A Simpler and Parallelizable $O(\sqrt{\log n})$-approximation Algorithm for Sparsest Cut

Vladimir Kolmogorov
vnk@ist.ac.at
Institute of Science and Technology Austria (ISTA)

ABSTRACT

Currently, the best known tradeoff between approximation ratio and complexity for the Sparsest Cut problem is achieved by the algorithm in [Sherman, FOCS 2009]: it computes $O(\sqrt{(\log n)/\varepsilon})$-approximation using $O(n^{\varepsilon}\log^{O(1)} n)$ maxflows for any $\varepsilon \in [\Theta(1/\log n), \Theta(1)]$. It works by solving the SDP relaxation of [Arora-Rao-Vazirani, STOC 2004] using the Multiplicative Weights Update algorithm (MW) of [Arora-Kale, JACM 2016]. To implement one MW step, Sherman approximately solves a multicommodity flow problem using another application of MW. Nested MW steps are solved via a certain "chaining" algorithm that combines results of multiple calls to the maxflow algorithm.

We present an alternative approach that avoids solving the multicommodity flow problem and instead computes "violating paths". This simplifies Sherman's algorithm by removing a need for a nested application of MW, and also allows parallelization: we show how to compute $O(\sqrt{(\log n)/\varepsilon})$-approximation via $O(\log^{O(1)} n)$ maxflows using $O(n^{\varepsilon})$ processors.

We also revisit Sherman's chaining algorithm, and present a simpler version together with a new analysis.

CCS CONCEPTS

• **Theory of computation → Parallel algorithms**.

KEYWORDS

Sparsest Cut, approximation algorithms, parallel algorithms

ACM Reference Format:
Vladimir Kolmogorov. 2024. A Simpler and Parallelizable $O(\sqrt{\log n})$-approximation Algorithm for Sparsest Cut. In *Proceedings of the 36th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA '24), June 17–21, 2024, Nantes, France.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3626183.3659969

1 INTRODUCTION

Partitioning a given undirected graph $G = (V, E)$ into two (or more) components is a fundamental problem in computer science with many real-world applications, ranging from data clustering and network analysis to parallel computing and VLSI design. Usually, desired partitions should satisfy two properties: (i) the total cost of edges between different components should be small, and (ii) the components should be sufficiently balanced. For partitions with two components $(S, \bar{S})$ this means that $E(S, \bar{S})$ should be small and $\min\{|S|, |\bar{S}|\}$ should be large, where $E(S, \bar{S})$ is the total number of edges between S and $\bar{S}$ (or their total weight in the case of weighted graphs).

One of the most widely studied versions is the Sparsest Cut problem whose goal is to minimize the ratio $\frac{E(S,\bar{S})}{\min\{|S|,|\bar{S}|\}}$ (called *edge expansion*) over partitions $(S, \bar{S})$. Another well-known variant is the c-Balanced Separator problem: minimize $E(S, \bar{S})$ over *c-balanced partitions*, i.e. partitions satisfying $\min\{|S|, |\bar{S}|\} \geq cn$ where $n = |V|$ and $c \in (0, \frac{1}{2})$ is a given constant.

Both problems are NP-hard, which forces one to study approximation algorithms, or pseudoapproximation algorithms in the case of Balanced Separator. (An algorithm for Balanced Separator is said to be a λ-pseudoapproximation if it computes a c'-balanced partition $(S, \bar{S})$ whose cost $E(S, \bar{S})$ is at most λ times the optimal cost of the c-Balanced Partition problem, for some constants $c' \leq c$). Below we discuss known results for the Sparsest Cut problem. They also apply to Balanced Separator: in all previous works, whenever there is an $O(\lambda)$-approximation algorithm for Sparsest Cut then there is an $O(\lambda)$-pseudoapproximation algorithm for Balanced Separator with the same complexity.

The first nontrivial guarantee was obtained by Leighton and Rao [17], who presented a $O(\log n)$-approximation algorithm based on a certain LP relaxation of the problem. The approximation factor was improved to $O(\sqrt{\log n})$ in another seminal paper by Arora, Rao and Vazirani [5] who used an SDP relaxation. Arora, Hazan and Kale [3] showed how to (approximately) solve this SDP in $\tilde{O}(n^2)$ time using multicommodity flows while preserving the $O(\sqrt{\log n})$ approximation factor. Arora and Kale later developed in [4] a more general method for solving SDPs that allowed different tradeoffs between approximation factor and complexity; in particular, they presented an $O(\log n)$-approximation algorithm using $O(\log^{O(1)} n)$ maxflow computations, and a simpler version of $O(\sqrt{\log n})$-approximation with $\tilde{O}(n^2)$ complexity.

The algorithms in [3, 4] were based on the Multiplicative Weights Update method. An alternative approach based on the so-called *cut-matching game* was proposed by Khandekar, Rao and Vazirani [12]; their method computes $O(\log^2 n)$-approximation for Sparsest Cut using $O(\log^{O(1)} n)$ maxflows. This was later improved to $O(\log n)$-approximation by Orecchia, Schulman, Vazirani and Vishnoi [20].

The line of works above culminated in the result of Sherman [24], who showed how to compute $O(\sqrt{(\log n)/\varepsilon})$-approximation for Sparsest Cut using $O(n^{\varepsilon}\log^{O(1)} n)$ maxflows for any $\varepsilon \in [\Theta(1/\log n), \Theta(1)]$. This effectively subsumes previous results, as taking $\varepsilon = \Theta(1/\log n)$ yields an $O(\log n)$ approximation using $O(\log^{O(1)} n)$ maxflows, while a sufficiently small constant ε achieves an $O(\sqrt{\log n})$-approximation and improves on the $\tilde{O}(n^2)$

SPAA '24, June 17–21, 2024, Nantes, France

ACM ISBN 979-8-4007-0416-1/24/06
https://doi.org/10.1145/3626183.3659969

runtime in [3, 4]. In particular, using the recent almost linear-time maxflow algorithm [9] yields $O(n^{1+\varepsilon})$ complexity for $O(\sqrt{\log n})$-approximation. (As usual, we assume in this paper that graph G has $m = O(n \log n)$ edges. This can be achieved by sparsifying the graph using the algorithm of Benczúr and Karger [6], which with high probability preserves the cost of all cuts up to any given constant factor.)

Our contributions In this paper we present a new algorithm that computes an $O(\sqrt{(\log n)/\varepsilon})$ approximation for Sparsest Cut w.h.p. whose expected runtime is $O((n^{\varepsilon} \log^{O(1)} n) \cdot T_{\texttt{maxflow}})$ for given $\varepsilon \in [\Theta(1/\log n), \Theta(1)]$, where $T_{\texttt{maxflow}} = \Omega(n)$ is the runtime of a maxflow algorithm on a graph with n nodes and $O(n \log n)$ edges. It has the following features:

(i) It simplifies Sherman's algorithm in two different ways.

(ii) The algorithm is parallelizable: it can be implemented on $O(n^{\varepsilon})$ processors in expected parallel runtime $O((\log^{O(1)} n) \cdot T_{\texttt{maxflow}})$ (in any version of the PRAM model), where "parallel runtime" is defined as the maximum runtime over all processors. Note that this is an exponential improvement over complexity $O((n^{\varepsilon} \log^{O(1)} n) \cdot T_{\texttt{maxflow}})$ of the sequential version.

(iii) To prove algorithm's correctness, we introduce a new technique, which we believe may yield smaller constants in the $O(\cdot)$ notation. Note that there are numerous papers that optimize constants for problems with a constant factor approximation guarantee. We argue that this direction makes just as much sense for the Sparsest Cut problem. The question can be naturally formulated as follows: what is the fastest algorithm to compute $C\sqrt{\log n}$-approximation for a given constant C? Alternatively, for maxflow-based algorithms one may ask what is the smallest $C = C_{\varepsilon}$ such that there is a $C\sqrt{\log n}$-approximation algorithm that uses $\tilde{O}(n^{\varepsilon})$ maxflow, for given $\varepsilon > 0$. Unfortunately, it is impossible to directly compare our constant with that of Sherman: we believe that the paper [24] contains a numerical mistake (see the footnote in Section 3.4). An additional complication is that optimizing the constants may not be an easy task. Due to these considerations, we formulate our claim differently: our proof technique should lead to smaller constants since our analysis is more compact and avoids case analysis present in [24].

To explain details, we need to give some background on Sherman's algorithm. It builds on the work of Arora and Kale [4] who approximately solve an SDP relaxation using a (matrix) Multiplicative Weights update algorithm (MW). The key subroutine is to identify constraints that are violated by the current primal solution. Both [4] and [24] do this by solving a multicommodity flow problem. Arora and Kale simply call Fleischer's multicommodity flow algorithm [10], while Sherman designs a more efficient customized method for approximately solving this flow problem using another application of MW. Our algorithm avoids solving the multicommodity flow problem, and instead searches for "violating paths", i.e. paths that violate triangle inequalities in the SDP relaxation. We show that this can be done by a simple randomized procedure that does not rely on MW. Furthermore, independent calls to this procedure return violating paths that are mostly disjoint, which allows parallelization: we can compute many such paths on different processors and then take their union.

In order to compute violating paths, we first design procedure $\texttt{Matching}(u)$ that takes vector $u \in \mathbb{R}^d$ and outputs a directed matching on a given set $S \subset \mathbb{R}^d$ with $|S| = \Theta(n)$. (It works by calling a maxflow algorithm and then postprocessing the flow). The problem then boils down to the following: given vector u randomly sampled from the Gaussian distribution, we need to sample vectors $u_1, \dots, u_K$ so that set $\texttt{Matching}(u_1) \circ \dots \circ \texttt{Matching}(u_K)$ contains many paths $(x_0, x_1, \dots, x_K)$ with "large stretch", i.e. with $\langle x_K - x_0, u \rangle \geq \Omega(K)$. (Here "$\circ$" is the operation that "chains together" paths in a natural way). This was also a key task in [24], where it was needed for implementing one inner MW step.

Sherman's chaining algorithm can be viewed as an algorithmization of the proof in the original ARV paper [5] and its subsequent improvement by Lee [16]. We present a simpler chaining algorithm with a very different proof; as stated before, the new proof may yield smaller constants.

Concurrent work After finishing the first draft of the paper [13], we learned about a very recent work by Lau-Tung-Wang [15], which considers a generalization of Sparsest Cut to *directed* graphs. The authors presented an algorithm which, in the case of undirected graphs, also simplifies Sherman's algorithm by computing violating paths instead of solving a multicommodity flow problem. Unlike our paper, [15] does not consider parallelization, and uses Sherman's chaining algorithm as a black box.

Another very recent paper by Agarwal-Khanna-Li-Patil-Wang-White-Zhong [1] presented a parallel algorithm for *approximate maxflow* with polylogarithmic depth and near-linear work in the PRAM model. Using this algorithm, they presented, in particular, an $O(\log^3 n)$-approximation sparsest cut algorithm with polylogarithmic depth and near-linear work (by building on the work [19] who showed how to replace exact maxflow computations in the cut-matching game [12, 20] with approximate maxflows).

Other related work The problem of computing a cut of conductance $\tilde{O}(\sqrt{\phi})$ assuming the existence of a cut of conductance $\phi \in [0, 1]$ (with various conditions on the balancedness) has been considered in [14, 21, 23]. Papers [14, 23] presented distributed algorithms for this problem (in the CONGEST model), while [22] observed that the BalCut algorithm in [21] is parallelizable: it can be implemented in near-linear work and polylogarithmic depth. Note that ϕ can be much smaller than 1. [7] presented a distributed CONGEST algorithm for computing an expander decomposition of a graph.

