1)$ deterministically. Also, as $\pi(R)/\pi(L) \le 4$, each vertex i has degree at most $4\pi(i)$, and thus $M_t = \Pi^{-\frac{1}{2}} L_{\text{sym}}(D_i) \Pi^{-\frac{1}{2}} \preccurlyeq 4I$.

Otherwise, by Claim 5.3 and Claim 5.5,

$$\sum_{t=1}^{T}\langle L_{\mathrm{sym}}(D_t), Y_t\rangle \geq \Omega\left(\frac{T}{\log n}\right) - \frac{\delta\cdot T}{\log n}, \text{ with probability at least } 1-\exp\left(-\Omega\left(\frac{\delta^2 T}{\log^2 n}\right)\right).$$

Therefore, by setting $T = O\left(\frac{\log^2 n}{\eta^2\delta^2}\right)$ where η and δ are small enough constants, we have that $\sum_{t=1}^{T}\langle L_{\mathrm{sym}}(D_t), Y_t\rangle \gtrsim T/\log(n)$ with constant probability. Also, in this case, $\|M_t\| \leq 1$ as each vertex i has degree at most $\pi(i)$.

Therefore, plugging in $\eta = \frac{1}{4}$ and width bound $\rho = 4$ to (5.5), we conclude that $\vec{\phi}_\pi(\overline{D}) \gtrsim 1/\log n$, which implies Theorem 1.7 by multiplying T on both sides.

5.2.2 Approximating Directed Edge Expansion

As in [KRV06, OSVV08, Lou10], a corollary of the cut-matching game is an approximation algorithm for approximating directed edge expansion.

Algorithm 9 Cut-Matching-Game Approximation Algorithm (G, κ)

Initiate a cut-matching game where the cut player follows Algorithm 7. Each iteration, do the following:

1. Given the cut L, R returned by the cut player and a congestion value κ, compute a bidirectional max flow on $(L, R, \kappa, \beta = 1)$ using Algorithm 2.
2. If we obtain a cut S, output S and terminate. If we obtain saturating flows in both directions, $\vec{f}$ and $\overleftarrow{f}$, construct the demand graph of the circulation $f = \frac{1}{2}(\vec{f} + \overleftarrow{f})$ as follows:
   (a) Let $\vec{D}$ be the demand graph for $\vec{f}$. That is, for each $i \in L$ and $j \in R$, if there is a flow path $p \in \vec{f}$, then we add an edge (i, j) with weight $\vec{f}_p$.
   (b) Construct $\overleftarrow{D}$ from $\overleftarrow{f}$ the same way.
   (c) The matching player plays $D_t = \frac{1}{2}(\vec{D} + \overleftarrow{D})$.

Note that this algorithm is essentially a special case of Algorithm 3, where we implement the Oracle in a similar manner as in the project max flow algorithm (Algorithm 4). That is, we project our embedding vectors in a random direction and call bi-directional maxflow with $\beta = 1$. Then we either output a cut and terminate or update the embedding with the symmetric Laplacian of a circulation.

Corollary 5.6. *Given an edge capacitated directed graph $G = (V, E, w)$ and a parameter $\kappa > 0$, there is an algorithm using the cut-matching game in Theorem 1.7 such that, in $O(\log^2 n)$ iterations, either builds a directed Eulerian subgraph to certify that $\vec{\phi}_\pi(G) \gtrsim \frac{1}{\kappa}\cdot\frac{1}{\log n}$ or outputs a cut S with $\vec{\phi}_\pi(S) \lesssim \frac{1}{\kappa}$. Furthermore, each iteration can be computed using $O(1)$ single-commodity flows.*

Proof. If at any point, the algorithm outputs a cut during step 2, then by Lemma 3.3, we find a cut of directed edge expansion at most $O(1/\kappa)$ as $\beta = 1$. On the other hand, if for some $T = \Theta(\log^2 n)$ iterations, we always find a saturating flow in step 2, then the cut-matching game would have proceeded for T iterations. Then, by Theorem 1.7, the average demand graph $\frac{1}{T}\sum_{t=1}^{T} D_t$ (which is the average matching played by the matching player) has second eigenvalue at least $\Omega(1/\log n)$ with constant probability. Thus, we have constructed a circulation with congestion at most κ whose demand graph D has π-weighted edge expansion at least $\Omega(1/\log n)$, and this implies that $\vec{\phi}_\pi(G) \gtrsim \frac{1}{\kappa} \cdot \frac{1}{\log n}$. □

6 Other Generalizations

As mentioned in the introduction, the reweighted eigenvalue framework captures also vertex expansion and hypergraph edge expansion. In each case, the framework produces an SDP for which a rounding algorithm with Cheeger-like guarantee exists; see [LTW23]. In this section, we show that, analogous to the case of directed edge expansion, by adding ℓ_2^2 triangle inequality constraints to the SDPs for vertex expansion and hypergraph edge expansion, we obtain tighter relaxations which have an integrality gap of $O(\sqrt{\log n})$ to the respective expansion quantities. Moreover, there is an almost linear-time rounding algorithm for each of the SDPs.

6.1 Directed Vertex Expansion

A vertex-capacitated directed graph $G = (V, E, \pi)$ is a graph equipped with a vertex weight/capacity function $\pi : V \to \mathbb{R}_+$. Given such a graph, let $S \subset V$ be a nonempty subset of vertices. The set of out-neighbors of S is defined as $\partial^+(S) := \{v \notin S \mid \exists u \in S \text{ with } uv \in E\}$, and the directed vertex expansion $\vec{\psi}_\pi(S)$ and $\vec{\psi}_\pi(G)$ are defined as

$$\vec{\psi}_\pi(S) := \frac{\min\left\{\pi\big(\partial^+(S)\big), \pi\big(\partial^+(\overline{S})\big)\right\}}{\min\left\{\pi(S), \pi(\overline{S})\right\}} \quad \text{and} \quad \vec{\psi}_\pi(G) := \min_{\emptyset \neq S \subset V} \vec{\psi}_\pi(S).$$

Note that these definitions capture undirected vertex expansion as a special case. Also, let

$$\mathcal{F}_v(G) := \left\{ F : E \to \mathbb{R}_{\geq 0} \,\middle|\, \sum_{j:ij \in E} F(i,j) = \sum_{j:ji \in E} F(j,i) \leq \pi(i) \;\; \forall i \in V \right\}$$

denote the set of feasible vertex-capacitated circulations on G.

By adding ℓ_2^2 triangle inequality constraints to the embedding in [LTW23, Proposition 3.3], we arrive at the following program:

Definition 6.1 (Vertex Reweighted Eigenvalue with Triangle Inequalities)**.** *Given a vertex-capacitated*

directed graph $G = (V, E, \pi)$. The $\lambda_\pi^{\Delta_v}(G)$ program for directed vertex expansion is

$$\begin{aligned}\lambda_\pi^{\Delta_v}(G) := \min_{v_1,\ldots,v_n \in \mathbb{R}^n} \max_{F \in \mathcal{F}_v(G)} \quad & \frac{1}{2}\sum_{ij \in E} F(i,j) \cdot \|v_i - v_j\|^2 \\ \text{subject to} \quad & \sum_{i \in V} \pi(i) \cdot v_i = \vec{0} \\ & \sum_{i \in V} \pi(i) \cdot \|v_i\|^2 = 1 \\ & \|v_i - v_j\|^2 + \|v_j - v_k\|^2 \geq \|v_i - v_k\|^2 \quad \forall i,j,k \in V.\end{aligned}$$

Note that this is almost identical to λ_π^Δ for edge expansion in Definition 1.2. The only difference here is that F is constrained by vertex capacity constraints instead of edge capacity constraints. An analogous proof to Proposition 3.4 shows that $\lambda_\pi^{\Delta_v}$ is indeed a relaxation of $\vec{\psi}_\pi$, and by combining Lemma 3.5 and a vertex version of Lemma 3.1 (see Lemma 6.9 below), we can bound the integrality gap of this SDP relaxation as follows:

Theorem 6.2 (Vertex Integrality Gap)**.** *Let G be a directed graph with vertex weights $\pi : V \to \mathbb{R}^+$. Then we have*

$$\lambda_\pi^{\Delta_v}(G) \lesssim \vec{\psi}_\pi(G) \lesssim \sqrt{\log n} \cdot \lambda_\pi^{\Delta_v}(G).$$

In Section 2.2.3, we gave fast algorithms for approximating directed edge expansion using MMWU and explander flows. These techniques can be easily adapted to approximating vertex expansion by changing edge capacited flows to vertex capacitated flows. Moreover, it can be shown that the dual of the $\lambda_\pi^{\Delta_v}(G)$ SDP can be interpreted as finding the best vertex-capacitated expander flow to certify that G has large directed vertex expansion. We can thus obtain the following vertex analogous of Theorem 1.5, Theorem 1.7, and Proposition 1.4.

