$O(\sqrt{\log n})$ Approximation to Sparsest Cut in $\tilde{O}(n^2)$ Time

Sanjeev Arora[*]
Princeton University
35 Olden St,
Princeton NJ 08540
arora@cs.princeton.edu

Elad Hazan[†]
IBM Almaden
650 Harry Rd,
San Jose, CA 95120
hazan@us.ibm.com

Satyen Kale[†]
Microsoft Research
1 Microsoft Way,
Redmond, WA 98052
sakale@microsoft.com

Abstract

We show how to compute $O(\sqrt{\log n})$-approximations to Sparsest Cut and Balanced Separator problems in $\tilde{O}(n^2)$ time, thus improving upon the recent algorithm of Arora, Rao and Vazirani (2004). Their algorithm uses semidefinite programming and required $\tilde{O}(n^{9.5})$ time. Our algorithm relies on efficiently finding *expander flows* in the graph and does not solve semidefinite programs. The existence of expander flows was also established by Arora, Rao, and Vazirani.

1 Introduction

Partitioning a graph into two (or more) large pieces while minimizing the size of the "interface" between them is a fundamental combinatorial problem. Graph partitions or separators are central objects of study in the theory of Markov chains, geometric embeddings and are a natural algorithmic primitive in numerous settings, including clustering, divide and conquer algorithms, PRAM emulation, VLSI layout, and packet routing in distributed networks. Since finding optimal separators is NP-hard, one is forced to settle for approximation algorithms (see [28]).

The following are some of the basic problems in this class. We are given a graph $G = (V, E)$ with specified capacities c_e for every edge e. Let $n \doteq |V|, m \doteq |E|$. For any cut $(S, \bar{S})$ where $\bar{S} = V \setminus S$ and $|S| \leq n/2$, the *edge expansion* of the cut is $E(S, \bar{S})/|S|$, where $E(S, \bar{S})$ is the total capacity of the edges crossing the cut. In the Sparsest Cut problem we wish to determine the cut with the smallest edge expansion:

$$\alpha(G) = \min_{S \subseteq V, |S| \leq n/2} \frac{E(S, \bar{S})}{|S|}. \tag{1}$$

A cut $(S, \bar{S})$ is *c-balanced* if both $S, \bar{S}$ have at least cn vertices. In the minimum c-Balanced Separator problem we wish to determine $\alpha_c(G)$, the minimum expansion of c-balanced cuts. In the Graph Conductance problem we wish to determine

$$\Phi(G) = \min_{S \subseteq V, E(S) \leq E(V)/2} \frac{E(S, \bar{S})}{E(S)}, \tag{2}$$

where $E(S)$ denotes the sum of degrees (in terms of edge capacities) of nodes in S.

Efforts to design good approximation algorithms for these NP-hard problems have spurred the development of many subfields of theoretical computer science. The earliest algorithms relied on spectral methods introduced — in the context of Riemannian manifolds — by Cheeger [11] and improved by Alon and Milman [3] and Alon [2]. Though this connection between eigenvalues and conductance only yields a weak approximation (the worst-case approximation ratio is n), it has had enormous influence in a variety of areas, including random walks, pseudorandomness, error-correcting codes, and routing.

[*]Supported by David and Lucile Packard Fellowship, NSF grants CCR 0205594, CCR 0098180, and MSPA-MCS 0528414.
[†]Work done while the author was at Princeton University.

Leighton and Rao [21] designed the first true approximation by giving $O(\log n)$-approximations for Sparsest Cut and Graph Conductance and $O(\log n)$-pseudo-approximations for the minimum c-Balanced Separator. They used a linear programming relaxation of the problem based on multicommodity flows proposed in [27]. Leighton, Rao and others used similar ideas to design approximation algorithms for numerous NP-hard problems, see the surveys [28, 29]. Furthermore, efforts to improve these ideas led to progress in other areas, such as fast computations of multicommodity flows and packing-covering linear programs [30, 26, 15], and efficient geometric embeddings of metric spaces [22]; see also [7].