2 BACKGROUND: ARORA-KALE FRAMEWORK

We will describe the algorithm only for the c-Balanced Separator problem. As shown in [4], the Sparsest Cut problem can be solved by a very similar approach, essentially by reducing it to the c-Balanced Separator problem for some constant c; we refer to [4] for details.[1]

Let $c_e \geq 0$ be the weight of edge e in G. The standard SDP relaxation of the c-Balanced Separator problem can be written

[1] These details are actually not included in the journal version [4], but can be found in [11] or in https://www.cs.princeton.edu/~arora/pubs/mmw.pdf, Appendix A.

in vector form as follows:

$$\min \sum_{e=\{i,j\}\in E} c_e ||v_i - v_j||^2 \tag{1a}$$

$$||v_i||^2 = 1 \qquad \forall i \tag{1b}$$

$$||v_i - v_j||^2 + ||v_j - v_k||^2 \geq ||v_i - v_k||^2 \qquad \forall i,j,k \tag{1c}$$

$$\sum_{i<j} ||v_i - v_j||^2 \geq 4c(1-c)n^2 \tag{1d}$$

The optimum of this SDP divided by 4 is a lower bound on the minimum c-Balanced Separator problem. Arora and Kale considered a slightly different relaxation:

$$\min \sum_{e=\{i,j\}\in E} c_e ||v_i - v_j||^2 \qquad \min C \bullet X \tag{2a}$$

$$||v_i||^2 = 1 \qquad X_{ii} = 1 \qquad \forall i \tag{2b}$$

$$\sum_{j=1}^{\ell(p)} ||v_{p_j} - v_{p_{j-1}}||^2 \geq ||v_{p_{\ell(p)}} - v_{p_0}||^2 \qquad T_p \bullet X \geq 0 \qquad \forall p \tag{2c}$$

$$\sum_{i,j\in S: i<j} ||v_i - v_j||^2 \geq an^2 \qquad K_S \bullet X \geq an^2 \qquad \forall S \tag{2d}$$

$$X \succeq 0 \tag{2e}$$

Here p stands for a path $p = (p_0, \ldots, p_{\ell(p)})$ in graph G, notation "$\forall S$" means all subsets $S \subseteq V$ of size at least $(1 - c/4)n$, and $a = 3c - 4c^2$. Note that triangle inequalities (1c) imply path inequalities (2c), while constraints (1b) and (1d) imply constraints (2d) (see [4]). SDP (2) may be looser than (1), but its optimum divided by 4 is still a lower bound on the minimum c-Balanced Separator problem.

The dual of (2) is as follows. It has variables y_i for every node i, f_p for every path p, and z_S for every set S of size at least $(1 - c/4)n$. Let $\mathtt{diag}(y)$ be the diagonal matrix with vector y on the diagonal:

$$\max \sum_i y_i + an^2 \sum_S z_S \tag{3a}$$

$$\mathtt{diag}(y) + \sum_p f_p T_p + \sum_S z_S K_S \preceq C \tag{3b}$$

$$f_p, z_S \geq 0 \quad \forall p, S \tag{3c}$$

2.1 Matrix multiplicative weights algorithm

To solve the above SDP, [4] converts an optimization problem to a feasibility problem by replacing the objective (2a) with the constraint

$$C \bullet X \leq \alpha \tag{2a$'$}$$

As shown in [3], it suffices to try $O(\log n)$ values of threshold α if we are willing to accept the loss by a constant factor in the approximation ratio. Let (2$'$) be the system consisting of constraints (2a$'$) and (2b)-(2e). To check the feasibility of this system, Arora and Kale apply the Matrix Multiplicative Weights (MW) algorithm which we review in Appendix A. The main computational subroutine is procedure Oracle that, given current matrix X of the form $X = V^T V$, $V \in \mathbb{R}^{n\times n}$, should either (i) find an inequality violated by X, or (ii) find a $\Theta(1)$-balanced cut of value at most $\kappa\alpha$ where $\Theta(\kappa)$ is the desired approximation factor. Working directly with matrix V would be too slow (even storing it requires $\Theta(n^2)$ space and thus $\Omega(n^2)$ time). To reduce complexity, Arora and Kale work instead with matrix $\tilde{V} \in \mathbb{R}^{d\times n}$, $d \ll n$ so that $X \approx \tilde{X} \stackrel{\text{def}}{=} \tilde{V}^T\tilde{V}$. Let $v_1, \ldots, v_n \in \mathbb{R}^n$ and $\tilde{v}_1, \ldots, \tilde{v}_n \in \mathbb{R}^d$ be the columns of V and $\tilde{V}$, respectively. Below we give a formal specification of Oracle. Note that the oracle has access only to vectors $\tilde{v}_1, \ldots, \tilde{v}_n$.

Input: vectors $v_1, \ldots, v_n \in \mathbb{R}^n$ and $\tilde{v}_1, \ldots, \tilde{v}_n \in \mathbb{R}^d$ satisfying

$$||\tilde{v}_i||^2 \leq 2 \qquad \forall i \tag{4a}$$

$$\sum_{i,j\in V: i<j} ||\tilde{v}_i - \tilde{v}_j||^2 \geq \tfrac{an^2}{4} \tag{4b}$$

$$\left|\, ||\tilde{v}_i||^2 - ||v_i||^2 \,\right| \leq \gamma(||\tilde{v}_i||^2 + \tau) \qquad \forall i \tag{4c}$$

$$\left|\, ||\tilde{v}_i - \tilde{v}_j||^2 - ||v_i - v_j||^2 \,\right| \leq \gamma(||\tilde{v}_i - \tilde{v}_j||^2 + \tau) \qquad \forall i,j \tag{4d}$$

for some constants $\gamma, \tau > 0$. Let $X = V^T V$ and $\tilde{X} = \tilde{V}^T\tilde{V}$ where $V \in \mathbb{R}^{n\times n}$ and $\tilde{V} \in \mathbb{R}^{d\times n}$ are the matrices with columns $\{v_i\}$ and $\{\tilde{v}_i\}$, respectively.

Output: either (i) variables $f_p \geq 0$ and symmetric matrix $F \preceq C$ such that

$$\left(\sum_p f_p T_p - F\right) \bullet X \leq -\alpha \tag{5}$$

or (ii) a $\Theta(1)$-balanced cut of value at most $\kappa\alpha$.

The number of iterations of the MW algorithm will depend on the maximum possible spectral norm of matrix $\frac{\alpha}{n} I + \sum_p f_p T_p - F$. This parameter is called the *width* of the oracle, and will be denoted as ρ. We will use the bound $\rho = ||\frac{\alpha}{n} I + \sum_p f_p T_p - F|| \leq \frac{\alpha}{n} + ||\sum_p f_p T_p - F||$.

Theorem 2.1 ([4]). *If in the MW algorithm the first* $T = \lceil \frac{4\rho^2 n^2 \ln n}{\epsilon^2} \rceil$ *calls to* Oracle *output option* (i) *then the optimum value of SDP* (2) *is at least* $\alpha - \epsilon$.

2.2 Oracle implementation

To implement the oracle, Arora and Kale interpret values f_p as a *multicommodity flow* in graph G, i.e. a flow that sends f_p units of demand between the endpoints of p. Given this flow, introduce the following notation. Let f_e be the flow on edge e (i.e. $f_e = \sum_{p\ni e} f_p$). Let d_{ij} be the total flow between nodes i and j (i.e. $d_{ij} = \sum_{p\in\mathcal{P}_{ij}} f_p$ where $\mathcal{P}_{ij}$ is the set of paths from i to j). Finally, let d_i be the total flow from node i (i.e. $d_i = \sum_j d_{ij}$). Given parameter $\pi > 0$, a *valid π-regular flow* is one that satisfies capacity constraints: $f_e \leq c_e$ for all edges e and $d_i \leq \pi$ for all nodes i.

The oracle in [4] computes a π-regular flow f for some parameter π, and sets F to be the Laplacian of the *flow graph* (i.e. the weighted graph where edge e has weight f_e). Capacity constraints then ensure that $F \preceq C$ (because $C - F$ is the Laplacian of the weighted graph with weights $c_e - f_e \geq 0$ on edges $e \in E$). Let D be the Laplacian of the *demand graph* (i.e. the complete weighted graph where edge $\{i,j\}$ has weight d_{ij}). It can be checked that $\sum_p f_p T_p = F - D$. Thus, the oracle needs to ensure that $D \bullet X \geq \alpha$, or equivalently

$$\sum_{i<j} d_{ij} ||v_i - v_j||^2 \geq \alpha \tag{6}$$

All degrees in the demand graph are bounded by π, therefore $||D|| \leq 2\pi$. Thus, the width of the oracle can be bounded as $||\rho|| \leq \frac{\alpha}{n} + ||D|| \leq \frac{\alpha}{n} + 2\pi$.

Below we summarize three known implementations of the oracle. The first two are due to Arora and Kale [4] and the third one is due to Sherman [24].

(1) Using $O(1)$ expected maxflow computations, the oracle computes either a π-regular flow with $\pi = O(\frac{\alpha \log n}{n})$ or a $\Theta(1)$-balanced cut of capacity at most $O(\alpha \log n)$.
(2) Using $O(1)$ expected multicommodity flow computations, the oracle computes either a π-regular flow with $\pi = O(\frac{\alpha}{n})$ or a $\Theta(1)$-balanced cut of capacity at most $O(\alpha\sqrt{\log n})$.
(3) Let $\varepsilon \in [O(1/\log n), \Omega(1)]$. Using $O(n^{\varepsilon} \log^{O(1)} n)$ expected maxflow computations, the oracle either computes a π-regular flow with $\pi = O(\frac{\alpha}{\varepsilon n})$, or a $\Theta(1)$-balanced cut of capacity at most $O\left(\alpha\sqrt{\frac{\log n}{\varepsilon}}\right)$.

By the discussion in Section 2.1, these oracles lead to algorithms with approximation factors $O(\log n)$, $O(\sqrt{\log n})$ and $O\left(\sqrt{\frac{\log n}{\varepsilon}}\right)$, respectively.

To conclude this section, we discuss how to verify condition (6) in practice. (Recall that we only have an access to approximations $\tilde{v}_i$ of vectors v_i.)

Proposition 2.2. *Suppose parameters τ, γ in eq.* (4c)-(4d) *satisfy $\tau \le 2$ and $\gamma \le \frac{\alpha}{20n\pi}$. Then condition*

$$\sum_{i<j} d_{ij} ||\tilde{v}_i - \tilde{v}_j||^2 \ge 2\alpha \tag{7}$$

implies condition (6).

Proof. Denote $z_{ij} = ||v_i - v_j||^2$ and $\tilde{z}_{ij} = ||\tilde{v}_i - \tilde{v}_j||^2$. We have $||\tilde{v}_i||^2 \le 2$, $||\tilde{v}_j||^2 \le 2$ and hence $\tilde{z}_{ij} \le 8$. By Theorem A.2 we then have $|\tilde{z}_{ij} - z_{ij}| \le \gamma(\tilde{z}_{ij} + \tau) \le 10\gamma$. This implies that $\sum_{i<j} d_{ij}|\tilde{z}_{ij} - z_{ij}| \le 10\gamma \cdot \sum_{i<j} d_{ij} \le 10\gamma \cdot 2n\pi \le \alpha$. The claim follows. □

3 OUR ALGORITHM

In this section we present our implementation of the oracle. To simplify notation, we assume in this section that vectors $\tilde{v}_i$ for $i \in V$ are unique, and rename the nodes in V so that $\tilde{v}_x = x$ for each $x \in V$. Thus, we now have $V \subseteq \mathbb{R}^d$. The "true" vector in $\mathbb{R}^n$ corresponding to $x \in V$ is still denoted as v_x.

Recall that the oracles in [4, 24] do one of the following:

- *output a cut;*
- *output multicommodity flows f_p satisfying* (6)*, and set F to be its flow graph.*

Our oracle will use a third option described in the lemma below.

Lemma 3.1. *Let M be a set of paths on V such that each $p \in M$ violates the path inequality by some amount $\frac{1}{2}\Delta > 0$. In other words, we require that $T_p \bullet X \le -\frac{1}{2}\Delta$, or equivalently*

$$\sum_{j=1}^{\ell(p)} ||v_{p_j} - v_{p_{j-1}}||^2 \;\le\; ||v_{p_{\ell(p)}} - v_{p_0}||^2 - \tfrac{1}{2}\Delta \tag{8}$$

Let $\mathcal{G}_F$ and $\mathcal{G}_D$ be respectively flow and demand graphs of the multicommodity flow defined by M (where each path carries one unit of flow). Set $f_p = \frac{2\alpha}{|M|\Delta}$ for all $p \in M$, $f_p = 0$ for $p \notin M$, and $F = 0$. Then these variables give a valid output of the oracle with width $\rho \le \frac{\alpha}{n} + \frac{4\alpha(\pi_F + \pi_D)}{|M|\Delta}$ where π_F, π_D are the maximum degrees of $\mathcal{G}_F, \mathcal{G}_D$, respectively.