Theorem 6.3 (Fast $O(\sqrt{\log n})$ Approximation to $\vec{\psi}_\pi(G)$)**.** *For small enough $\epsilon > 0$, there is a randomized algorithm that, given any vertex-capacitated directed graph $G = (V, E, \pi)$, uses $\widetilde{O}(n^{3\epsilon})$ directed max-flow computations to compute a cut $S \subseteq V$, such that $\vec{\psi}_\pi(S) \lesssim \sqrt{\frac{\log n}{\epsilon}} \cdot \lambda_\pi^{\Delta_v}(G)$ with constant probability.*

Theorem 6.4 (Cut Matching Game for Directed Vertex Expansion)**.** *In the cut-matching game for directed graphs, there is a cut player strategy so that, in $O(\log^2 n)$ iterations, the union of the matchings played by the matching player is an Eulerian graph with vertex expansion $\Omega(\log n)$.*

Proposition 6.5 (Vertex Dual Certificate)**.** *Given a graph G with vertex weights $\pi : V \to \mathbb{R}_+$, there exists a feasible circulation $F \in \mathcal{F}_v(G)$ such that:*

$$\phi_\pi(F) \gtrsim \frac{\vec{\psi}_\pi(G)}{\sqrt{\log n}}.$$

Remark 6.6 (undirected case)**.** *We remark that our definition of directed vertex expansion also captures undirected vertex expansion, and that all the results presented above apply to the undirected*

case. Note that the vector program presented in [FHL08, Section 2.3] can also be rounded to give an $O(\sqrt{\log n})$ approximation of vertex expansion (vertex expansion defined here is within a constant factor of "minimum ratio vertex cut" in their paper and is different from the "vertex expansion" in their appendix). Our program has the advantages that it admits a fast primal-dual rounding algorithm and that it has an arguably simpler form.

While it is possible to prove Theorem 6.3, Theorem 6.4, and Proposition 6.5 directly by analyzing the $\lambda_\pi^{\Delta_v}$ program, a simpler way to obtain these results is by a reduction from vertex expansion to edge expansion.

Proposition 6.7 (Reduction from Directed Vertex Expansion to Directed Edge Expansion)**.** *Let $G = (V, E, \pi)$ be a vertex-capacitated directed graph. Then, there exists an edge-capacitated directed graph $G' = (V', E', w)$ over vertex weights π', such that $|V'| = O(|V|)$, $|E'| = O(|V| + |E|)$, and $\vec{\phi}_\pi(G') \sim \vec{\psi}_\pi(G)$. Moreover, such graph G' can be constructed in linear time, and given any $\emptyset \neq S' \subset V$ we can compute in linear time an $\emptyset \neq S \subset V$ such that $\vec{\psi}_\pi(S) \lesssim \vec{\phi}_\pi(S')$.*

For brevity, we describe the (somewhat standard) reduction and leave the verification to the reader. For each vertex i in G, create two copies i_{in} and i_{out} in G', where $\pi'(i_{in}) = \pi(i)$ and $\pi'(i_{out}) = \delta$ for a small positive $\delta \ll \min_i \pi(i)$, then draw an edge from i_{in} to i_{out} with edge weight $\pi(i)$. For each edge $e = ij \in E$, draw an edge from i_{out} to j_{in} with edge weight $M \gg \sum_i \pi(i)$. Observe that this reduction can be computed in $O(n + m)$ time, and after that we simply apply this reduction to obtain Theorem 6.3, Theorem 6.4, and Proposition 6.5 from their edge expansion counterparts.

6.1.1 Vertex Cheeger Rounding

In this section, we will prove the following generalization of Theorem 1.6 to the vertex case:

Theorem 6.8 (Vertex Fast Cheeger-type Rounding)**.** *Given an graph G with vertex capacities $\pi : V \to \mathbb{R}_+$, there is an almost-linear time algorithm to obtain a set S such that $\vec{\psi}_\pi(G) \lesssim \sqrt{\vec{\psi}_\pi(S) \cdot \log \frac{d_{\max}}{\vec{\psi}_\pi(G)}}$, where $d_{\max}$ is the maximum unweighted degree of G.*

While the previous theorems and proposition can be proved via reduction to $\vec{\phi}_\pi$, the same is not true of Theorem 6.8 because Theorem 1.6 only holds for the special case of directed edge conductance rather than the general $\vec{\phi}_\pi$. Instead, we need a version of Lemma 3.1 for vertex expansion.

Lemma 6.9 (Vertex Metric Rounding Lemma)**.** *Given a graph $G = (V, E)$, let $d(\cdot, \cdot)$ be a metric on V, and let $\pi : V \to \mathbb{R}^+$ be an arbitrary weight function over V. Suppose we are given disjoint vertex subsets $L, R \subseteq V$ as input to the algorithm. Let $r := \pi(R)/\pi(L)$ and $r' := \max\{1, r\}$. Then there is an algorithm using $O(\log n)$ maximum flow computations to output a set S with*

$$\vec{\psi}_\pi(S) \lesssim \frac{r' \cdot \max_{F \in \mathcal{F}_v(G)} \sum_{i,j \in V} F(i,j) \cdot d(i,j)}{\sum_{i \in R} \pi(i) \cdot d(i, L)}.$$

Lemma 6.10 (Unsaturated Case, Vertex Version)**.** *Suppose Algorithm 2 with vertex capacities outputs a cut S. Then $\vec{\psi}_\pi(S) \leq (\kappa/\beta r' - 1)^{-1}$, where $r' := \max\{1, r\}$.*

Given these two modifications, the proof of Lemma 6.9 follows by combining Lemma 6.10 and Lemma 3.2 as in the proof of Lemma 3.1. The rest of the proof of Theorem 6.8 is analogous to that for Theorem 1.6 in Section 5.1.1. First, we apply Theorem 6.3, and if we determine through this algorithm that $\vec{\psi}_\pi(G)$ is small, then the $O(\sqrt{\log n})$ approximation dominates the Cheeger bound. If on the other hand, we determine that $\vec{\psi}_\pi(G) = \Omega(\frac{1}{\log^{1.5} n})$, then we solve the reweighted eigenvalue program for directed vertex expansion as defined in [LTW23, Definition 1.2] in $O(\log^{1.5} n)$ iterations of MMWU. Then, we apply Cheeger rounding to find a set S in such that $\vec{\psi}_\pi(S) \lesssim \sqrt{\vec{\psi}_\pi(G) \log \frac{d_{\max}}{\vec{\psi}_\pi(G)}}$ as guaranteed by [LTW23]. To round the ℓ_1 program, we use flow rounding by applying Lemma 6.9 to attain an almost-linear runtime.

6.2 Directed Hypergraph Expansion

An edge-capacitated directed hypergraph $H = (V, E, w)$ consists of a set E of weighted directed hyperedges over vertex set V. For each edge $e \in E$, $e = (H_e, T_e)$ where $H_e, T_e \subseteq V$ are the head sets and tail sets in e respectively, and $w(e)$ is its weight. Given such a graph over vertex weights $\pi : V \to \mathbb{R}_+$, let $S \subset V$ be a nonempty subset of vertices. The set of out-neighbours of S is defined as $\delta^+(S) := \{e \in E : T_e \cap S \neq \emptyset \text{ and } H_e \cap S^c \neq \emptyset\}$, and the directed hypergraph expansion $\vec{\phi}_\pi(S)$ and $\vec{\phi}_\pi(H)$ are defined as

$$\vec{\phi}_\pi(S) := \frac{\min(w(\delta^+(S)), w(\delta^+(\overline{S})))}{\min(\pi(S), \pi(\overline{S}))} \quad \text{and} \quad \vec{\phi}_\pi(H) := \min_{\emptyset \neq S \subset V} \vec{\phi}_\pi(S).$$

Note that this captures expansion in undirected hypergraphs by taking $H_e = T_e$ for each $e \in E$, and also directed expansion in ordinary graphs by constraining $|H_e| = |T_e| = 1$.

We again derive our SDP by adding ℓ_2^2 triangle inequalities to the reweighted eigenvalue program for directed hypergraphs. Although the program is not readily available in [LTW23], its derivation follows the same idea of reducing to the simple case of undirected edge expansion in ordinary graphs, via edge-constrained circulations on the clique graph. Concretely:

Definition 6.11 (Directed Hypergraph Reweighted Eigenvalue with Triangle Inequalities)**.** *Given an edge-capacitated directed hypergraph $H = (V, E, w)$ over vertex weights $\pi : V \to \mathbb{R}_+$. Let*

$$\mathcal{F}(H) := \Bigg\{ F : V \times V \to \mathbb{R}_{\geq 0} \Bigg| \exists \{F_e : H_e \times T_e \to \mathbb{R}_{\geq 0}\}_{e \in E} \text{ s.t. } F(i,j) = \sum_{e : i \in H_e, j \in T_e} F_e(i,j),$$
$$\sum_{i \in H_e, j \in T_e} F_e(i,j) \leq w(e) \ \forall e \in E,$$
$$\sum_{i \in V} F(i,j) = \sum_{k \in V} F(j,k) \ \forall j \in V \Bigg\}$$

be the set of feasible circulations on H. The $\lambda_\pi^\Delta(H)$ program for directed hypergraph expansion is

$$\begin{aligned}\lambda_\pi^\Delta(H) := \min_{v_1,\dots,v_n\in\mathbb{R}^n} \max_{F\in\mathcal{F}_h(H)} \quad & \sum_{i<j}\frac{1}{2}(F(i,j)+F(j,i))\,\|v_i-v_j\|^2 \\ \text{subject to} \quad & \sum_{i\in V}\pi(i)\cdot v_i = \vec{0} \\ & \sum_{i\in V}\pi(i)\cdot\|v_i\|^2 = 1 \\ & \|v_i-v_j\|^2+\|v_j-v_k\|^2 \geq \|v_i-v_k\|^2 \quad \forall i,j,k\in V.\end{aligned}$$