Proof. We have $F \preceq C$ and $(\sum_p T_p - F) \bullet X \le \sum_p f_p \cdot (-\frac{1}{2}\Delta) = -\alpha$, so condition (5) holds. It can be checked that $\sum_p f_p T_p = \frac{2\alpha}{|M|\Delta}(\tilde{F} - \tilde{D})$ where $\tilde{F}$ and $\tilde{D}$ are the Laplacians of respectively $\mathcal{G}_F$ and $\mathcal{G}_D$. Therefore, $\rho = ||\frac{\alpha}{n} I + \sum_p f_p T_p - F|| = ||\frac{\alpha}{n} I + \frac{2\alpha}{|M|\Delta}(\tilde{F} - \tilde{D})|| \le ||\frac{\alpha}{n} I|| + \frac{2\alpha}{|M|\Delta}(||\tilde{F}|| + ||\tilde{D}||) \le \frac{\alpha}{n} + \frac{2\alpha}{|M|\Delta}(2\pi_F + 2\pi_D)$. □

Note that we do not require paths in M to be paths in the original graph G. Thus, we need to use SDP relaxation (2) with all possible paths p (with a certain bound on the length), which is still a valid relaxation of the problem.

The following proposition shows how to verify condition (8) for unobserved variables $v_x \in \mathbb{R}^n$ using observed variables $x \in \mathbb{R}^d$. Its proof is very similar to that of Proposition 2.2, and is omitted.

Proposition 3.2. *Suppose parameters τ, γ in Theorem A.2 satisfy $\tau \le 2$ and $\gamma \le \frac{\Delta}{20(K+1)}$ for some integer $K \ge 1$. Then condition*

$$\sum_{j=1}^{\ell(p)} ||p_j - p_{j-1}||^2 \;\le\; ||p_{\ell(p)} - p_0||^2 - \Delta \tag{9}$$

implies condition (8)*, assuming that $\ell(p) \le K$.*

Remark 1. *Sherman [24] explicitly tries to (approximately) solve the multicommodity flow problem: find valid flows $\{f_p\}$ in G with demands $\{d_{xy}\}$ that maximize $\sum_{x,y} d_{xy}||x - y||^2$. This is done by an iterative scheme via the Multiplicative Weights (MW) framework. Using the option in Lemma 3.1 has the following advantages over this approach.*

(1) We can avoid another application of MW and thus simplify the algorithm.
(2) The oracle can be easily parallelized: we can compute different "violating paths" on different processors and then take their union. (Of course, we still need to make sure that these paths are "sufficiently disjoint" so that the degrees of graphs $\mathcal{G}_F, \mathcal{G}_D$ and hence the oracle width remain small).
In contrast, parallelizing an MW algorithm cay be a difficult task. (There exists some work on parallelizing algorithms involving MW, e.g. [2, 8]; however, they consider very specific problems.)

3.1 Correlated Gaussians and measure concentration

We write $u \sim \mathcal{N}$ to indicate that u is a random vector in $\mathbb{R}^d$ with Gaussian independent components $u_i \sim \mathcal{N}(0, 1)$. Throughout the paper notation $\mathtt{Pr}_u[\cdot]$ means the probability under distribution $u \sim \mathcal{N}$. We write $(u, u') \sim \mathcal{N}_\omega$ for $\omega \in [0, 1)$ to indicate that (u, u') are random vectors in $\mathbb{R}^d \times \mathbb{R}^d$ such that for each $i \in [d]$, pair (u_i, u'_i) is an independent 2-dimensional Gaussian with mean $(0, 0)^T$ and covariance matrix $\begin{pmatrix} 1 & \omega \\ \omega & 1 \end{pmatrix}$. We write $u' \sim_\omega u$ to indicate that u' is an *ω-correlated copy of u* [24], i.e. (u, u') is generated according to $(u, u') \sim \mathcal{N}_\omega$ conditioned on fixed u. It can be checked that for each $i \in [d]$, u'_i is an independent Gaussian with mean $\omega \cdot u_i$ and variance $1 - \omega^2$. Note, if $(u, u') \sim \mathcal{N}_\omega$ then $u \sim \mathcal{N}$ and $u' \sim \mathcal{N}$.

Conversely, the process $u \sim \mathcal{N}, \hat{u} \sim_\omega u$ generates pair (u, u') with distribution $\mathcal{N}_\omega$. The same is true for the process $u' \sim \mathcal{N}, u \sim_\omega u'$.

The key property for obtaining an $O(\sqrt{\log n})$-approximation algorithm is *measure concentration* of the Gaussian distribution. This property can be expressed in a number of different ways; we will use the following version.[2]

Theorem 3.3 ([18]). *Consider sets $\mathcal{A} \subseteq \mathbb{R}^d$, $\mathcal{B} \subseteq \mathbb{R}^d$ with $\mathtt{Pr}_u[u \in \mathcal{A}] = \mathtt{Pr}_{u'}[u' \in \mathcal{B}] = \delta$. Then*

$$\mathtt{Pr}_{(u,u') \sim \mathcal{N}_\omega}[(u, u') \in \mathcal{A} \times \mathcal{B}] \geq \delta^{2/(1-\omega)}$$

Given a sequence of numbers $\omega_1, \ldots, \omega_{k-1} \in [0, 1)$, we write $(u_1, \ldots, u_k) \sim \mathcal{N}_{\omega_1,\ldots,\omega_{k-1}}$ to indicate the following distribution: sample $u_1 \sim \mathcal{N}$, then $u_2 \sim_{\omega_1} u_1$, then $u_3 \sim_{\omega_2} u_2$, ..., then $u_k \sim_{\omega_{k-1}} u_{k-1}$. If $\omega_1 = \ldots = \omega_{k-1} = \omega$ then we write $\mathcal{N}_\omega^k$ instead of $\mathcal{N}_{\omega_1,\ldots,\omega_{k-1}}$ for brevity. Finally, if some values of the sequence $(u_1, \ldots, u_k)$ are fixed, e.g. (u_1, u_k), then we write $(u_1, \ldots, u_k) \sim \mathcal{N}_{\omega_1,\ldots,\omega_{k-1}}|(u_1, u_k)$ to indicate that $(u_1, \ldots, u_k)$ is obtained by sampling from $\mathcal{N}_{\omega_1,\ldots,\omega_{k-1}}$ conditioned on fixed values (u_1, u_k). In that case $(u_2, \ldots, u_{k-1})$ are random variables that depend on (u_1, u_k).

3.2 Procedure Matching(u)

In this section we describe a procedure that takes vector $u \in \mathbb{R}^d$ and either outputs a directed matching M on nodes V or terminates the oracle. In this procedure we choose constants c', Δ, σ (to be specified later), and denote

$$\pi = \frac{6\alpha}{c' n \Delta} \tag{10}$$

Algorithm 1: Matching(u).

1. compute $w_x = \langle x, u \rangle$ for each $x \in V$
2. sort $\{w_x\}_{x \in V}$, let A, B be subsets of V with $|A| = |B| = 2c'n$ containing nodes with the least and the greatest values of w_x, respectively
3. let G' be the graph obtained from G by adding new vertices s, t and edges $\{\{s, x\} : x \in A\} \cup \{\{y, t\} : y \in B\}$ of capacity π
4. compute maximum s-t flow and the corresponding minimum s-t cut in G'
5. if capacity of the cut is less than $c'n\pi = \frac{6\alpha}{\Delta}$ then return this cut and terminate the oracle
6. use flow decomposition to compute multicommodity flows f_p and demands d_{xy} (see text)
7. if flows f_p satisfy condition (7) then return these flows and terminate the oracle
8. let $M_{\mathtt{all}} = \{(x, y) \in A \times B : d_{xy} > 0, w_y - w_x \geq \sigma\}$ and $M_{\mathtt{short}} = \{(x, y) \in M_{\mathtt{all}} : ||x - y||^2 \leq \Delta\}$
9. pick maximal matching $M \subseteq M_{\mathtt{short}}$ and return M

Let us elaborate line 6. Given flow f' in G', we compute its flow decomposition and remove flow cycles. Each path in this decomposition has the form $p' = (s, p, t)$ where $p = (x, \ldots, y)$ with $x \in A$, $y \in B$. For each such p we set $f_p = f'_{p'}$, and accordingly increase demand d_{xy} by $f'_{p'}$. Note that we need to know only the endpoints of p, and not p itself. This computation can be done in $O(m \log n)$ time using dynamic trees [25]. (The same subroutine was used in [24]).

For the purpose of analysis we make the following assumption: if Algorithm 1 terminates at line 5 or 7 then it returns $\emptyset$ (the empty matching). Thus, we always have $\mathtt{Matching}(u) \subseteq V \times V$ and $|\mathtt{Matching}(u)| \leq |V|$. The following result is proved in Appendix B.

Lemma 3.4. *(a) If the algorithm terminates at line 5 then the returned cut is c'-balanced.*
(b) There exist positive constants c', σ, δ for which either (i) $\mathbb{E}_u|\mathtt{Matching}(u)| \geq \delta n$, or (ii) Algorithm 1 for $u \sim \mathcal{N}$ terminates at line 5 or 7 with probability at least $\Theta(1)$.

In the remainder of the analysis we assume that case (i) holds in Lemma 3.4(b). (In the case of case (ii) procedures that we will describe will terminate the oracle at line 5 or 7 with probability $\Theta(1)$).

We assume that procedure $\mathtt{Matching}(\cdot)$ satisfies the following skew-symmetry condition: for any u, matching $\mathtt{Matching}(-u)$ is obtained from $\mathtt{Matching}(u)$ by reversing all edge orientations. (This can be easily enforced algorithmically).

3.3 Matching covers

We say that a *generalized matching* is a set M of paths of the form $p = (p_0, \ldots, p_k)$ with $p_0, \ldots, p_k \in V$ such that each node $x \in V$ has at most one incoming and at most one outgoing path. We say that path q is *violating* if $q = (\ldots, p, \ldots)$ and path $p = (p_0, \ldots, p_{\ell(p)})$ satisfies (9). We denote $M^{\mathtt{violating}}$ and $M^{\mathtt{nonviolating}}$ to be the set of violating and nonviolating paths in generalized matching M, respectively. Below we will only be interested in the endpoints of paths $p \in M$ and violating/nonviolating status of p. Thus, paths in M will essentially be treated as edges with a Boolean flag. For generalized matchings M_1, M_2 we define

$$M_1 \circ M_2 = \{(p, x, q) : (p, x) \in M_1, (x, q) \in M_2\}$$

where p, q are paths and x is a node. Clearly, $M_1 \circ M_2$ is also a generalized matching. For generalized matching M, vector $u \in \mathbb{R}^d$ and value $\sigma \in \mathbb{R}$ we define

$$\mathtt{Truncate}_\sigma(M; u) = M^{\mathtt{violating}} \cup \{(x, \ldots, y) \in M^{\mathtt{nonviolating}} : \langle y - x, u \rangle \geq \sigma\}$$

We will consider algorithms for constructing generalized matchings that have the following form: given vector $u \in \mathbb{R}^d$, sample vectors $(u_1, \ldots, u_k)$ according to some distribution that depends on u, and return $\mathtt{Matching}(u_1) \circ \ldots \circ \mathtt{Matching}(u_k)$. Any such algorithm specifies a *matching cover* as defined below.

Definition 3.5. A *generalized matching cover* (or just *matching cover*) is function $\mathcal{M}$ that maps vector $u \in \mathbb{R}^d$ to a distribution over generalized matchings. It is called *skew-symmetric* if sampling $M \sim \mathcal{M}(-u)$ and then reversing all paths in M produces the same distribution as sampling $M \sim \mathcal{M}(u)$. We define $\mathtt{size}(\mathcal{M}) = \frac{1}{n}\mathbb{E}_{M \sim \mathcal{M}(\mathcal{N})}[|M|]$ where $\mathcal{M}(\mathcal{N})$ denotes the distribution $u \sim \mathcal{N}, M \sim \mathcal{M}(u)$. We say that $\mathcal{M}$ is *σ-stretched* (resp. *L-long*) if $\langle y - x, u \rangle \geq \sigma$ (resp. $||y - x||^2 \leq L$) for any $u \in \mathbb{R}^d$ and

[2] Theorem 3.3 is formulated in [18] for the discrete cube, but the proof also works for the Gaussian distribution. For completeness, we reproduce the proof in the full version of this paper [13].

any nonviolating $(x, \dots, y) \in \mathsf{supp}(\mathcal{M}(u))$. $\mathcal{M}$ is *k-hop* if any path $p \in M \in \mathsf{supp}(\mathcal{M}(\mathcal{N}))$ has the form $p = (p_0, p_1, \dots, p_k)$ where $||p_i - p_{i-1}||^2 \le \Delta$ for all $i \in [k]$. We write $\mathcal{M} \subseteq \mathcal{M}'$ for (coupled) matching covers $\mathcal{M}, \mathcal{M}'$ if $M \subseteq M'$ for any u and $M \sim \mathcal{M}(u)$, $M' \sim \mathcal{M}(u')$.

If $\mathcal{M}$ is a matching cover then we define matching cover $\mathcal{M}_{[\sigma]}$ as follows: given vector u, sample $M \sim \mathcal{M}(u)$ and let $\mathtt{Truncate}_\sigma(M; u)$ be the output of $\mathcal{M}_{[\sigma]}(u)$. Clearly, $\mathcal{M}_{[\sigma]}$ is σ-stretched.

The following lemma shows how matching covers can be used. We say that direction $u \in \mathbb{R}^d$ is *regular* if $\langle y-x, u\rangle < \sqrt{6 \ln n} \cdot ||y-x||$ for all distinct $x, y \in V$.

Lemma 3.6. *Let $\mathcal{M}$ be a k-hop matching cover.*
(a) *$\mathcal{M}$ is $((k+1)\Delta)$-long.*
(b) *If $\mathcal{M}$ is $\sqrt{6(k+1)\Delta \ln n}$-stretched then $M^{\mathsf{nonviolating}} = \emptyset$ for any regular u and $M \in \mathsf{supp}(\mathcal{M}(u))$.*
(c) *There holds $\mathsf{Pr}_u[u \text{ is regular}] \ge 1 - \frac{1}{n}$.*

Proof. **(a)** By definitions, any nonviolating path $p = (p_0, p_1, \dots, p_k) \in M \in \mathsf{supp}(\mathcal{M}(\mathcal{N}))$ satisfies $||p_k - p_0||^2 \le \Delta + \sum_{i=1}^k ||p_i - p_{i-1}||^2 \le \Delta + \sum_{i=1}^k \Delta = (k+1)\Delta$.

(b) Suppose there exists nonviolating path $(x, \dots, y) \in M \in \mathsf{supp}(\mathcal{M}(u))$. Elements x, y must be distinct. Part (a) gives $||y - x|| \le \sqrt{(k+1)\Delta}$. Regularity of u thus implies that $\langle y - x, u\rangle < \sqrt{6 \ln n} \cdot \sqrt{(k+1)\Delta}$ - a contradiction.

(c) Consider distinct $x, y \in V$. The quantity $\langle y - x, u\rangle$ is normal with zero mean and variance $||y-x||^2$ under $u \sim \mathcal{N}$. Thus, $\mathsf{Pr}_u[\langle y - x, u\rangle \ge c||y-x||] = \mathsf{Pr}[\mathcal{N}(0, ||y-x||) \ge c||y-x||] = \mathsf{Pr}[\mathcal{N}(0,1) \ge c] \le e^{-c^2/2} = \frac{1}{n^3}$ for $c = \sqrt{6 \ln n}$. There are at most n^2 distinct pairs $x, y \in V$, so the union bound gives the claim. □

3.4 Chaining algorithms

By construction, `Matching` is a (deterministic) 1-hop σ-stretched skew-symmetric matching cover. Next, we discuss how to "chain together" matchings returned by $\mathtt{Matching}(\cdot)$ to obtain a matching cover with a large stretch, as required by Lemma 3.6(b).

Sherman's algorithm First, we review Sherman's algorithm [24]. In addition to vector u, it takes a sequence $b = (b_1, \dots, b_K) \in \{0,1\}^K$ as an input.

Algorithm 2: $\mathtt{SamplePaths}^b(u_1)$.

1 let k be the number of 1's in b and k' be the number of 0's in b, so that $k + k' = K$
2 sample $(u_1, \dots, u_k) \sim \mathcal{N}_\omega^k | u_1$ and $(u'_1, \dots, u'_{k'}) \sim \mathcal{N}_0^{k'}$ independently where $\omega = 1 - 1/k$
3 let $(\bar{u}_1, \dots, \bar{u}_K)$ be the sequence obtained by merging sequences $(u_1, \dots, u_k)$ and $(u'_1, \dots, u'_{k'})$ at positions specified by b in a natural way
4 return $\mathtt{Matching}(\bar{u}_1) \circ \dots \circ \mathtt{Matching}(\bar{u}_K)$

Theorem 3.7. *For any $\delta > 0$ and $\sigma > 0$ there exist positive constants c_1, c_2, c_3 with the following property. Suppose that* `Matching` *is a 1-hop σ-stretched skew-symmetric matching cover with* $\mathtt{size}(\mathtt{Matching}) \ge \delta$, *and $K\Delta \le c_1$. Then there exists vector $b \in \{0,1\}^K$ for which* $\mathtt{size}\left(\mathtt{SamplePaths}^b_{[c_2 K]}\right) \ge e^{-c_3 K^2}$.

In practice vector b is not known, however it can be sampled uniformly at random which decreases the expectation by a factor of 2^{-K}, which does not change the bound $e^{-\Theta(K^2)}$ in Theorem 3.7. Note that [24] does not use the notion of "violating paths", and accordingly Theorem 3.7 is formulated slightly differently in [24]. However, the proof in [24] can be easily adapted to yield Theorem 3.7. [3]

Robustness to deletions and parallelization Note that Theorem 3.7 can also be applied to any matching cover $\mathtt{Matching}' \subseteq \mathtt{Matching}$ satisfying $\mathtt{size}(\mathtt{Matching}') \ge \delta'$ for some positive constant $\delta' < \delta$. Thus, we can adversarially "knock out" some edges from `Matching` and still get useful bounds on the size of the output. This is a key proof technique in this paper; to our knowledge, it has not been exploited before. Our first use of this technique is for parallelization. We need to show that running Algorithm 2 multiple times independently produces many disjoint paths. Roughly speaking, our argument is as follows. Consider the i-th run, and assume that $|M_{\mathsf{prev}}|$ is small where M_{prev} is the union of paths computed in the first $i-1$ runs. Let us apply Theorem 3.7 to the matching cover obtained from `Matching` by knocking out edges incident to nodes in M_{prev}. It yields that the i-th run produces many paths that are node-disjoint from M_{prev}, as desired. We refer to Section 3.5 for further details.

New chaining algorithm Next, we discuss how a similar proof technique (combined with additional ideas) can be used to simplify Sherman's chaining algorithm. We will show that Theorem 3.7 still holds if we consider vectors b of the form $b = (1, \dots, 1)$. The algorithm thus becomes as follows.

Algorithm 3: $\mathtt{SamplePaths}^K(u_1)$.

1 sample $(u_1, \dots, u_K) \sim \mathcal{N}_\omega^K | u_1$ where $\omega = 1 - 1/K$
  // equivalently, sample $u_2 \sim_\omega u_1$, $u_3 \sim_\omega u_2$, …, $u_K \sim_\omega u_{K-1}$
2 return $\mathtt{Matching}(u_1) \circ \dots \circ \mathtt{Matching}(u_K)$

Theorem 3.8. *For any $\delta > 0$ and $\sigma > 0$ there exist positive constants c_1, c_2, c_3 with the following property. Suppose that*

[3] We believe that [24] has a numerical mistake. Namely, consider value $\delta \in (0,1)$ and weighted graph (V, E, w) with $n = |V|$ nodes and non-negative weights w such that $w(x, A) = w(A, x) \le 1$ for all $x \in V$ and $A \subseteq V$ (where $w(X, Y) = \sum_{(x,y) \in X \times Y} w(x, y)$). [24, proof of Lemma 4.8] essentially makes the following claim:

- *For any subset $B \subseteq V$ with $w(V, B) \ge \delta |B|$ there exists $A \subseteq V$ such that either (i) $|A| \ge \delta |B|$ and $w(x, B) \ge \delta^3$ for any $x \in A$, or (ii) $|A| \ge \frac{1}{\delta}|B|$ and $w(x, B) \ge \frac{\delta^2}{n}|B|$ for any $x \in A$.*

A counterexample can be constructed as follows. Assume that $\delta|B|$ and $\frac{1}{\delta^4}$ are integers. For $\delta|B| - 1$ nodes x set $w(x, B) = 1$, for $\frac{1}{\delta^4}$ nodes x set $w(x, B) = \delta^4$, and for remaining nodes set $w(x, B) = 0$. Then the claim above is false if $\delta|B| - 1 + \frac{1}{\delta^4} < \frac{1}{\delta}|B|$. The statement can be corrected as follows:

- *For any $\lambda \le \delta$ and any subset $B \subseteq V$ with $w(V, B) \ge \delta |B|$ there exists $A \subseteq V$ such that either (i) $|A| \ge \frac{\delta}{3}|B|$ and $w(x, B) \ge \lambda$ for any $x \in A$, or (ii) $|A| \ge \frac{\delta}{3\lambda}|B|$ and $w(x, B) \ge \frac{\delta}{3n}|B|$ for any $x \in A$. (If both conditions are false then $w(V, B) < \frac{\delta}{3}|B| \cdot 1 + \frac{\delta}{3\lambda}|B| \cdot \lambda + n \cdot \frac{\delta}{3n}|B| = \delta|B|$, which is impossible since $w(V, y) \ge \delta$ for each $y \in B$).*

Then the proof in [24] still works, but with different constants.

Matching is a 1-hop σ-stretched skew-symmetric matching cover with $\texttt{size}(\texttt{Matching}) \geq \delta$, $K\Delta \leq c_1$, *and* $K = 2^r$ *for some integer* r. *Then* $\texttt{size}\left(\texttt{SamplePaths}^K_{[c_2 K]}\right) \geq e^{-c_3 K^2}$.

We prove this theorem in Section 4. Our proof technique is very different from [24], and relies on two key ideas:

(1) We use induction on $k = 2^0, 2^1, 2^2, \ldots, K$ to show that $\texttt{SamplePaths}^k_{[\sigma_k]}$ has a sufficiently large size assuming that $\texttt{size}(\texttt{Matching}) \geq \delta_k$, for some sequences $\delta_1 < \delta_2 < \delta_4 < \ldots < \delta_K = \delta$ and $\sigma_1 < \sigma_2 < \sigma_4 < \ldots < \sigma_K = \Theta(K)$. To prove the claim for $2k$, we show that for each node x, function $\mu_x(u)$ is "sufficiently spread" where $\mu_x(u)$ is the expected out-degree of x in $M \sim \texttt{SamplePaths}^k_{[\sigma_k]}(u)$. In order to do this, we "knock out" edges (x, y) from $\texttt{Matching}(u)$ for a $\Theta(\delta_{2k} - \delta_k)$ fraction of vectors u with the largest value of $\mu_x(u)$, and then use the induction hypothesis for the smaller matching cover of size δ_k. We conclude that $\mu_x(u)$ is sufficiently large for $\Theta(\delta_{2k} - \delta_k)$ fraction of u's. We then use Theorem 3.3 and skew-symmetry to argue that chaining $\texttt{SamplePaths}^k_{[\sigma_k]}$ with itself gives a matching cover of large size.

(2) We work with "extended matching covers" instead of matching covers. These are functions $\mathcal{M}$ that take a **pair** of vectors $(u_1, u_k) \in \mathbb{R}^d \times \mathbb{R}^d$ as input, sample $(u_1, \ldots, u_k) \sim \mathcal{N}^k_\omega$ conditioned on fixed u_1, u_k, and return (a subset of) $\texttt{Matching}(u_1) \circ \ldots \circ \texttt{Matching}(u_k)$. This guarantees that if (x, y) is removed from $\texttt{Matching}(u_1)$ then x has no outgoing edge in $\mathcal{M}(u_1, u_k)$ (which is needed by argument above).

Our proof appears to be more compact than Sherman's proof, and also does not rely on case analysis. Accordingly, we believe that our technique should give smaller constants in the $O(\cdot)$ notation (although neither proof explicitly optimizes these constants).