The intuition for defining feasible circulations on directed hypergraphs this way is that they correspond to Eulerian reweightings of an underlying "clique graph" K_H of the directed hypergraph H, where for each edge (H_e, T_e), we add an arc ij from every $i \in H_e$ to $j \in T_e$. The definition $\lambda_\pi^\triangle(H)$ is a natural one for various reasons. First, it can be shown that $\lambda_\pi^\triangle(H)$ is a relaxation of $\vec{\phi}_\pi(H)$. Second when H is an undirected hypergraph and π is the total weighted degree, i.e. $H_e = T_e$, $\forall e \in E$, and $\pi(i) = \sum_{e\ni i} w(e)$, then $\lambda_\pi^\triangle(H)$ is exactly the reweighted eigenvalue program for undirected hypergraphs as defined in [LTW23, Section 5.1] but with ℓ_2^2 triangle inequalities. Third, (iii) just as our program for directed graphs is a relaxation of the SDP in [ACMM05], this program is a relaxation of the SDP in [CS18]. Note also that λ_π^Δ for ordinary graphs may be considered a special case of Definition 6.11.

Again, our main results for edge expansion extend to hypergraph expansion.

Theorem 6.12 (Hypergraph Integrality Gap)**.** *Let $H = (V, E, w)$ be an edge-capacitated directed hypergraph with vertex weights $\pi : V \to \mathbb{R}^+$. Then we have*

$$\lambda_\pi^\Delta(H) \lesssim \vec{\phi}_\pi(H) \lesssim \sqrt{\log n}\cdot\lambda_\pi^\Delta(H)$$

Theorem 6.13 (Fast $O(\sqrt{\log n})$ Approximation to $\vec{\phi}_\pi(H)$)**.** *For small enough $\epsilon > 0$, there is a randomized algorithm that, given any edge-capacitated directed hypergraph $H = (V, E, w)$ over vertex measure $\pi : V \to \mathbb{R}_+$, uses $\widetilde{O}(n^{3\epsilon})$ directed max-flow computations to compute a cut $S \subseteq V$, such that $\vec{\phi}_\pi(S) \lesssim \sqrt{\frac{\log n}{\epsilon}}\cdot\lambda_\pi^\Delta(H)$ with constant probability.*

Proposition 6.14 (Hypergraph Dual Certificate)**.** *Given a hypergraph $H = (V, E, w)$ with vertex weights $\pi : V \to \mathbb{R}_+$, there exists a feasible circulation $F \in \mathcal{F}_h(H)$ such that:*

$$\phi_\pi(F) \gtrsim \frac{\vec{\phi}_\pi(H)}{\sqrt{\log n}}.$$

We can also define a cut-matching game for directed hypergraphs, where the matching player is required to return an Eulerian subgraph of the clique graph K_H satisfying the indegree and outdegree constraints as in Algorithm 8.

Theorem 6.15 (Cut Matching Game for Directed Hypergraph Expansion)**.** *In the cut-matching game for directed hypergraphs, there is a cut player strategy so that, in $O(\log^2 n)$ iterations, the union of the matchings played by the matching player is a feasible circulation on H with hypergraph expansion $\Omega(\log n)$.*

The key for obtaining these results is to relate hypergraph expansion of H to the edge expansion of an ordinary derived graph G_H as in [CS18], which we present here for completeness.

Definition 6.16 (Derived Graph of Directed Hypergraphs [CS18, Fact 1.1])**.** *Let $H = (V, E, w)$ be an edge-capacitated directed hypergraph over vertex weights $\pi : V \to \mathbb{R}_+$. The derived graph $G_H = (V', E', w')$ over vertex weights $\pi' : V' \to \mathbb{R}_+$ is defined as follows:*

- $V' := V \cup \{i_e^{in} : e \in E\} \cup \{i_e^{out} : e \in E\}$
- $E' := \{(j, i_e^{in}) : j \in H_e, e \in E\} \cup \{(i_e^{in}, i_e^{out}) : e \in E\} \cup \{(i_e^{out}, k) : k \in T_e, e \in E\}$
- $w'(j, i_e^{in}) = w'(i_e^{out}, k) = \infty$ *and* $w'(i_e^{in}, i_e^{out}) = w(e)$ *for all* $e \in E$, $(j, k) \in H_e \times T_e$
- $\pi'(i) = \pi(i)$ *for all* $i \in V$, *and* $\pi'(i_e^{in}) = \pi'(i_e^{out}) = 0$ *for all* $e \in E$.

From [CS18, Fact 1], there is a correspondence between subsets $S \subseteq V$ and $S' \subseteq V'$ so that $\vec{\phi}_\pi(S) \sim \vec{\phi}_{\pi'}(S')$. Therefore, if we perform a black-box reduction from $\vec{\phi}_\pi(H)$ to $\vec{\phi}_{\pi'}(G_H)$, we obtain Theorem 6.13, Theorem 6.15, and Proposition 6.14, although the approximation guarantees using this approach degrade to $O(\sqrt{\log(n+m)})$ or $O(\log(n+m))$ (since $|V'| = \Theta(n+m)$), which are worse when $m = \omega(\text{poly}(n))$.

To obtain these results in full, one needs to derive hypergraph analogues of Lemma 3.1, and of the algorithms in Section 4 and Section 5.2. To this end, the key modification is to replace the bidirectional max-flow algorithm in Algorithm 2 by its hypergraph counterpart, and we may leave the other components essentially unchanged. We will need to define flows on hypergraphs H and obtain a hypergraph version of min-cut-max-flow theorem, and this is achieved by considering flows and cuts on the derived graph G_H:

1. Given L, R, a partition of V, we add vertices $\{s, t\}$ to G_H with s connected to L and t connected to R as in Algorithm 2 and maximum $s - t$ and $t - s$ flows.
2. Each flow path is of the form $(s, j_1, i_{e_1}^{in}, i_{e_1}^{out}, j_2, i_{e_2}^{in}, i_{e_2}^{out}, \ldots, j_\ell, t)$, where $(j_t, j_{t+1}) \in H_{e_t} \times T_{e_t}$ for all $1 \le t \le \ell - 1$. It corresponds to the flow path $(s, j_1, j_2, \ldots, j_\ell, t)$ in the respective s-t flow problem in the clique graph K_H. One can then check that bidirectional flows on G_H correspond to Eulerian reweightings on K_H, i.e. feasible circulations on H.
3. The min-cut-max-flow theorem yields an s-t cut in G_H. Since $w'(j, i_e^{in})$ and $w'(i_e^{out}, k)$ are large, the cut edges will only be in one of the following types: (s, j), (j', t), or (i_e^{in}, i_e^{out}) (where $j, j' \in V$). Thus, we derive a hypergraph version of Lemma 3.3, whose proof follows closely that of the original version. Consequently, we obtain a hypergraph version of Lemma 3.1.

Thus, the overall idea for generalizing our arguments for directed graphs to directed hypergraphs is to use the derived graph G_H to compute bi-directional flows. Then we can either find a directed sparse cut or many feasible circulations in $\mathcal{F}(H)$, whose average can be used to certify that $\vec{\phi}_\pi(H)$ is large through MMWU.

Finally, we will give the following generalization of Theorem 1.6 to *undirected* hypergraph conductance, improving on the runtime of the algorithm in [LTW23, Section 5]. Recall that for undirected hypergraphs, $H_e = T_e$ for all $e \in E$.

Theorem 6.17 (Hypergraph Fast Cheeger-type Rounding)**.** *Given an edge-capacitated undirected hypergraph $H = (V, E, w)$ with vertex weights $\pi(i) = \sum_{e:i\in H_e} w(e)$, there is an almost-linear time algorithm to obtain a set $S \subseteq V$ such that $\phi_\pi(S) \lesssim \sqrt{\phi_\pi(G) \cdot \log r}$, where $r := \max_e |H_e|$ is the maximum edge size of H.*

The proof of Theorem 6.17 is analogous to that for vertex expansion, except that this time, it suffices to use threshold rounding as in [LTW23], which can be done in linear time.

We remark that by using the directed hypergraph metric rounding lemma outlined before, as well as a version of Lemma 3.3 for directed hypergraphs, a fast Cheeger-type rounding algorithm exists for directed hypergraphs, with the guarantee that

$$\vec{\phi}_\pi(S) \lesssim \sqrt{\vec{\phi}_\pi(G) \cdot \log \frac{r}{\vec{\phi}_\pi(G)}}.$$

This would necessitate a Cheeger inequality for directed hypergraphs, which is not available in [LTW23] but follows readily from their technique.

7 Summary

In this paper, we have given a unifying approach for generalizing all the major approximation algorithms for undirected edge expansion to other settings, including directed edge expansion, directed vertex expansion and directed hypergraph expansion. These algorithms may be summarized in a one-sentence formula: use flows to implement an MMWU algorithm for solving a reweighted eigenvalue program or playing a cut-matching game. Such short formula either recovers or improves all relevant past results.