Next, we discuss implications of Theorem 3.8. Below we denote $\lceil\!\lceil z \rceil\!\rceil = 2^{\lceil \log_2 z \rceil}$ to be the smallest $K = 2^r$, $r \in \mathbb{Z}_{\geq 0}$ satisfying $K \geq z$.

Corollary 3.9. *Suppose that* Matching *is a 1-hop σ-stretched skew-symmetric matching cover with* $\texttt{size}(\texttt{Matching}) \geq \delta$. *Define* $A = \frac{12}{c_2^2}$, $B = \frac{1}{2A\sqrt{c_3}}$ *and* $\varepsilon^{\max} = \frac{c_1}{2AB^2}$. *Suppose that* $\Delta = B\sqrt{\frac{\varepsilon}{\ln n}}$ *where* $\varepsilon \in (0, \varepsilon^{\max}]$ *and* $A\Delta \ln n \geq 1$. *If* $K = \lceil\!\lceil A\Delta \ln n \rceil\!\rceil$ *then*

$$\mathbb{E}_{M \sim \texttt{SamplePaths}^K(\mathcal{N})}[|M^{\texttt{violating}}|] \geq n^{1-\varepsilon} - 1$$

Proof. Note that $K \in [A\Delta \ln n, 2A\Delta \ln n]$, and $K\Delta \leq 2A\Delta^2 \ln n = 2AB^2\varepsilon \leq 2AB^2\varepsilon^{\max} = c_1$. Denote $\mathcal{M} = \texttt{SamplePaths}^K_{[c_2 K]}$. Theorem 3.8 gives $\texttt{size}(\mathcal{M}) \geq e^{-c_3 K^2} \geq e^{-c_3 (2A\Delta \ln n)^2} = n^{-\varepsilon}$. We also have $\frac{6(K+1)\Delta \ln n}{(c_2 K)^2} \leq \frac{12\Delta \ln n}{c_2^2 K} \leq \frac{12\Delta \ln n}{c_2^2 A\Delta \ln n} = 1$ and so the precondition of Lemma 3.6(b) holds for $\mathcal{M}$.

Let $x = \mathbb{E}_{u \sim \mathcal{N} | u \text{ is regular}, M \sim \mathcal{M}(u)}[|M|] = \mathbb{E}_{u \sim \mathcal{N} | u \text{ is regular}, M \sim \mathcal{M}(u)}[|M^{\texttt{violating}}|]$ (by Lemma 3.6(b)), $y = \mathbb{E}_{u \sim \mathcal{N} | u \text{ is not regular}, M \sim \mathcal{M}(u)}[|M|] \leq n$ and $p = \Pr_u[u \text{ is not regular}] \leq \frac{1}{n}$ (by Lemma 3.6(c)). We have $n^{1-\varepsilon} \leq \mathbb{E}_{M \sim \mathcal{M}(\mathcal{N})}[|M|] = (1-p)x + py \leq (1-p)x + \frac{1}{n} \cdot n$ and so $(1-p)x \geq n^{1-\varepsilon} - 1$. Therefore, $\mathbb{E}_{M \sim \texttt{SamplePaths}^K(\mathcal{N})}[|M^{\texttt{violating}}|] \geq (1-p)x \geq n^{1-\varepsilon} - 1$. □

3.5 Final algorithm

In the algorithm below we use the following notation: $V(p) \subseteq V$ is the set of nodes through which path p passes, and $V(M) = \bigcup_{p \in M} V(p)$. Furthermore, if p is violating then $p^{\texttt{violating}}$ is a subpath of p satisfying (8). Note that lines 1-3 compute sets of paths $\tilde{M}_1, \ldots, \tilde{M}_N$, which are then combined into a single set $M \subseteq \bigcup_i \tilde{M}_i$ using one of the two options. Option 1 will be mainly used for the analysis, while option 2 will be used for an efficient parallel implementation.

Algorithm 4: Computing violating paths.

1 **for** $i = 1, \ldots, N$ **do**
2 sample $u \sim \mathcal{N}$, call $M_i = \texttt{SamplePaths}^K(u)$
3 let $\tilde{M}_i = \{p^{\texttt{violating}} : p \in M_i^{\texttt{violating}}\}$
4 option 1: set $M := \emptyset$, then for $i = 1, \ldots, N$ update
$$M := M \cup \{p \in \tilde{M}_i : V(p) \cap V(M) = \emptyset\}$$
5 option 2: let M be a maximal set of paths in $M^* \stackrel{\text{def}}{=} \bigcup_{i=1}^N \tilde{M}_i$ s.t. $V(p) \cap V(q) = \emptyset$ for $p, q \in M$, $p \neq q$
6 return M

Theorem 3.10. *Suppose that* Matching *is a 1-hop σ-stretched skew-symmetric matching cover with* $\texttt{size}(\texttt{Matching}) \geq \delta$, *and let* $A, B, \varepsilon^{\max}$ *be the constants defined on Corollary 3.9 for value* $\delta/2$. *Suppose that* $\Delta = B\sqrt{\frac{\varepsilon}{\ln n}}$ *where* $\varepsilon \in (0, \varepsilon^{\max}]$ *and* $A\Delta \ln n \geq 1$. *If* $K = \lceil\!\lceil A\Delta \ln n \rceil\!\rceil$ *and* $N \geq \frac{\delta n^\varepsilon}{4K(1-n^{\varepsilon-1})}$ *then* $\mathbb{E}[|M|] \geq \frac{\delta n}{8K}$ *where* M *is the output of Algorithm 4 with option 1.*

Proof. Let $\mathcal{E}_i$ be the event that set M at the beginning of iteration i satisfies $|M| \leq a := \frac{\delta n}{4K}$, and let $\gamma_i = \Pr[\mathcal{E}_i]$. Let x_i be the expected number of paths that have been added to M at iteration i, so that $\mathbb{E}[|M|] = x_1 + \ldots + x_N$ for the final set M. Let y_i be the expected number of paths that have been added to M at iteration i conditioned on event $\mathcal{E}_i$. Clearly, we have $x_i \geq \gamma_i y_i \geq \gamma y_i$ where $\gamma := \gamma_N$.

Next, we bound y_i. Let M be the set at the beginning of iteration i, and suppose that $\mathcal{E}_i$ holds, i.e. $|M| \leq \frac{\delta n}{4K}$. Denote $U = V(M)$, then $|U| \leq K|M| \leq \frac{\delta n}{4}$. Let $\texttt{Matching}' \subseteq \texttt{Matching}$ be the (skew-symmetric) matching cover obtained by removing from $\texttt{Matching}(u)$ edges (x, y) and (y, x) with $x \in U$ (for all $u \in \mathbb{R}^d$). Clearly, we have $\texttt{size}(\texttt{Matching}') \geq \delta - \frac{1}{2}\delta = \frac{1}{2}\delta$. Let $\texttt{SamplePaths}'^K$ be the matching cover given by Algorithm 3 where Matching is replaced by Matching′. Clearly, we have $\texttt{SamplePaths}'^K \subseteq \texttt{SamplePaths}^K$. By Corollary 3.9 applied to Matching′, $\texttt{SamplePaths}'^K(\mathcal{N})$ produces at least $n^{1-\varepsilon} - 1$ violating paths in expectation. By construction, all these paths p satisfy $V(M) \cap V(p) = \emptyset$, and therefore $y_i \geq n^{1-\varepsilon} - 1$.

We showed that $\mathbb{E}[|M|] \geq \sum_{i=1}^N \gamma(n^{1-\varepsilon} - 1) = \gamma b$ for the final set M where $b := N(n^{1-\varepsilon} - 1)$. We also have $\mathbb{E}[|M|] \geq (1-\gamma)a$, and hence $\mathbb{E}[|M|] \geq \min_{\gamma \in [0,1]} \max\{\gamma b, (1-\gamma)a\} = \frac{ab}{a+b}$ (the minimum is attained at $\gamma = a/(a+b)$). By assumption, we have $b \geq a$, and so $\mathbb{E}[|M|] \geq \frac{1}{2}a$. □

Next, we analyze Algorithm 4 with option 2. Recall that "parallel runtime" is the maximum runtime over all processors.

Lemma 3.11. *(a) Let M and M' be the outputs of Algorithm 4 with options 1 and 2, respectively (for a given run of the loop in lines 1-3). Then $|M'| \geq |M|/K^3$.*
(b) Algorithm 4 with option 2 can be implemented on N processors with parallel runtime $O(KT_{\texttt{maxflow}} + n(Kd + K\log^2 n + \log N)))$ (in any version of PRAM).

Proof. **(a)** By the construction of M, each node $v \in V(M^*)$ is contained in at most K paths $p \in M$ (all of them belong to some $\tilde{M}_i$ for fixed i). Therefore, $|V(M^*)| \geq |M|/K$. We also have $|V(M')| \geq |V(M^*)|/K$ and $|M'| \geq |V(M')|/K$, since $|V(p)| \leq K$ for any $p \in M^*$. Putting these inequalities together gives the claim.
(b) Clearly, sets $\tilde{M}_i$ for $i \in [N]$ can be computed in parallel on N processors in time $O(K(T_{\texttt{maxflow}} + nd + n\log^2 n)))$ per processor (since computing dot products $\langle x, u\rangle$ in Algorithm 1 takes time $O(Knd)$, and flow decompositions take time $O(Km\log n) = O(Kn\log^2 n)$). We can assume that after computing $\tilde{M}_i$, processor i computes a maximal set of paths $\tilde{M}'_i \subseteq \tilde{M}_i$ that are pairwise node-disjoint, and updates $\tilde{M}_i := \tilde{M}'_i$. Clearly, this step does not affect the output of line 5.

It remains to discuss how to implement line 5 (computing a maximal set of paths M in M^* which are pairwise node-disjoint). For $N = 2$ this can be easily done in $O(n)$ time: processor 1 sends $\tilde{M}_1$ to processor 2, and processor 2 computes the answer. The general case can be reduced to the case above using a divide-and-conquere strategy with a computation tree which is a binary tree whose leaves are the N processors. The depth of this tree is $O(\log N)$, and hence the parallel runtime of this procedure is $O(n\log N)$. □

Note in Algorithm 4 step 3 can be run in parallel on N processors; after all of them finish, we can run the rest of algorithm on a single machine. By putting everything together, we obtain

Theorem 3.12. *There exists an algorithm for* Balanced Separator *that given $\varepsilon \in [\Theta(1/\log n), \Theta(1)]$, produces $O(\sqrt{(\log n)/\varepsilon})$-pseudoapproximation w.h.p.. Its expected parallel runtime is $O((\log^{O(1)} n)T_{\texttt{maxflow}})$ on $O(n^\varepsilon)$ processors (in any version of PRAM).*

Proof. Let δ, σ be as in Lemma 3.4. Set parameters as in Theorem 3.10, and require additionally that $\varepsilon \leq \frac{1}{2}$. Condition $A\Delta \ln n \geq 1$ means that this can be done for $\varepsilon \in [\Theta(1/\log n), \Theta(1)]$. By Theorem 3.10 and Lemma 3.11, the output of Algorithm 4 satisfies $\mathbb{E}[|M|] \geq \frac{\delta n}{8K^{1+h}}$ where $h = 0$ if option 1 is used, and $h = 3$ if option 2 is used.

To implement the oracle, run Algorithm 4 until either procedure `Matching`$(\cdot)$ (Alg. 1) terminates at lines 5 or 7, or until we find a set M of violating paths with $|M| \geq \frac{\delta n}{16K^{1+h}}$. In the latter case use Lemma 3.1 to set the variables. Since we always have $|M| \leq n$, the expected number of runs will be $O(n/\frac{n}{K^{1+h}}) = O(K^{1+h})$.

If the oracle terminates at line 5 of Alg. 1, then it returns a c'-balanced cut of cost at most $\frac{6\alpha}{\Delta} = O\left(\alpha\sqrt{\frac{\log n}{\varepsilon}}\right)$. Otherwise it returns valid variables f_p, F. If the oracle terminates at line 7 then its width is $\rho = O(\frac{\alpha}{n} + \pi) = O(\frac{\alpha}{n\Delta})$. Now suppose that the oracle finds set M of violating paths with $|M| \geq \frac{\delta n}{16K^{1+h}}$. Clearly, degrees π_F, π_D in Lemma 3.1 satisfy $\pi_F = O(K)$ and $\pi_D = O(1)$, so the oracle's width in this case is $\rho = O(\frac{\alpha}{n} + \frac{\alpha K^{2+h}}{n\Delta})$. In both cases we have $\rho = O(\frac{\alpha K^{2+h}}{n\Delta})$. Thus, the number of calls to `Oracle` for a fixed value of α is $T = O(\frac{\rho^2 n^2 \log n}{\alpha^2}) = O(\frac{K^{4+2h}\log n}{\Delta^2})$.