On the practical side, it is worth noting that the algorithms presented in this paper are almost-linear time. While we have theoretical guarantee on their runtimes and approximation ratios, we are curious about whether they may be implemented to find good sparse cuts in large graphs quickly. Such implementation would bring these algorithms into the practical realm; in particular, fast spectral algorithms for computing hypergraph sparse cuts would be useful in certain machine learning applications, and fast algorithms for finding reweightings could be useful in graphical neural networks for hypergraphs and directed graphs.

We believe our approach leaves much room for further research into graph partitioning problems. Since multi-way graph partitioning has found many applications in clustering and classification, one interesting open area of research is to design fast approximation algorithms for multi-way graph partitioning and generalize it to the vertex, directed graph, and hypergraph settings. In [Yos19], Yoshida recovered Cheeger-type inequalities for partitioning problems on all submodular functions, which is more general than directed hypergraphs. Another open problem could be to use flows and reweighted eigenvalues to obtain fast approximation algorithms for partitioning problems on more general classes of submodular functions.

References

[ACMM05] Amit Agarwal, Moses Charikar, Konstantin Makarychev, Yury Makarychev. $O(\sqrt{\log n})$ *approximation algorithm for min UnCut, min 2CNF deletion, and directed cut problems.* In Proceedings of the 37th Annaul ACM Symposium on Theory of Computing (STOC), 573–581, 2005.

[Alo86] Noga Alon. *Eigenvalues and expanders.* Combinatorica, 6, 83–96, 1986.

[AM85] Noga Alon, Vitali Milman. λ_1*, isoperimetric inequalities for graphs, and superconcentrators.* Journal of Combinatorial Theory, Series B, 38(1), 73–88, 1985.

[And10] Matthew Andres. *Approximation algorithms for the edge-disjoint paths problem via Raecke decompositions.* In Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS), 277-286, 2010.

[AK07] Sanjeev Arora, Satyen Kale. *A combinatorial, primal-dual approach to semidefinite programs.* In Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC), 227–236, 2007.

[ARV09] Sanjeev Arora, Satish Rao, Umesh Vazirani. *Expander flows, geometric embeddings and graph partitioning.* Journal of the ACM, 56(2):1–37, 2009.

[BGS20] Aaron Bernstein, Maximilian Probst Gutenberg, and Thatchaphol Saranurak. *Deterministic decremental reachability, SCC, and shortest paths via directed expanders and congestion balancing.* In Proceedings of IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS), 1123–1134, 2020.

[CE13] Chandra Chekuri, and Alina Ene. *Poly-logarithmic approximation for maximum node disjoint paths with constant congestion.* In proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms (SODA), 326–341, 2013.

[CLTZ18] T.-H. Hubert Chan, Anand Louis, Zhihao Gavin Tang, Chenzi Zhang. *Spectral properties of hypergraph Laplacian and approximation algorithms.* Journal of the ACM, 65(3):1–48, 2018.

[CS18] T.-H. Hubert Chan, Bintao Sun. *SDP primal-dual approximation algorithms for directed hypergraph expansion and sparsest cut with product demands.* In Proceedings of the 24th Annual International Computing and Combinatorics Conference (COCOON), 688-700, 2018.

[CKLPPS22] Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, Sushant Sachdeva. *Maximum flow and minimum-cost flow in almost linear-time.* In Proceedings of the 63rd IEEE Annual Symposium on Foundations of Computer Science (FOCS), 612–623, 2022.

[Chu12] Julia Chuzhoy. *Routing in undirected graphs with constant congestion.* Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC), 855–874, 2012.

[CK19] Julia Chuzhoy and Sanjeev Khanna. *A new algorithm for decremental single-source shortest paths with applications to vertex-capacitated flow and cut problems*. In Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC), 389–400, 2019.

[CGLNPS20] Julia Chuzhoy, Yu Gao, Jason Li, Danupon Nanongkai, Richard Peng, and Thatchaphol Saranurak. *A Deterministic Algorithm for Balanced Cut with Applications to Dynamic Connectivity, Flows, and Beyond*. In Proceedings of the 61st IEEE Annual Symposium on Foundations of Computer Science (FOCS), 1159–1167, 2020.

[CL16] Julia Chuzhoy and Shi Li. *A polylogarithmic approximation algorithm for edge-disjoint paths with congestion 2*. Journal of the ACM 63(5), 1–51, 2016.

[CS21] Julia Chuzhoy and Thatchaphol Saranurak. *Deterministic algorithms for decremental shortest paths via layered core decomposition*. In Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA), 2478–2496.

[JL84] William B Johnson and Joram Lindenstrauss *Extensions of Lipschitz mappings into a Hilbert space*. Contemporary Mathematics. 26, 189–206, 1984.

[FHL08] Uriel Feige, MohammadTaghi Hajiaghayi, James R. Lee. *Improved approximation algorithms for minimum weight vertex separators*. SIAM Journal on Computing, 38(2):629–657, 2008.

[Kal07] Satyen Kale. *Efficient algorithms using the multiplicative weights update method*. PhD thesis, Princeton University, 2007.

[KRV06] Rohit Khandekar, Satish Rao, Umesh Vazirani. *Graph partitioning using single commodity flows*. In Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC), 385–390, 2006.

[KLT22] Tsz Chiu Kwok, Lap Chi Lau, Kam Chuen Tung. *Cheeger inequalities for vertex expansion and reweighted eigenvalues*. In Proceedings of the 62nd IEEE Annual Symposium on Foundations of Computer Science (FOCS), 366–377, 2022.

[Kol23] Vladimir Kolmogorov. *A simpler and parallelizable $O(\sqrt{\log n})$)-approximation algorithm for Sparsest Cut*. In arXiv: 2307.00115

[LTW23] Lap Chi Lau, Kam Chuen Tung, Robert Wang. *Cheeger inequalities for directed graphs and hypergraphs using reweighted eigenvalues*. In Proceedings of the 55th Annual Symposium on Theory of Computing (STOC), 2023.

[LR99] Tom Leighton, Satish Rao. *Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms*. Journal of the ACM 46(6), 787–832, 1999.

[Lou10] Anand Louis. *Cut-matching games on directed graphs*. In arXiv:1010.1047, 2010.

[Lou15] Anand Louis. *Hypergraph Markov operators, eigenvalues and approximation algorthms*. In Proceedings of the 47th Annual Symposium on Theory of Computing (STOC), 713–722, 2015.

[LM14] Anand Louis, Yury Makarychev. *Approximation algorithms for hypergraph small set expansion and small set vertex expansion*. In Proceedings of APPROX-RANDOM, 339–355, 2014.

[LRS13] Yin Tat Lee, Satish Rao, Nikhil Srivastava. *A new approach to computing maximum flows using electrical flows*. In Proceedings of the 54th annual ACM symposium on Theory of Computing (STOC). 755–764, 2013.

[LRV13] Anand Louis, Prasad Raghavendra, Santosh Vempala. *The complexity of approximating vertex expansion*. In Proceedings of the 54th IEEE Annual Symposium on Foundations of Computer Secience (FOCS), 360–369, 2013.

[OZ22] Sam Olesker-Taylor, Luca Zanetti. *Geometric bounds on the fastest mixing Markov chain*. In the 13th Innovations in Theoretical Computer Science Conference (ITCS 2022).

[OSVV08] Lorenzo Orecchia, Leonard Schulman, Umesh Vazirani, Nisheeth Vishnoi. *On partitioning graphs via single commodity flows*. In Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC), 461–470, 2008.

[Rot16] Thomas Rothvoss. *Lecture notes on the ARV algorithm for sparsest cut*. In arXiv preprint arXiv:1607.00854, 2016.

[She09] Jonah Sherman. *Breaking the multicommodity flow barrier for $O(\sqrt{\log n})$-approximations to sparsest cut*. In Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS), 363–372, 2009.

[Sin22] Alistair Sinclair. *CS271 Randomness and Computation Lecture 18 Fall 2022*. https://people.eecs.berkeley.edu/ sinclair/cs271/f22.html, 2022.

[Tre16] Luca Trevisan. *Lecture notes on graph partitioning, expanders and spectral methods*. 2016.

[Van16] Ramon Van Handel. *Probability in High Dimension, APC 550 Lecture Notes Princeton University*. 2016.

[Yos19] Yuichi Yoshida. *Cheeger inequalities for submodular transformations*. In Proceedings of the 30th ACM-SIAM Symposium on Discrete Algorithms (SODA), 2582–2601, 2019.

A Missing Proofs of Section 3

Proof of Proposition 3.4. Let $\emptyset \neq S \subset V$. We construct an SDP solution to show that $\xi(G) \leq 2\vec{\phi}_\pi(S)$. Consider the vector solution

$$v_i := \begin{cases} (a, 0, \ldots, 0), & \text{if } i \in S, \\ (b, 0, \ldots, 0), & \text{otherwise .} \end{cases}$$

where $a, b \in \mathbb{R}$ satisfies $a\pi(S) + b\pi(S^c) = 0$ and $a^2\pi(S) + b^2\pi(S^c) = 1$. Note that such (a, b) must exist. A routine check reveals that all the constraints on v_i are satisfied. It remains to show that

$$\frac{1}{2} \sum_{ij \in E} F(i,j) \|v_i - v_j\|^2 \leq 2\vec{\phi}_\pi(S) \qquad \forall F \in \mathcal{F}(G).$$

Solving for a and b, we see that $(a-b)^2 = \pi(V)/\pi(S)\pi(S^c)$. Then,

$$\begin{aligned} \frac{1}{2} \sum_{ij \in E} F(i,j) \|v_i - v_j\|^2 &= \frac{1}{2} \left[\sum_{i \in S, j \in S^c} + \sum_{i \in S^c, j \in S} \right] F(i,j)(a-b)^2 \\ &= (F(S, S^c) + F(S^c, S)) \cdot \frac{\pi(V)}{2\pi(S)\pi(S^c)} \\ &\leq \frac{F(S, S^c) + F(S^c, S)}{\min(\pi(S), \pi(S^c))} \\ &\leq \frac{2\min(\delta^+(S), \delta^+(S^c))}{\min(\pi(S), \pi(S^c))} = 2\vec{\phi}_\pi(S), \end{aligned}$$

where the last inequality uses the fact that $F \in \mathcal{F}(G)$ is an Eulerian reweighting, so that $F(S, S^c) = F(S^c, S) \leq \min(w(\delta^+(S)), w(\delta^+(S^c)))$. This finishes the proof that $\lambda_\pi^\Delta(G) \leq 2\vec{\phi}_\pi(G)$. □

Proof of Lemma 3.5. The algorithm for the unweighted version in Theorem 2.2 proceeds as follows. Let $\sigma > 0$ be a suitable absolute constant. First, choose a random direction $u \sim \mathbb{S}^{n-1} \subseteq \mathbb{R}^n$, and order the vertices i by $\langle v_i, u \rangle$. Second, if the median value is M, set L to be the set of vertices i such that $\langle v_i, u \rangle \geq M + \sigma/\sqrt{n}$, and set R to be the set of vertices i such that $\langle v_i, u \rangle < M$. Third, while there are pairs $(i, j) \in L \times R$ such that $\|v_i - v_j\|^2 < \Delta = \Theta(1/\sqrt{\log n})$, remove i from L and j from R. If $|L| \geq \Omega(n)$ and $|R| \geq \Omega(n)$, then the procedure successfully finds two large subsets that are at least $\Delta \geq \Omega(1/\sqrt{\log n})$ ℓ_2^2-distance apart. Refer to [ARV09] for complete details.

To prove the π-weighted version in Lemma 3.5, we do a reduction to the unweighted case. Recall the assumption that $\pi(V) = 1$. Let $K \in \mathbb{N}$ such that $K \cdot \min_{i \in V} \pi(i) \geq 1/2$, and let $\pi'(i) := \lceil K\pi(i) \rceil$ for $i \in V$. We may further assume that $\min_{i \in V} \pi(i) \geq \Omega(1/\operatorname{poly}(n))$ (vertices with smaller measure may be ignored), so that $K \leq O(\operatorname{poly}(n))$. Create $\pi'(i)$ copies of v_i and feed the embedding to the unweighted algorithm. Note that the embedding consists of $\Theta(K)$ vectors. In the end of the unweighted algorithm, w.h.p. the output sets L and R will have size $\Theta(K)$ each, and they will be at least $\Omega(1/\sqrt{\log n})$ ℓ_2^2-distance apart.

Note that if one copy of v_i is in either of the output set, we may include all copies of v_i in that output set, without affecting the distance between L and R. Then, the π-measure of vertices in L will be at least

$$\sum_{i\in L}\pi(i)\geq\sum_{i\in L}\frac{\pi'(i)}{2K}=|L|/2K\geq\Omega(1);$$

same for R. We have proved that w.h.p. $\pi(L),\pi(R)\geq\Omega(1)$. The runtime is polynomial in the number of vectors which is $\Theta(K)$, and hence polynomial in n.

To get rid of the K-dependence in the runtime, we may modify the unweighted algorithm as follows: In the second step, compute the weighted median. In the third step, instead of removing both vertices i and j, subtract $\min(\pi(i),\pi(j))$ from both $\pi(i)$ and $\pi(j)$, and remove the vertex whose π-measure drops to zero. □

Proof of Fact 3.7. Direct calculation gives

$$\begin{aligned}\sum_{i,j\in V}\pi(i)\pi(j)\left\|v_i-v_j\right\|^2 &= \sum_{i,j\in V}\pi(i)\pi(j)\left(\|v_i\|^2+\|v_j\|^2-2\langle v_i,v_j\rangle\right)\\ &= 2\left(\sum_{i\in V}\pi(i)\right)\left(\sum_{j\in V}\pi(j)\left\|v_j\right\|^2\right)-2\left\|\sum_{i\in V}\pi(i)v_i\right\|^2\\ &= 2\cdot 1-2\cdot 0=2.\end{aligned}$$

□

Proof of Lemma 3.8. Direct calculation gives

$$\begin{aligned}2 &= \sum_{i,j\in V}\pi(i)\pi(j)d(i,j)\\ &\leq s^2\sum_{i,j\in V}\pi(i)\pi(j)\Big[d(i,L)+diam(L)+d(j,L)\Big]\\ &= s^2(\pi(V)^2 diam(L)+2\pi(V)\cdot\sum_{j\in V}\pi(j)d(j,L))\\ &= s^2(diam(L)+2\cdot\sum_{j\in R}\pi(j)d(j,L)),\end{aligned}$$

where the inequality comes from applying the s-relaxed triangle inequality twice and the last equality uses $\pi(V)=1$. Rearranging gives the desired result. □

B Missing proofs of Section 4

Proof of Proposition 4.3. We will make use of [Kal07, Lemma 5], which we will restate as follows:

Lemma B.1 ([Kal07, Lemma 5]). *Given a set of embedding vectors $v_1, \dots v_n$ such that $\sum_{i,j} \|v_i - v_j\|^2 > 4n^2/5$. Then one of the two cases hold:*

- *There is a node i such that $|B(i, \frac{1}{2\sqrt{10}})| > n/4$*
- *There is a set of nodes $S \subseteq V$ and an $i_0 \in S$ such that $\forall i \in S$, $\|v_i - v_{i_0}\|^2 = O(1)$ and $\sum_{i,j \in S} \|v_i - v_j\|^2 = \Omega(n^2)$*

Note that this immediately implies Proposition 4.3 in the case where π is uniform. For general π, suppose we re-scale π so that for each i, $\pi(i)$ is an integer. Then we have $\sum_{i,j} \pi(i)\pi(j) \|v_i - v_j\|^2 = \pi(V)^2$. We define a new set of vertices V' with $|V'| = \pi(V)$ with embedding vectors $w : V' \to \mathbb{R}^n$. In particular, we replace each $i \in V$ with $\pi(i)$ vertices in V' each embedded at the point v_i. Then we have $\sum_{i',j' \in V'} \left\|w_{i'} - w_{j'}\right\|^2 = \sum_{i,j \in V} \pi(i)\pi(j) \|v_i - v_j\|^2 = \pi(V)^2$. Now, by applying [Kal07, Lemma 5], we must have one of the following two cases

- There is a vertex $i' \in V'$ such that $|\{j' : \left\|w_{j'} - w_{i'}\right\| \leq \frac{1}{2\sqrt{10}}\}| > \pi(V)/4$ which means there is a $i \in V$ such that $\pi(B(i, \frac{1}{2\sqrt{10}})) > \pi(V)/4$
- There is a set of nodes $S' \subseteq V'$ and an $i'_0 \in S$ such that $\forall i' \in S'$, $\left\|w_{i'} - w_{i'_0}\right\|^2 = O(1)$ and $\sum_{i',j' \in S'} \left\|w_{i'} - w_{j'}\right\|^2 = \Omega(\pi(V)^2)$. This means there is a set of nodes $S \subseteq V$ and an $i_0 \in S$ such that $\forall i \in S$, $\|v_i - v_{i_0}\|^2 = O(1)$ and $\sum_{i,j \in S} \pi(i)\pi(j) \|v_i - v_j\|^2 = \Omega(\pi(V)^2)$

□

proof of Lemma 4.11. We will prove the theorem via a simple reduction to [Kal07, Lemma 14], which we will state as follows:

Lemma B.2 ([Kal07, Lemma 14]). *Suppose there are vectors $v_1, \dots, v_n$ such that $\|v_i\|^2 \leq 1$ for each i and $\sum_{i,j} \|v_i - v_j\|^2 \geq an^2$ for some constant a. Let $c \leq a/256$. Then with probability $8c$ over u, There exists sets L, R, of size at least $2cn$ such that for each $i \in L$ and $j \in R$, we have $\langle v_j - v_j, u\rangle \geq \sigma$.*

Now suppose we re-scale π so that for each i, $\pi(i)$ is an integer. Then we have $\sum_{i,j} \pi(i)\pi(j) \|v_i - v_j\|^2 \geq a\pi(V)^2$ for some constant a. We define a new set of vertices V' with $|V'| = \pi(V)$ with embedding vectors $w : V' \to \mathbb{R}^n$. In particular, we replace each $i \in V$ with $\pi(i)$ vertices in V' each embedded at the point v_i. Then we have $\sum_{i',j' \in V'} \left\|w_{i'} - w_{j'}\right\|^2 = \sum_{i,j \in V} \pi(i)\pi(j) \|v_i - v_j\|^2 \geq a\pi(V)^2$. Then, applying [Kal07, Lemma 14], we have that with probability at least $8c$ over $u \sim N(0, I)$, two sets $L', R' \subseteq V'$, each of size at least $2c\pi(V)$ such that for all $i \in L', j \in R'$, $\langle v_j - v_i, u\rangle \geq \sigma$. This implies that there exist $L, R \subseteq V$ such that $\pi(L), \pi(R) \geq 2c\pi(V)$. □

B.1 Sherman Main Theorem

First, we will formally define the distribution used in Theorem 4.9. In the statement of the theorem, the distribution is over K vectors $u_1, \ldots, u_K$ for some $K = O(\sqrt{\log n})$. The explicit distribution is over a shuffling of independent Gaussian vectors and correlated Gaussian vectors as defined in section 5.4.1 of [She09].