Next, we bound the complexity of computing approximations $\tilde{v}_1, \ldots, \tilde{v}_{n_{\texttt{orig}}}$ for fixed α as described in Theorem A.2 in Section A.1. (Here we assume familiarity with Appendix A). We have $\tau = \Theta(1)$ and $\gamma = \Theta\{\min\{\frac{\alpha}{n\pi}, \frac{\Delta}{K}\}) = \Theta(\frac{\Delta}{K})$, thus we need to use dimension $d = \Theta(\frac{\log n}{\gamma^2}) = \Theta(\frac{K^2\log n}{\Delta^2})$. We need to compute Tkd matrix-vector products of form $A \cdot u$ where $k = O(\max\{(\frac{\rho n\log n}{\alpha})^2, \log n\}) = O(\frac{K^4\log^2 n}{\Delta^2})$. Each matrix A has the form $\sum_{r=1}^{O(T)} N^{(r)}$, and each $N^{(r)}$ can be represented as a sum of $O(K)$ "easy" matrices (e.g. corresponding to matchings) for which the multiplication with a vector takes $O(n)$ time. To summarize, the overall complexity of computing approximations $\tilde{v}_1, \ldots, \tilde{v}_{n_{\texttt{orig}}}$ is $Tkd \cdot TKn = O(\frac{K^{15+4h}\log^5 n}{\Delta^8} n)$, which is $O(n\log^{O(1)} n)$ since $\Delta = \Theta\left(\sqrt{\frac{\varepsilon}{\log n}}\right) \in \left[\Theta(\frac{1}{\log n}), \Theta(\frac{1}{\sqrt{\log n}})\right]$ and $K = \Theta(\Delta\log n) = O(\sqrt{\log n})$. These computations can be done on a single processor; their runtime is subsumed by the claimed $O((\log^{O(1)} n)T_{\texttt{maxflow}})$ bound.

We saw that $d = O(\log^{O(1)} n)$. From Lemma 3.11 we can now conclude that the algorithm's expected parallel runtime is $T \cdot O(KT_{\texttt{maxflow}} + n(Kd + K\log^2 n + \log N))) = O((\log^{O(1)} n)T_{\texttt{maxflow}})$ on $N = \Theta(n^\varepsilon/K)$ processors, assuming that option 2 is used. Note that the output of the overall algorithm is correct w.h.p. since vectors $\tilde{v}_1, \ldots, \tilde{v}_{n_{\texttt{orig}}}$ approximate original vectors only w.h.p. (see Theorem A.2). □

4 PROOF OF THEOREM 3.8

We will need the following definition.

Definition 4.1. A *k-hop extended matching cover* is function $\mathcal{M}$ that maps vectors $u, u' \in \mathbb{R}^d$ to a distribution over generalized matchings. It is *skew-symmetric* if sampling $M \sim \mathcal{M}(-u', -u)$ and then reversing all paths in M produces the same distribution as sampling $M \sim \mathcal{M}(u, u')$. We define $\texttt{size}_\omega(\mathcal{M}) = \frac{1}{n}\mathbb{E}_{M\sim\mathcal{M}(\mathcal{N}_\omega)}[|M|]$ where $\mathcal{M}(\mathcal{N}_\omega)$ denotes the distribution $(u, u') \sim \mathcal{N}_\omega, M \sim \mathcal{M}(u, u')$. We say that $\mathcal{M}$ is *σ-stretched* (resp. *L-long*) if $\min\{\langle y - x, u\rangle, \langle y - x, u'\rangle\} \geq \sigma$ (resp. $||y - x||^2 \leq L$) for any $u, u' \in \mathbb{R}^d$ and any nonviolating $(x, \ldots, y) \in \texttt{supp}(\mathcal{M}(u, u'))$. We write $\mathcal{M} \subseteq \tilde{\mathcal{M}}$ for (coupled) extended matching covers $\mathcal{M}, \tilde{\mathcal{M}}$ if $M \subseteq \tilde{M}$ for any u, u' and $M \sim \mathcal{M}(u, u')$, $\tilde{M} \sim \tilde{\mathcal{M}}(u, u')$.

For an extended matching cover $\mathcal{M}$ we can define matching cover $\mathcal{M}_\omega$ as follows: given vector u, sample $u' \sim_\omega u$, $M \sim \mathcal{M}(u, u')$ and let M be the output of $\mathcal{M}(u)$. Clearly, if $\mathcal{M}$ is σ-stretched then $\mathcal{M}_\omega$ is also σ-stretched, and $\texttt{size}_\omega(\mathcal{M}) = \texttt{size}(\mathcal{M}_\omega)$.

We now proceed with the proof of Theorem 3.8. Define $\mathcal{K} = \{2^0, 2^1, 2^2, \ldots, K\}$, and let $\dot{\mathcal{K}} = \mathcal{K} - \{K\}$. (Recall that K has the form $K = 2^r$ for some integer r). Let us choose positive numbers β_k for $k \in \dot{\mathcal{K}}$ (to be specified later), and define numbers $\{\sigma_k\}_{k\in\mathcal{K}}$ via the

following recursion:

$$\sigma_1 = \sigma \tag{11a}$$
$$\sigma_{2k} = (1+\omega^k)\sigma_k - \beta_k \qquad \forall k \in \dot{\mathcal{K}} \tag{11b}$$

For a matching cover $\mathcal{M} \subseteq \texttt{Matching}$ and integer $k \in \mathcal{K}$ we define extended matching cover $\mathcal{M}^k$ as follows:

- given vectors (u_1, u_k), sample $(u_1, u_2, \ldots, u_k) \sim \mathcal{N}_\omega^k|(u_1, u_k)$, compute $M = \mathcal{M}(u_1) \circ \ldots \circ \mathcal{M}(u_k)$ and let $\texttt{Truncate}_{\sigma_k}(M; u_1) \cap \texttt{Truncate}_{\sigma_k}(M; u_k)$ be the output of $\mathcal{M}^k(u_1, u_k)$.

By definition, $\mathcal{M}^k$ is σ_k-stretched. It can be seen that $(\mathcal{M}^K)_{\omega^{K-1}} \subseteq \texttt{SamplePaths}^K$ for any $\mathcal{M} \subseteq \texttt{Matching}$. Our goal will be to analyze $\texttt{size}_{\omega^{K-1}}(\mathcal{M}^K) = \texttt{size}((\mathcal{M}^K)_{\omega^{k-1}})$ for $\mathcal{M} = \texttt{Matching}$.

Theorem 4.2. *Choose an increasing sequence of numbers $\{\delta_k\}_{k\in\mathcal{K}}$ in $(0,1)$, and define sequence $\{\lambda_k\}_{k\in\mathcal{K}}$ via the following recursions:*

$$\lambda_1 = \delta_1 \tag{12a}$$
$$\lambda_{2k} = \theta_{2k}\lambda_k^2, \quad \theta_{2k} = \tfrac{1}{2}\left(\tfrac{1}{2}(\delta_{2k} - \delta_k)\right)^{2/(1-\omega)} \qquad \forall k \in \dot{\mathcal{K}} \tag{12b}$$

Suppose that

$$\exp\left(-\frac{\beta_k^2}{2(k+1)(1-\omega^k)\Delta}\right) \le \tfrac{1}{2}\lambda_{2k} \qquad \forall k \in \dot{\mathcal{K}} \tag{13}$$

Then for any $k \in \mathcal{K}$ and any skew-symmetric matching cover $\mathcal{M} \subseteq \texttt{Matching}$ with $\texttt{size}(\mathcal{M}) \ge \delta_k$ there holds $\texttt{size}_{\omega^{k-1}}(\mathcal{M}^k) \ge \lambda_k$.

By plugging appropriate sequences $\{\delta_k\}$ and $\{\beta_k\}$ we can derive Theorem 3.8 as follows. (The proof of Lemma 4.3 is given in the full version of this paper [13]).

Lemma 4.3. *Define $\delta_k = (1 - \frac{1}{2k})\delta$ and $\beta_k = 2\phi\sigma k\sqrt{1-\omega^k}$ where $\phi = \frac{1}{\sigma}\left(K\Delta \cdot 5\ln\frac{16}{\delta}\right)^{1/2}$. Then (13) holds, and*

$$\lambda_k \ge \left(\frac{16}{\delta}\right)^{-2Kk} \tag{14}$$

Furthermore, if $\phi \le \frac{1}{8}$ (or equivalently $K\Delta < \frac{\sigma^2}{320\ln\frac{16}{\delta}}$) then

$$\sigma_K \ge \tfrac{1}{16}\,\sigma\,K \tag{15}$$

4.1 Proof of Theorem 4.2

To prove the theorem, we use induction on $k \in \mathcal{K}$. For $k = 1$ the claim is trivial. In the remainder of this section we assume that the claim holds for $k \in \dot{\mathcal{K}}$, and prove it for $2k$ and skew-symmetric matching cover $\mathcal{M} \subseteq \texttt{Matching}$ with $\texttt{size}(\mathcal{M}) \ge \delta_{2k}$. Clearly, the skew-symmetry of $\mathcal{M}$ implies that $\mathcal{M}^k$ is also skew-symmetric. We introduce the following notation; letter x below always denotes a node in S.

- Let $\mu_x(u, u')$ be the expected out-degree of x in $M \sim \mathcal{M}^k(u, u')$.
- Let $v_x(u) = \mathbb{E}_{u'\sim_\rho u}[\mu_x(u, u')]$ where $\rho := \omega^{k-1}$.
- Let $\mathcal{A}_x$ be a subset of $\mathbb{R}^d$ of Gaussian measure $\delta := \frac{1}{2}(\delta_{2k} - \delta_k)$ containing vectors u with the largest value of $v_x(u)$, i.e. such that $\gamma_x := \inf\{v_x(u) : u \in \mathcal{A}_x\} \ge \sup\{v_x(u) : u \in \mathbb{R}^d - \mathcal{A}_x\}$.
- Let $\dot{\mathcal{M}} \subseteq \mathcal{M}$ be the matching cover obtained from $\mathcal{M}$ by removing edges (x, y) from $\mathcal{M}(u)$ and edges (y, x) from $\mathcal{M}(-u)$ for each $u \in \mathcal{A}_x$. Clearly, $\dot{\mathcal{M}}$ is a skew-symmetric matching cover with $\texttt{size}(\dot{\mathcal{M}}) \ge \delta_{2k} - 2\delta = \delta_k$.
- Let $\dot{\mu}_x(u, u')$ be the expected out-degree of x in $M \sim \dot{\mathcal{M}}^k(u, u')$.
- Let $\dot{v}_x(u) = \mathbb{E}_{u'\sim_\rho u}[\dot{\mu}_x(u, u')]$, and let $\lambda_x = ||\dot{v}||_1$. Note that $\frac{1}{n}\sum_{x\in V}\lambda_x = \texttt{size}_\rho(\dot{\mathcal{M}}^k) \ge \lambda_k$ where the last inequality is by the induction hypothesis.

Lemma 4.4. *There holds $\gamma_x \ge \lambda_x$.*

Proof. By the definition of $\dot{\mathcal{M}}$, we have $\dot{\mu}_x(u, u') = 0$ for any $u \in \mathcal{A}_x$ and $u' \in \mathbb{R}^d$. This implies that $\dot{v}_x(u) = 0$ for any $u \in \mathcal{A}_x$. We have $\dot{\mathcal{M}} \subseteq \mathcal{M}$ and thus $\dot{\mathcal{M}}^k \subseteq \mathcal{M}^k$. This implies that $\dot{v}_x(u) \le v_x(u) \le \gamma_x$ for any $u \in \mathbb{R}^d - \mathcal{A}_x$. We can now conclude that $\lambda_x = ||\dot{v}_x||_1 \le \gamma_x \cdot 1$.