Definition B.3. *(correlated sequence of Gaussian vectors) Let $\mathcal{N}_\rho^k$ be a distrbution over vectors $u_1, \ldots, u_k \in \mathbb{R}^d$ such that each u_i has distribution $\mathcal{N}(0, I_d)$, and u_{i+1} is ρ-correlated with u_i. In particular, if we define the matrix $U \in \mathbb{R}^{k \times d}$ with $U_{i,j} = u_i(j)$, then each column of the matrix, $(u_1(i), ...u_k(i))$, is a k-dimensional multivariate normal distribution with covariance matrix $\Sigma_{a,b} = \rho^{|a-b|}$, and the d columns are mutually independent.*

To prove Theorem 4.9, we first note that it is invariant under scaling of π, which means we once again assume WLOG that $\pi(i)$ is an integer for each i. Once again, we apply the reduction in which we have a set of $\pi(V)$ vertices, call it V', and for each $i \in V$, we embed $\pi(i)$ vertices in V' at the point v_i. Given the π-fractional matching cover $\mathcal{M}$, we can define a matching cover $\mathcal{M}'$ over V' as follows. Given a vector u, for each $i, j \in V \times V$, we add $\mathcal{M}_u(i, j)$ edges between the corresponding vertices at v_i and v_j in V'. Clearly, $\mathcal{M}'$ is a (σ, δ) matching cover V'. The idea of Sherman's original argument was to show that if k matchings chained together does not give many paths between vertices i, j such that $\|v_i - v_j\|^2 \geq l$, then there is a vertex i such that with constant probability over random vectors $u \sim \mathcal{N}(0, I)$, we have $\langle v_j - v_i, u \rangle \geq \Omega(k\sigma)$ for some j such that $\|v_i - v_j\|^2 \leq l$. Then he shows that for large enough k (in particular $k \approx \sqrt{l \log n}$), the probability of the later event happening cannot be $\Omega(1)$ by union bounding over all i, j pairs. In our case, even though we have $\pi(V)$ points, there are still only n distinct positions so it still suffices to union bound over n points instead of $\pi(V)$ points. For completeness, we will give the details of the argument in the rest of the section.

Sherman defines a *uniform (σ, δ)-matching cover* [She09, Definition 5.4.1] as a (σ, δ)-matching cover in which each vertex has at least a δ probability of having out-degree 1 in the matching. By iteratively pruning vertices V' whose probability of being matched is less than $\delta/4$, we obtain $X \subseteq V'$ of size at least $\pi(V)/4$ such that $\mathcal{M}'$ is a $(\sigma, \delta/4)$-uniform matching cover over X.

Definition B.4. *Given a distribution $\mathcal{D}$ over vectors $u_1, \ldots, u_k$, and a matching-cover over the vertices X, a vertex i is $(\sigma, \delta, \gamma, l)-$covered, by $\mathcal{D}$ if with probability at least δ over a random Gaussian $u \sim \mathcal{N}(0, I)$,*

$$\Pr_{u_2, \ldots, u_k} [\exists j \in B(i, l) : \ (i, j) \in \mathcal{M}'(\mathcal{D}), \ \langle v_j - v_i, u \rangle \geq \sigma | u_1 = u] \geq \gamma \tag{B.1}$$

Now, we will use the following lemma from [She09]:

Lemma B.5 ([She09, Lemma 5.4.8])**.** *Let $\mathcal{M}'$ be a (σ, δ)-uniform matching cover of X where $\delta \leq 1/16$. Let $l \leq \sigma/2^7 \sqrt{\log(1/\delta)}$ and $k \geq 1$. Then one of the following must occur:*

1. *There are distributions $\mathcal{D}^0, \ldots, \mathcal{D}^k$ such that for every $b \leq k$, at least $\delta^{6b}|X|$ vertices are $(b\sigma/4, \delta^8, \delta^{56bk}, \sqrt{l})$-covered in $\mathcal{M}'(\mathcal{D}^b)$*

2. *There is a efficiently sample-able distribution $\mathcal{D}^*$ such that at least $\delta^{6k}|X|$ vertices i have at least δ^{59k^2} probability of having an out-going edge to some $j \in i \backslash B(i,l)$ in $\mathcal{M}'(\mathcal{D}^*)$. Furthermore, $\mathcal{D}^*$ is a shuffling of $\mathcal{N}_{1-1/k}^{k'}$ with $\mathcal{N}_0^{k''}$ for some $k' \le k$ and $k'' \le 6k$.*

To show that case 1 cannot hold for too many rounds, we will use the following lemma

Lemma B.6. *Let $\mathcal{M}'$ be any matching cover for X and $\gamma > 0$. There are no vertices $i \in X$ that are $(\sqrt{2l\log(n/\delta)}, \delta, \gamma, \sqrt{l})$-covered by $\mathcal{M}(\mathcal{D})$*

Proof. Let $i \in X$ be arbitrary, and let $j \in B(i,l)$. This means $\|v_j - v_i\|^2 \le l$. Then by Fact 5.2, we have

$$\Pr_u[\langle v_j - v_i, u\rangle \ge \sqrt{2l\log(n/\delta)}] \le \exp(-\log(n/\delta))$$

Since there are at most n distinct embedding positions in X, the probability over u_1 that there is any $j \in B(i,l)$ such that $\langle v_j - v_i, u_1\rangle \ge l\sqrt{2\log(n/\delta)}$ is at most $(n-1)\delta/n$. In this case, the conditional probability in (B.1) must be 0, which is less than γ, for greater than $1-\delta$ fraction of u_1. □

Proof of Theorem 4.9. For any constant l, if $k \ge C\sqrt{l\log n}$ for some constant C, then case 1 of Lemma B.5 would imply that there is some $x \in X$ that contradicts Lemma B.6. Thus, we must be in case 2 for some $k < C\sqrt{l\log n}$. This means that for some distribution $\mathcal{D}^*$, the expected number of edges in $\mathcal{M}'(\mathcal{D}^*)$ between vertices $i, j \in X$ such that $\|v_i - v_j\|^2 \ge l$ is at least $e^{-O(k^2)}|X|$. Finally, we note that w paths between vertices embedded at v_i and v_j in $\mathcal{M}'(u_1, \ldots, u_k)$ correspond to a path of weight w between i and j in $\mathcal{M}$. Moreover, the algorithm for constructing $\mathcal{P}_{u_1,\ldots,u_k}$ in $\mathcal{M}(u_1, \ldots, u_k)$ is equivalent to the natrual algorithm chaining together 0/1-matchings in $\mathcal{M}'(u_1, \ldots, u_k)$. Thus, the expected total weight of paths in $\mathcal{P}_{u_1,\ldots,u_k}$ between vertices i, j at distance at least l apart is at least $e^{O(-k^2)}\pi(V)$.

Finally, we note that while our distribution $\mathcal{D}^*$ is a shuffling of $\mathcal{N}_{1-1/k}^{k'}$ and $\mathcal{N}_0^{k''}$, we have not yet given a way to find the correct ordering. However, since $k' + k'' = O(\sqrt{l\log n})$, the total number of sequences is at most $O(\sqrt{l\log n}!) = n^{o(1)}$. Thus, if we pick a random shuffling, the probability that it will be the correct ordering is $n^{-o(1)}$. Thus, after taking into account the randomness over shuffling orders, the expected total weight of paths between vertices i, j at distance at least l apart is at least $n^{-o(1)}e^{O(-k^2)}\pi(V) = e^{-O(k^2)}\pi(V)$ since $k^2 = \Theta(l\log n)$. □

B.2 Matrix Exponential

In this section, we give details on how to implement the matrix exponential step in algorithms 6, 3 and 7 and prove Lemma 4.18. Given feedback matrices $M_1, \ldots, M_t$, such that $\|M_i\| \le \rho$ for each $i \in [t]$, we would like to approximately compute the Gram decomposition of the matrix

$$Y_t = \frac{\Pi^{-1/2}\exp(-\frac{\eta}{\rho}\sum_{i=1}^t M_i)\Pi^{-1/2} - \Pi^{1/2}\mathbb{1}\mathbb{1}^\top\Pi^{1/2}}{\mathrm{tr}(\exp(-\frac{\eta}{\rho}\sum_{i=1}^t M_i)) - 1}$$

Note in particular that if we take $A = \frac{\eta}{\rho}\sum_{i=1}^{t} M_i$, then it suffices to compute the rows of the matrix $\exp(-\frac{1}{2}A)\Pi^{-1/2}$ projected into the space orthogonal to $\Pi^{1/2}\mathbb{1}$. Since computing the matrix exponential exactly is costly, we will instead compute a low-dimensional approximation of its rows by multiplying the matrix with random vectors and applying the Johnson-Lindenstrauss lemma. Thus, the problem of computing the embedding vectors reduces to the problem of computing $\exp(S)u$ for some vector u and symmetric matrix S. For our purpose, it suffices to compute the first terms in the Taylor expansion of the matrix exponential.