□

Let $\mathcal{H}$ be an extended matching cover where $\mathcal{H}(u_1', u_2')$ is defined as follows:

(∗) sample $(u_1', u_1, u_2, u_2') \sim \mathcal{N}_{\rho,\omega,\rho}|(u_1', u_2')$, sample $M_1 \sim \mathcal{M}^k(u_1', u_1)$, sample $M_2 \sim \mathcal{M}^k(u_2, u_2')$, compute $M = M_1 \circ M_2$, output $M^{\text{good}} = \texttt{Truncate}_{\sigma_{2k}}(M; u_1') \cap \texttt{Truncate}_{\sigma_{2k}}(M; u_2')$.

It can be seen that $\mathcal{H} \subseteq \mathcal{M}^{2k}$ under a natural coupling. Consider the following process: sample $(u_1', u_2') \sim \mathcal{N}_{\rho\omega\rho}$ and then run procedure (∗). By definitions, we have $\texttt{size}_{\rho\omega\rho}(\mathcal{H}) = \frac{1}{n}\mathbb{E}[|M^{\text{good}}|]$. Let M_x^{good} be the set of paths in M that go through node x, and denote $\tau_x = \mathbb{E}[|M_x^{\text{good}}|]$. From these definitions we get that $\texttt{size}_{\rho\omega\rho}(\mathcal{H}) = \frac{1}{n}\sum_{x\in V}\tau_x$.

Lemma 4.5. *There holds $\tau_x \ge \delta^{2/(1-\omega)}\lambda_x^2 - 2\varepsilon$ where $\varepsilon := \exp\left(-\frac{\beta_k^2}{2(k+1)(1-\omega\rho)\Delta}\right)$.*

Proof. Define random variable p_1 and p_2 as follows:

- if x has an incoming path in M_1 then let p_1 be this path, otherwise let $p_1 = \perp$;
- if x has an outgoing path in M_1 then let p_2 be this path, otherwise let $p_2 = \perp$.

Let $\mathcal{E}^{\text{good}}(p_1, p_2) = [p_1 \ne \perp \wedge p_2 \ne \perp]$. Let $\mathcal{E}_1^{\text{bad}}(p_1, u_2')$ be the event that $p_1 = (y_1, \ldots, x) \ne \perp$, p_1 is nonviolating and $\langle x - y_1, u_2'\rangle < \sigma := \omega\rho\sigma_k - \beta_k$. Similarly, let $\mathcal{E}_2^{\text{bad}}(p_2, u_1')$ be the event that $p_2 = (x, \ldots, y_2) \ne \perp$, p_2 is nonviolating and $\langle y_2 - x, u_1'\rangle < \sigma$. Note, if $[\mathcal{E}^{\text{good}}(p_1, p_2) \wedge \neg\mathcal{E}_1^{\text{bad}}(p_1, u_2')]$ holds then $p_1 \circ p_2 = (y_1, \ldots, x, \ldots, y_2)$ is either violating or satisfies $\langle y_2 - y_1, u_2'\rangle \ge \sigma_k + \sigma = \sigma_{2k}$ (and similarly for $\mathcal{E}_2^{\text{bad}}(p_2, u_1')$). Therefore, $\tau_x \ge \texttt{Pr}[\mathcal{E}^{\text{good}}(p_1, p_2)] - \texttt{Pr}[\mathcal{E}_1^{\text{bad}}(p_1, u_2')] - \texttt{Pr}[\mathcal{E}_2^{\text{bad}}(p_2, u_1')]$.

Clearly, vectors (u_1', u_1, u_2, u_2') are distributed according to $\mathcal{N}_{\rho,\omega,\rho}$. Equivalently, they are obtained by the following process: sample $(u_1, u_2) \sim \mathcal{N}_\omega$, sample $u_1' \sim_\rho u_1$, sample $u_2' \sim_\rho u_2$. Define $\mathcal{B}_x = \{u : -u \in \mathcal{A}_x\}$. By Theorem 3.3, we will have $(u_1, u_2) \in \mathcal{A}_x \times \mathcal{B}_x$ with probability at least $\delta^{2/(1-\omega)}$. Conditioned on the latter event, we have $\texttt{Pr}[p_1 \ne \perp] \ge \gamma_x$ and $\texttt{Pr}[p_2 \ne \perp] \ge \gamma_x$ (independently), where the claim for p_1 follows from the

skew-symmetry of $\mathcal{M}^k$. This implies that $\Pr[\mathcal{E}^{\text{good}}(p_1,p_2)] \geq \delta^{2/(1-\omega)}\gamma_x^2 \geq \delta^{2/(1-\omega)}\lambda_x^2$.

We claim that $\Pr[\mathcal{E}_1^{\text{bad}}(p_1,u_2')] \leq \varepsilon$. Indeed, it suffices to prove that for fixed u_1', u_1, p_1 we have $\Pr[\mathcal{E}_1^{\text{bad}}(p_1,u_2')] \leq \varepsilon$ under $u_2 \sim_\omega u_1$, $u_2' \sim_\rho u_2$ (or equivalently under $u_2' \sim_{\omega\rho} u_1$). Assume that $p_1 = (y_1,\dots,x) \neq \perp$ is nonviolating (otherwise the desired probability is zero and the claim holds). Since $\mathcal{M}^k$ is σ_k-stretched, we have $\langle x-y_1, u_1\rangle \geq \sigma_k$. We also have $r := ||x-y_1|| \leq \sqrt{(k+1)\Delta}$ by Lemma 3.6(a). The quantity $\langle x-y_1, u_2'\rangle$ is normal with mean $\omega\rho\langle x-y_1,u_1\rangle \geq \omega\rho\sigma_k$ and variance $(1-(\omega\rho)^2)r^2$ (since e.g. we can assume that $x-y_1 = (r,0,\dots,0)$ by rotational symmetry, and then use the definitions of correlated Gaussians for 1-dimensional case). Therefore, $\Pr_{u_2'\sim_{\omega\rho}u_1}\left[\langle x-y_1,u_2'\rangle < \sigma\right]$

$$\leq \Pr\left[\mathcal{N}(\omega\rho\sigma_k, (1-(\omega\rho)^2)^{1/2}r) < \omega\rho\sigma_k - \beta_k\right]$$

$$= \Pr\left[\mathcal{N}(0,1) < -\frac{\beta_k}{(1-(\omega\rho)^2)^{1/2}r}\right] < \exp\left(-\frac{\beta_k^2}{2(1-(\omega\rho)^2)r^2}\right) \leq \varepsilon$$

In a similar way we prove that $\Pr[\mathcal{E}_2^{\text{bad}}(p_2,u_1')] \leq \varepsilon$. The lemma follows.

□

We showed that

$$\texttt{size}_{\rho\omega\rho}(\mathcal{H}) \geq \left(\frac{1}{n}\sum_{x\in V}\delta^{2/(1-\omega)}\lambda_x^2\right) - 2\varepsilon$$

Let us minimize the bound on the RHS under constraint $\frac{1}{n}\sum_{x\in V}\lambda_x \geq \lambda_k$. Clearly, the minimum is obtained when $\lambda_x = \lambda_k$ for all $x \in V$, in which case the bound becomes

$$\texttt{size}_{\rho\omega\rho}(\mathcal{H}) \geq \delta^{2/(1-\omega)}\lambda_k^2 - 2\varepsilon \geq \lambda_{2k}$$

where we used (12b) and (13).

A MATRIX MULTIPLICATIVE WEIGHTS (MW) ALGORITHM

It this section we review the method of Arora and Kale [4] for the checking the feasibility of system (2′) consisting of constraints (2a′) and (2b)-(2e). The algorithm is given below.

Algorithm 5: MW algorithm.

1 **for** $t = 1,2,\dots,T$ **do**

2 compute

$$W^{(t)} = \exp\left(\eta\sum_{r=1}^{t-1}N^{(r)}\right), \quad X^{(t)} = n\cdot\frac{W^{(t)}}{\texttt{Tr}(W^{(t)})}$$

3 either output `Fail` or find "feedback matrix" $N^{(t)}$ of the form

$$N^{(t)} = \texttt{diag}(y) + \sum_p f_p T_p + \sum_S z_S K_S - F$$

where $f_p \geq 0$, $z_S \geq 0$, $\sum_i y_i + an^2\sum_S z_S \geq \alpha$, F is a symmetric matrix with $F \preceq C$, and $N^{(t)} \bullet X^{(t)} \leq 0$.

It can be checked that $N^{(t)} \bullet X \geq 0$ for all feasible X satisfying (2′) (see [4]). Thus, matrix $N^{(t)}$ can be viewed as a cutting plane that certifies that $X^{(t)}$ is infeasible (or lies on the boundary of the feasible region).

The procedure at line 3 is called `Oracle`, and the maximum possible spectral norm $||N^{(t)}||$ of the feedback matrix is called the *width* of the oracle. This width will be denoted as ρ. [4]

Theorem A.1 ([4]). *Set $\eta = \frac{\epsilon}{2\rho^2 n}$ and $T = \lceil\frac{4\rho^2 n^2 \ln n}{\epsilon^2}\rceil$. If Algorithm 5 does not fail during the first T iterations then the optimum value of SDP (2) is at least $\alpha - \epsilon$.*

The oracle used in [4] has the following property: if it fails then it returns a cut which is $\frac{c}{512}$-balanced and has value at most $\kappa\alpha$, where value κ depends on the implementation. One of the implementations achieves $\kappa = O(\sqrt{\log n})$ and has width $\rho = \tilde{O}(\frac{\alpha}{n})$. (The runtime of this oracle will be discussed later). Setting $\epsilon = \alpha/2$ yields an $O(\kappa) = O(\sqrt{\log n})$ approximation algorithm that makes $\tilde{O}(1)$ calls to the oracle.

A.1 Gram decomposition and matrix exponentiation

Consider matrix $X = X^{(t)}$ computed at the t-th step of Algorithm 5. It can be seen that X is positive semidefinite, so we can consider its Gram decomposition: $X = V^TV$. Let $v_1,\dots,v_n$ be the columns of V; clearly, they uniquely define X. Note that computing $v_1,\dots,v_n$ requires matrix exponentiation, which is a tricky operation because of accuracy issues. Furthermore, even storing these vectors requires $\Theta(n^2)$ space and thus $\Omega(n^2)$ time, which is too slow for our purposes. To address these issues, Arora and Kale compute approximations $\tilde{v}_1,\dots,\tilde{v}_n$ to these vectors using the following result.

Theorem A.2 ([4, Lemma 7.2]). *For any constant $c > 0$ there exists an algorithm that does the following: given values $\gamma \in (0,\frac{1}{2})$, $\lambda > 0$, $\tau = O(n^{3/2})$ and matrix $A \in \mathbb{R}^{n\times n}$ of spectral norm $||A|| \leq \lambda$, it computes matrix $\tilde{V} \in \mathbb{R}^{d\times n}$ with column vectors $\tilde{v}_1,\dots,\tilde{v}_n$ of dimension $d = O(\frac{\log n}{\gamma^2})$ such that matrix $\tilde{X} = \tilde{V}^T\tilde{V}$ has trace n, and with probability at least $1-n^{-c}$, one has*

$$|\,||\tilde{v}_i||^2 - ||v_i||^2\,| \leq \gamma(||\tilde{v}_i||^2 + \tau) \qquad \forall i \qquad (16a)$$

$$|\,||\tilde{v}_i - \tilde{v}_j||^2 - ||v_i - v_j||^2\,| \leq \gamma(||\tilde{v}_i - \tilde{v}_j||^2 + \tau) \qquad \forall i,j \qquad (16b)$$

where $v_1,\dots,v_n$ are the columns of a Gram decomposition of $X = n\cdot\frac{\exp(A)}{\texttt{Tr}(\exp(A))}$. The complexity of this algorithm equals the complexity of computing kd matrix-vector products of the form $A\cdot u$, $u \in \mathbb{R}^n$, where $k = O(\max\{\lambda^2, \log\frac{n^{5/2}}{\tau}\})$.

Note that matrices A used in Algorithm 5 have norm at most $\eta\rho T$. Therefore, we can set $\lambda = \Theta(\frac{\rho n\log n}{\alpha})$ when applying Theorem A.2 to Algorithm 5. Parameters γ and τ will be specified later.

From now on we make the following assumption.

Assumption 1. *We have (unobserved) matrix $V \in \mathbb{R}^{n\times n}$ and (observed) matrix $\tilde{V} \in \mathbb{R}^{d\times n}$ with $X = V^TV$, $\tilde{X} = \tilde{V}^T\tilde{V}$ and $\texttt{Tr}(X) = \texttt{Tr}(\tilde{X}) = n$ satisfying conditions (16) where $v_1,\dots,v_n$ are the columns of V and $\tilde{v}_1,\dots,\tilde{v}_n$ are the columns of $\tilde{V}$.*

[4] Note that [4] formulated the algorithm in terms of the "loss matrix" $M^{(t)} = -\frac{1}{\rho}N^{(t)}$ that satisfies $||M^{(t)}|| \leq 1$. Namely, it used the update $W^{(t)} = \exp\left(-\bar{\eta}\sum_{r=1}^{t-1}M^{(r)}\right)$ with $\bar{\eta} = \rho\eta$, and set $\bar{\eta} = \frac{\epsilon}{2\rho n}$ in their Theorems 4.4 and 4.6.

A.2 Oracle implementation

Let us denote $S = \{i \in V : ||\tilde{v}_i||^2 \le 2\}$. We have $\sum_{i\in V} ||\tilde{v}_i||^2 = \mathtt{Tr}(\tilde{V}^T\tilde{V}) = n$ and thus $|S| \ge n/2$. First, one can eliminate an easy case.

Proposition A.3. *Suppose that $K_S \bullet \tilde{X} < \frac{an^2}{4}$. Then setting $y_i = -\frac{\alpha}{n}$ for all $i \in V$, $z_S = \frac{2\alpha}{an^2}$, $z_{S'} = 0$ for all $S' \ne S$, and $F = 0$ gives a valid output of the oracle with width $\rho = O(\frac{\alpha}{n})$ assuming that parameters τ, γ in Theorem A.2 satisfy $\gamma \le \frac{1}{2}$ and $\tau \le \frac{a}{2}$.*

Proof. Denote $z_{ij} = ||v_i - v_j||^2$ and $\tilde{z}_{ij} = ||\tilde{v}_i - \tilde{v}_j||^2$. We know that $\tilde{Z} := \sum_{ij} \tilde{z}_{ij} = K_S \bullet \tilde{X} < \frac{an^2}{4}$ where the sum is over $i, j \in S$. Also, $|z_{ij} - \tilde{z}_{ij}| \le \gamma(\tilde{z}_{ij} + \tau)$ for all i, j. This implies that

$$K_S \bullet X = \sum_{ij} z_{ij} < \tilde{Z} + \gamma\tilde{Z} + \frac{n^2}{2}\gamma\tau \le (1+\gamma)\frac{an^2}{4} + \frac{n^2}{2}\gamma\tau \le \frac{an^2}{2}$$

Note that $N^{(t)} = -\frac{\alpha}{n}I + \frac{2\alpha}{an^2}K_S$. We have $\sum_i y_i + an^2 \sum_S z_S = n \cdot (-\frac{\alpha}{n}) + an^2 \cdot \frac{2\alpha}{an^2} = \alpha$ and $N^{(t)} \bullet X = -\frac{\alpha}{n}I \bullet X + \frac{2\alpha}{an^2}(K_S \bullet X) \le -\frac{\alpha}{n} \cdot n + \frac{2\alpha}{an^2} \cdot \frac{an^2}{2} = 0$, as desired. Also, $||N^{(t)}|| \le || - \frac{\alpha}{n}I|| + ||\frac{2\alpha}{an^2}K_S|| = O(\frac{\alpha}{n})$. □

Now suppose that $\sum_{i,j\in S} ||\tilde{v}_i - \tilde{v}_j||^2 = K_S \bullet \tilde{X} \ge \frac{an^2}{4}$. We will set $N^{(t)}_{ij} = 0$ for all $(i, j) \notin S \times S$. When describing how to set the remaining entries of $N^{(t)}$, it will be convenient to treat $N^{(t)}, X, \tilde{X}$ etc. as matrices of size $|S| \times |S|$ rather than $n \times n$, and y as a vector of size $|S|$. (Note that X and $\tilde{X}$ are the submatrices of the original matrices, and $\mathtt{Tr}(X) \le n$). With some abuse of terminology we will redefine $V = S$ and $n = |S|$, and refer to the original variables as $V_{\mathtt{orig}}$ and $n_{\mathtt{orig}} \in [n, 2n]$. Thus, from now on we make the following assumption.

Assumption 2. *$||\tilde{v}_i||^2 \le 2$ for all $i \in V$ and $\sum_{i,j\in V: i<j} ||\tilde{v}_i - \tilde{v}_j||^2 \ge \frac{an^2_{\mathtt{orig}}}{4} \ge \frac{an^2}{4}$.*

Let us set $y_i = \frac{\alpha}{n}$ for all $i \in V$ and $z_S = 0$ for all S, then $\sum_i y_i + an^2_{\mathtt{orig}} \sum_S z_S = \alpha$. Note that $N^{(t)} = \frac{\alpha}{n}I + \sum_p f_p T_p - F$ and $\frac{\alpha}{n}I \bullet X \le \alpha$, so condition $(\sum_p f_p T_p - F) \bullet X \le -\alpha$ would imply that $N^{(t)} \bullet X \le 0$. Our goal thus becomes as follows.

Find variables $f_p \ge 0$ and symmetric matrix $F \preceq C$ such that $(\sum_p f_p T_p - F) \bullet X \le -\alpha$.

We arrive at the specification of `Oracle` given in Section 2.1.

B PROOF OF LEMMA 3.4

(a) We repeat the argument from [4]. Since the total flow is at most $\pi c' n$, at least $c'n$ newly added source edges are not saturated by the flow, and hence their endpoints are on the side of the source node in the cut obtained, which implies that that side has at least $c'n$ nodes. Similarly, the other side of the cut also has at least $c'n$ nodes, and thus the cut is c'-balanced.

(b) Assume that condition (ii) is false. By a standard argument, Assumption 2 implies the following: there exist constants $c' \in (0, c)$, $\beta > 0$ and $\sigma > 0$ such that with probability at least β Algorithm 1 reaches line 9 and we have $w_y - w_x \ge \sigma$ for all $x \in A$, $y \in B$ (see [11, Lemma 14]). Suppose that this event happens. We claim that in this case $|M| \ge \frac{1}{3}c'n$. Indeed, suppose this is false. Let $A' \subseteq A$ and $B' \subseteq B$ be the sets of nodes involved in M (with $|A'| = |B'| = |M| = k$). The total value of flow from A to B is at least $c'\pi n$ (otherwise we would have terminated at line 5). The value of flow leaving A' is at most $|A'| \cdot \pi \le \frac{1}{3}c'\pi n$. Similarly, the value of flow entering B' is at most $|B'| \cdot \pi \le \frac{1}{3}c'\pi n$. Therefore, the value of flow from $A - A'$ to $B - B'$ is at least $c'\pi n - 2 \cdot \frac{1}{3}c'\pi n = \frac{1}{3}c'\pi n$. For each edge $(x, y) \in M_{\mathtt{all}}$ with $x \in A - A'$, $y \in B - B'$ we have $||x - y||^2 > \Delta$ (otherwise M would not be a maximal matching in $M_{\mathtt{short}}$). Therefore,

$$\sum_{p:p=(x,\dots,y)} f_p||x-y||^2 \ge \sum_{\substack{p:p=(x,\dots,y)\\ x\in A-A', B\in B-B'}} f_p||x-y||^2 \ge \tfrac{1}{3}c'\pi n \cdot \Delta = 2\alpha$$

But then the algorithm should have terminated at line 7 - a contradiction.

REFERENCES

[1] Arpit Agarwal, Sanjeev Khanna, Huan Li, Prathamesh Patil, Chen Wang, Nathan White, and Peilin Zhong. 2024. Parallel Approximate Maximum Flows in Near-Linear Work and Polylogarithmic Depth. In *SODA*.

[2] Alexandr Andoni, Clifford Stein, and Peilin Zhong. 2020. Parallel approximate undirected shortest paths via low hop emulators. In *STOC*.

[3] S. Arora, E. Hazan, and S. Kale. 2010. $O(\sqrt{\log n})$ approximation to SPARSEST CUT in $\tilde{O}(n^2)$ time. *SIAM J. Comput.* 39(5) (2010), 1748–1771.

[4] S. Arora and S. Kale. 2016. A Combinatorial, Primal-Dual Approach to Semidefinite Programs. *J. ACM* 63(2) (2016), 1–35.

[5] S. Arora, S. Rao, and U. V. Vazirani. 2009. Expander flows, geometric embeddings and graph partitioning. *J. ACM* 56(2) (2009), 1–37.

[6] András A. Benczúr and David R. Karger. 1996. Approximating s-t minimum cuts in $O(n^2)$ time. In *STOC*. 47–55.

[7] Yi-Jun Chang and Thatchaphol Saranurak. 2019. Improved Distributed Expander Decomposition and Nearly Optimal Triangle Enumeration. In *PODC*.

[8] Chandra Chekuri and Kent Quanrud. 2019. Submodular function maximization in parallel via the multilinear relaxation. In *SODA*.

[9] Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. 2022. Maximum Flow and Minimum-Cost Flow in Almost-Linear Time. In *FOCS*.

[10] L. Fleischer. 2000. Approximating fractional multicommodity flow independent of the number of commodities. *SIAM Journal on Discrete Mathematics* 13(4) (2000), 505–520.

[11] S. Kale. 2007. *Efficient Algorithms Using the Multiplicative Weights Update Method.* Ph. D. Dissertation. Princeton University, Princeton, NJ. Technical Report TR-804-07.

[12] R. Khandekar, S. Rao, and U. Vazirani. 2006. Graph partitioning using single commodity flows. In *STOC*. 385–390.

[13] Vladimir Kolmogorov. 2023. A simpler and parallelizable $O(\sqrt{\log n})$-approximation algorithm for Sparsest Cut. *arXiv:2307.00115* (June 2023).

[14] Fabian Kuhn and Anisur Rahaman Molla. 2015. Distributed Sparse Cut Approximation. In *19th International Conference on Principles of Distributed Systems (OPODIS)*. 10:1–10:14.

[15] Lap Chi Lau, Kam Chuen Tung, and Robert Wang. 2024. Fast Algorithms for Directed Graph Partitioning Using Flows and Reweighted Eigenvalues. In *SODA*. arXiv:2306.09128 (June 2023).

[16] James R. Lee. 2005. On distance scales, embeddings, and efficient relaxations of the cut cone. In *SODA*. 92–101.

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

[18] Elchanan Mossel, Ryan O'Donnell, Oded Regev, Jeffrey E. Steif, and Benny Sudakov. 2006. Non-interactive correlation distillation, inhomogeneous Markov chains, and the reverse Bonami-Beckner inequality. *Israel Journal of Mathematics* 154 (2006), 299–336.

[19] Danupon Nanongkai and Thatchaphol Saranurak. 2017. Dynamic spanning forest with worst-case update time: adaptive, Las Vegas, and $O(n^{1/2-\varepsilon})$-time. In *STOC*. 1122–1129.

[20] L. Orecchia, L. J. Schulman, U. V. Vazirani, and N. K. Vishnoi. 2008. On partitioning graphs via single commodity flows. In *STOC*. 461–470.

[21] Lorenzo Orecchia and Nisheeth K. Vishnoi. 2011. Towards an SDP-based Approach to Spectral Methods: A Nearly-Linear-Time Algorithm for Graph Partitioning and Decomposition. In *SODA*.

[22] Richard Peng and Daniel A. Spielman. 2014. An efficient parallel solver for SDD linear systems. In *STOC*.
[23] Atish Das Sarma, Anisur Rahaman Molla, and Gopal Pandurangan. 2015. Distributed Computation of Sparse Cuts via Random Walks. In *Proceedings of the 16th International Conference on Distributed Computing and Networking (ICDCN)*. 1–10.
[24] J. Sherman. 2009. Breaking the multicommodity flow barrier for $O(\sqrt{\log n})$-approximations to sparsest cut. In *FOCS*. 363–372.
[25] Daniel D. Sleator and Robert Endre Tarjan. 1983. A data structure for dynamic trees. *J. of Computer and System Sciences* 26(3) (1983), 362–391.