Lemma B.7 ([Kal07, Lemma 23])**.** *Given a symmetric matrix S and a unit vector u, let $v = \sum_{i=0}^{k} \frac{1}{i!} S^i u$. If $k \geq \max(e^2 \|S\|, \ln \frac{1}{\tau})$, then v satisfies*

$$\|\exp(S)u - v\| \leq \|\exp(S)\|\,\tau$$

Moreover, the time it takes to compute v is $O(km)$ where m is the number of non-zero entries in S.

Given the previous result, we can give the algorithm for computing the matrix exponential based on [Kal07, Section 4.7]. However, there are two modifications we must make. First, to take into account the π vertex weights, we let $\pi_{\min} = \min_i \pi(i)$, and we will show that it suffices to take $\tau = \pi_{\min}/\operatorname{poly}(n)$. Second, we must take into account the projection into the subspace orthogonal to $\Pi^{1/2}\mathbb{1}$, which is the exactly the nullspace of A. We will call this subspace $\mathcal{U}_\pi^\perp$, and for a matrix M, we will define the matrix $M|_{\mathcal{U}_\pi^\perp}$ as the matrix whose columns are those of M projected into $\mathcal{U}_\pi^\perp$. We can then modify Lemma B.7 as follows:

Lemma B.8. *Let u be a unit vector in $\mathcal{U}_\pi^\perp$, and let v be the vector obtained from applying the Taylor approximation in Lemma B.7 for $k = \max(e^2 \|A\|, \ln \frac{1}{\tau})$ iterations to the matrix $\exp(-\frac{1}{2}A)u$. In $\widetilde{O}(\rho m)$ time, we can ensure that*

$$\left\|\exp(-\frac{1}{2}A)\Big|_{\mathcal{U}_\pi^\perp} u - v\right\| \leq \left\|\exp(-\frac{1}{2}A)\Big|_{\mathcal{U}_\pi^\perp}\right\| \tau$$

Proof. Let $P \in \mathbb{R}^{n \times n-1}$ be a semi-unitary matrix mapping $\mathcal{U}_\pi^\perp$ to $\mathbb{R}^{n-1}$. Note that this means $P^\top P = I_{n-1}$ and for any $x \in \mathcal{U}_\pi^\perp$, we have $PP^\top x = x$. Since $u \in \mathcal{U}_\pi^\perp$, we have $\exp(-\frac{1}{2}A)u = \exp(-\frac{1}{2}A)|_{\mathcal{U}_\pi^\perp} u$. Moreover, the Tayler approximation v is also in $\mathcal{U}_\pi^\perp$, which means

$$\begin{aligned}
\left\|\exp(-\frac{1}{2}A)\Big|_{\mathcal{U}_\pi^\perp} u - v\right\| &= \left\|P^\top \exp(-\frac{1}{2}A)\Big|_{\mathcal{U}_\pi^\perp} u - P^\top v\right\| \\
&= \left\|P^\top \exp(-\frac{1}{2}A)\Big|_{\mathcal{U}_\pi^\perp} PP^\top u - P^\top v\right\| \\
&= \left\|P^\top \exp(-\frac{1}{2}A) PP^\top u - P^\top v\right\| \\
&= \left\|\exp(-\frac{1}{2}P^\top AP) P^\top u - P^\top v\right\|
\end{aligned}$$

where the last equality follows from the fact that Range$(A) \in \mathcal{U}_\pi^\perp$, which means $(P^\top AP)^i = P^\top A^i P$ for any i. Finally, we check that

$$P^\top v = P^\top \sum_{i=1}^{k} \frac{(-1)^i}{2^i i!} A^i P P^\top u = \sum_{i=1}^{k} \frac{(-1)^i}{2^i i!} (P^\top AP)^i P^\top u$$

Thus, we can apply Lemma B.7 with $S = -\frac{1}{2} P^\top AP$ and the input unit vector being $P^\top u$ to obtain

$$\left\| \exp(-\frac{1}{2} P^\top AP) P^\top u - P^\top v \right\| \leq \left\| \exp(-\frac{1}{2} P^\top AP) \right\| \tau = \left\| \exp(-\frac{1}{2} A)\Big|_{\mathcal{U}_\pi^\perp} \right\| \tau$$

Finally, to bound the runtime, we see that it suffices to take $k = \max(e^2 \left\| P^\top AP \right\|, \ln(1/\tau))$. We can bound the matrix norm by $\|A\|$ since P is unitary. Since each M_t has spectral norm at most ρ, we have $\|A\| \leq \eta T$. In algorithms 6 and 7, we have $\eta = O(1)$ and $T = O(\log^3 n)$ and in algorithm 3, we have, $\eta = 1/\rho$ and $T = O(\rho^2 \log n)$. Thus, in the worst case, we have $\|A\| \leq \widetilde{O}(\rho)$. □

Now, we are ready to give our algorithm for approximating the matrix exponential:

Algorithm 10 Matrix Exponential

Input: a symmetric matrix: $A = \frac{\eta}{\rho} \sum_{i=1}^{t} M_i$, embedding dimension d, and accuracy parameter τ

1. Let U be a $n \times d$ matrix whose d columns form an orthogonal basis of a random $d = O(\log n)-$dimensional subspace orthogonal to the vector $\Pi^{1/2}\mathbb{1}$ with each column vector having length $\sqrt{n/d}$. Let U_1 be another random matrix defined similarly but whose columns are orthogonal to the vector $\mathbb{1}$ instead.

2. Pick $k \geq \Omega(\max(\rho, \log(1/\tau)))$. Compute $Z_\pi = \sum_{i=0}^{k} \frac{(-1)^i}{2^i i!} A^i \Pi^{-1/2} U_1$ and $Z = \sum_{i=0}^{k} \frac{(-1)^i}{2^i i!} A^i U$

3. Let $\hat{v}_1, \ldots \hat{v}_n$ be the rows of the matrix $Z_\pi / \sqrt{\mathrm{tr}(ZZ^\top)}$. Return these as the approximate embedding vectors.

For the sake of analysis, we will define the following matrices: let $W_\pi := \exp(-\frac{1}{2}A)\Pi^{-1/2}U_1 = \exp(-\frac{1}{2}A)|_{\mathcal{U}_\pi^\perp}\Pi^{-1/2}U_1$ and let $W := \exp(-\frac{1}{2}A)|U = \exp(-\frac{1}{2}A)|_{\mathcal{U}_\pi^\perp}U$. Note the second equalities follow from the fact that the columns of U and $\Pi^{-1/2}U_1$ are in $\mathcal{U}_\pi^\perp$. By applying the Johnson Lindenstrauss lemma, we can show that with good probability, the rows of the matrix W_π and W are good approximations to those of the matrix exponential.

Lemma B.9 (Johnson-Lindenstrauss Lemma ([JL84]))**.** *Let $x_1^\pi, \ldots x_n^\pi$ and $x_1, \ldots x_n$ be the rows of the matrices $\exp(-\frac{1}{2}A)|_{\mathcal{U}_\pi^\perp}\Pi^{-1/2}$ and $\exp(-\frac{1}{2}A)|_{\mathcal{U}_\pi}^\perp$ respectively. Let $r_1^\pi, \ldots r_n^\pi$ and $r_1, \ldots r_n$ be the rows of the matrices W_π and W respectively. For some $d = O(\frac{1}{\delta^2}\log n)$, we have that with probability $1 - n^{-1}$*

$$\left\| r_i^\pi - r_j^\pi \right\|^2 \in \left\| x_i^\pi - x_j^\pi \right\|^2 (1 \pm \delta) \quad \forall i, j \in V$$
$$\left\| r_i \right\|^2 \in \left\| x_i \right\|^2 (1 \pm \delta) \quad \forall i \in V$$

The following lemma shows that our approximations of W and W_π are also good for some $\tau = 1/\operatorname{poly}(\frac{n}{\pi_{\min}})$

Lemma B.10. *Let $Y' := W_\pi W_\pi^\top / \operatorname{tr}(WW^\top)$ and $Y'' = Z_\pi Z_\pi^\top / \operatorname{tr}(ZZ^\top)$. Suppose W satisfies $\operatorname{tr}(WW^\top) \in \operatorname{tr}(\exp(-A)|_{\mathcal{U}_\pi^\perp})(1 \pm \delta)$ for $\delta \le 1/5$. Then for small enough $\tau \in 1/\operatorname{poly}(\frac{n}{\pi_{\min}})$, we have*

$$\left\|Y'' - Y'\right\| \le O(\pi_{\min}^{-1} n^{3/2}) \cdot \tau$$

proof sketch. The proof of this lemma follows very closely to the proof of [Kal07, Lemma 25] so we will only sketch out the details. First, we define the error matrices $E_\pi = W_\pi - Z_\pi$ and $E = W - Z$. Let $w_1^\pi, \dots w_d^\pi$ and $z_1^\pi \dots z_d^\pi$ be the columns of the matrices W_π and Z_π respectively. By Lemma B.8, we have

$$\|E_\pi\|^2 \le \|E_\pi\|_F^2 = \sum_{i=1}^{d} \|w_i^\pi - z_i^\pi\|^2 \le d\pi_{\min}^{-1} \left\|\exp(-\frac{1}{2}A)|_{\mathcal{U}_\pi^\perp}\right\|^2 \tau^2 \le d\pi_{\min}^{-1} \left\|\exp(-A)|_{\mathcal{U}_\pi^\perp}\right\| \tau$$

Note that the second inequality follows from the fact that if u is a unit vector orthogonal to $\mathbb{1}$, then $\Pi^{-1/2}u$ is a vector of length at most $\pi_{\min}^{-1/2}$ orthogonal to $\Pi^{1/2}\mathbb{1}$. Similar calculations show that $||E||^2 \le d \left\|\exp(-A)|_{\mathcal{U}_\pi^\perp}\right\| \tau^2$. Using the E and E_π, matrices, we can bound the following matrix distances

$$\left\|W_\pi W_\pi^\top - Z_\pi Z_\pi^\top\right\| = \left\|E_\pi E_\pi^\top + E_\pi V_\pi^\top + V_\pi E_\pi^\top\right\| \le 3d\pi_{\min}^{-1} \left\|\exp(-A)|_{\mathcal{U}_\pi^\perp}\right\| \tau \tag{B.2}$$

And similar calculations show that

$$|\operatorname{tr}(WW^\top - ZZ^\top)| = |\operatorname{tr}(EE^\top + EV^\top + VE^\top)| \le 3d^{3/2} \left\|\exp(-A)|_{\mathcal{U}_\pi^\perp}\right\| \tau \tag{B.3}$$

Thus, we have

$$\begin{aligned}
\|Y' - Y''\| &\le \left\|\frac{W_\pi W_\pi^\top}{\operatorname{tr}(WW^\top)} - \frac{W_\pi W_\pi^\top}{\operatorname{tr}(VV^\top)}\right\| + \left\|\frac{W_\pi W_\pi^\top}{\operatorname{tr}(VV^\top)} - \frac{V_\pi V_\pi^\top}{\operatorname{tr}(VV^\top)}\right\| \\
&\le \frac{\left\|W_\pi W_\pi^\top\right\|}{\operatorname{tr}(WW^\top)} \cdot \frac{|\operatorname{tr}(WW^\top - \operatorname{tr}(ZZ^\top))|}{\operatorname{tr}(ZZ^\top)} + \frac{\left\|W_\pi W_\pi^\top - Z_\pi Z_\pi^\top\right\|}{\operatorname{tr}(ZZ^\top)} \\
&\le \pi_{\min}^{-1} \cdot \frac{3d^{3/2} \left\|\exp(-A)|_{\mathcal{U}_\pi^\perp}\right\| \tau}{\operatorname{tr}(ZZ^\top)} + \frac{3d\pi_{\min}^{-1} \left\|\exp(-A)|_{\mathcal{U}_\pi^\perp}\right\| \tau}{\operatorname{tr}(ZZ^\top)} \\
&\le \frac{6\pi_{\min}^{-1} d^{3/2} \left\|\exp(-A)|_{\mathcal{U}_\pi^\perp}\right\| \tau}{(1 - \delta - 3d^{3/2}\tau) \left\|\exp(-A)|_{\mathcal{U}_\pi^\perp}\right\| \tau} \\
&\le 8d^{3/2}\pi_{\min}^{-1}\tau
\end{aligned}$$

where the third inequality follows from bounds B.2, B.3 and $W_\pi W_\pi^\top \preccurlyeq \pi_{\min}^{-1} WW^\top$, and the fourth inequality follows from $\operatorname{tr}(WW^\top) \in (1 \pm \delta) \left\|\exp(-A)|_{\mathcal{U}_\pi^\perp}\right\|$ and B.3. □

Finally, we are ready to proof Lemma 4.18.

Proof of Lemma 4.18. Suppose we pick $\tau = \pi_{\min}/n$. By Lemma B.9, with probability at least $1 - d/n$, we have that for all $i, j \in V$

$$\|v_i - v_j\|^2 = \frac{\left\|x_i^\pi - x_j^\pi\right\|^2}{\sum_i \|x\|^2} \in \frac{\left\|r_i^\pi - r_j^\pi\right\|^2 (1 \pm \delta)}{\sum_i \|r_i\|^2 (1 \pm \delta)} = \langle L_{i,j}, Y' \rangle (1 \pm 2\delta)$$

and in particular, W satisfies $\mathrm{tr}(WW^\top) = \sum_i \|r\|_i^2 \in (1 \pm \delta) \sum_i \|x_i\|^2 = \mathrm{tr}(\exp(-A)|_{\mathcal{U}_\pi^\perp})(1 \pm \delta)$. Thus, Lemma B.10 implies that

$$|\langle L_{i,j}, Y' \rangle - \langle L_{i,j}, Y'' \rangle| \leq 2 \left\|Y - Y''\right\| \leq 1/n^{-c+3/2}$$

Since $\langle L_{i,j}, Y'' \rangle = \|\hat{v}_i - \hat{v}_j\|^2$, we have that $\|\hat{v}_i - \hat{v}_j\|^2 \in \|v_i - v_j\|^2 (1 \pm 2\delta) \pm n^{-\Omega(1)}$. Finally Lemma B.8 implies that the runtime of Algorithm 10 is $\widetilde{O}(\rho m)$. □

B.3 Fast Computation of Maximum Circulation

Recall that our main program in Definition 1.2 is to minimize the maximum edge-constrained circulation over all feasible embeddings $v_1, \ldots, v_n$. We remark that the inner maximization problem

$$\max_{F \in \mathcal{F}(G)} \sum_{i<j} \frac{1}{2}(F(i,j) + F(j,i)) \|v_i - v_j\|^2$$

is a special case of the minimum-cost flow in [CKLPPS22], each edge $e = ij$ having lower edge capacity 0, upper edge capacity $w(e)$, and cost $-\|v_i - v_j\|^2$ (so that it becomes a maximization problem), and each vertex i having demand $d(i) = 0$. Therefore, this problem can be computed in $O(m^{1+o(1)})$ time.

C Missing proofs of Section 6

Proof of Lemma 6.10. Suppose $\vec{f}$ is the non-saturating flow. We obtain a cut which consists of edges incident to either s or t and vertices of G, whose removal would make it impossible to go from s to t. Let $S \subseteq V$ be the set of vertices reachable from s after removing the cut edges and vertices. Let $V_s \subseteq L$ and $V_t \subseteq R$ be defined similarly to the edge-capacitated case.

Let $\nu \cdot \beta \cdot \pi(R)$ be the max-flow value of $\vec{f}$ with $\nu < 1$. Note that

$$\kappa \cdot \pi(\partial^+(S)) = \beta(\nu \cdot \pi(R) - r \cdot \pi(V_s) - \pi(V_t)),$$

because $\{si \mid i \in V_s\} \cup \partial_G^+(S) \cup \{jt \mid j \in V_t\}$ is the minimum s-t cut obtained, with total weight equal to $r \cdot \beta \cdot \pi(V_s) + \beta \cdot \pi(V_t) + \kappa \cdot \pi(\partial^+(S)) = \nu \cdot \beta \cdot \pi(R)$ by our construction of $\vec{G}$. Also, since $L \setminus (V_s \cup \partial^+(S)) \subseteq S$ and $R \setminus (V_t \cup \partial^+(S)) \subseteq V - S$, it follows that

$$\pi(S) \geq \pi(L) - \pi(V_s) - \pi(\partial^+(S)) \quad \text{and} \quad \pi(V - S) \geq \pi(R) - \pi(V_t) - \pi(\partial^+(S)).$$

Therefore,

$$\begin{aligned}
\frac{1}{\vec{\psi}_\pi(S)} = \frac{\min\{\pi(S), \pi(V-S)\}}{\pi(\partial^+(S))} &\geq \frac{\min\{\pi(R) - \pi(V_t) - \pi(\partial^+(S)),\ \pi(L) - \pi(V_s) - \pi(\partial^+(S))\}}{\pi(\partial^+(S))} \\
&= \frac{\kappa}{\beta} \cdot \frac{\min\{\pi(R) - \pi(V_t),\ \pi(L) - \pi(V_s)\}}{\nu \cdot \pi(R) - r \cdot \pi(V_s) - \pi(V_t)} - 1 \\
&\geq \frac{\kappa}{\beta} \cdot \min\left\{\frac{\pi(R) - \pi(V_t)}{\pi(R) - \pi(V_t)},\ \frac{\pi(L) - \pi(V_s))}{r \cdot \pi(L) - r \cdot \pi(V_s)}\right\} - 1 \\
&= \frac{\kappa}{\beta \cdot r'} - 1,
\end{aligned}$$

where the last inequality is because $\nu < 1$ and $\pi(R) = r \cdot \pi(L)$. Rearranging gives the desired conclusion. □