Recently, Arora, Rao and Vazirani (ARV) [5] designed an $O(\sqrt{\log n})$-approximation algorithm. They use semidefinite programming (SDP), a technique introduced in approximation algorithms by Goemans and Williamson [16]. The running time of the ARV algorithm is dominated by the solution of this SDP, which takes $\tilde{O}(n^{9.5})$ time[1] using interior point methods [1]. (Here and in the rest of the paper $\tilde{O}(\cdot)$ notation is used to suppress polylogarithmic factors.) New techniques in high-dimensional geometry introduced by their analysis have already found use in several recent manuscripts, most significantly in one that makes progress on a longstanding open question about the embeddability of ℓ_1 spaces into ℓ_2 ([10]).

The ARV paper also outlined an alternative approximation algorithm using *Expander flows*. These are multicommodity flows in the graph whose *demand graph* (i.e., the weighted graph with weights d_{ij} on edge $\{i, j\}$, where d_{ij} is the flow shipped between nodes i, j) is an expander. This flow can be seen as an *embedding* of the demand graph in the host graph, and thus this idea is descended from the work of Leighton and Rao, who showed how to approximate sparsest cut by embedding the complete graph (which in particular is an expander) in the host graph. The important difference here is the *edge congestion*, i.e., maximum amount of flow using an edge. The flows exhibited by Arora, Rao, Vazirani are efficient enough to work with a $\sqrt{\log n}$ factor lower congestion than Leighton-Rao flows. Thus these flows can be used to certify that the expansion is $\Omega(\alpha(G)/\sqrt{\log n})$. (Note that determining graph expansion is coNP-complete [9], so we cannot expect to have succinct certificates that prove that the expansion is exactly $\alpha(G)$.)

In addition to being an interesting graph theoretic fact — analogous to, say, the approximate max-flow min-cut theorem that underlies Leighton-Rao's result — the existence of such expander flows seems to hint at a faster version of the ARV approximation algorithm for Sparsest Cut. After all, computation of multicommodity flows is a highly developed area today. Thus an approximation algorithm for Sparsest Cut could try to route a multicommodity flow in the graph, and modify it using eigenvalue computations that check if the current demands form an expander. If an expander flow exists (given a certain upper bound on congestion) then the final multicommodity flow would converge to it. If the expander flow doesn't exist, the algorithm would presumably find a very sparse cut that "proves" this fact. The authors of [5] suggested this approach for designing faster algorithms, though the best algorithm they could come up with used the Ellipsoid method, and hence was less efficient even than the SDP-based one.

This paper presents a $\tilde{O}(n^2)$-time randomized algorithm that uses expander flows to compute a $O(\sqrt{\log n})$-approximation to Sparsest Cut. This essentially matches the running time of the best implementations of Leighton-Rao's $O(\log n)$-approximation (Benczúr and Karger [8], the last paper in a long line of work). Our algorithm computes an expander flow in a sparse weighted graph that is obtained by Benczúr and Karger using a (nontrivial) random sampling on the original graph. This expander flow suffices to certify the expansion of the original graph. Furthermore, the algorithm produces, in addition to expander flows, a distribution on $O(\log n)$ balanced cuts, which can be viewed as an ℓ_1 metric, and thus can be embedded in ℓ_2^2. Then we use the ideas of Arora-Rao-Vazirani to obtain the $O(\sqrt{\log n})$-approximate sparsest cut from this metric. Our algorithm can also yield expander flows in any weighted graph on m edges in $\tilde{O}(m^2)$ time.

We note that the ideas of ARV [5] have led to $O(\sqrt{\log n})$ approximation using SDPs for many other problems. Recently, Arora and Kale [4] gave a more general primal-dual schema for SDPs which they use to design a variety of fast approximation algorithms for many of these problems. They cannot improve the running time below $\tilde{O}(n^2)$ for Sparsest Cut and Balanced Separator if one insists on $O(\sqrt{\log n})$ approximation. But they can get $\tilde{O}(n^{1.5}+m)$ time for $O(\log n)$-approximation, which improves upon the best running times known for the Leighton-Rao approach. Khandekar, Rao and Vazirani [18] had earlier achieved $\tilde{O}(n^{1.5}+m)$ time for $O(\log^2 n)$-approximation, which was improved by Orecchia, Schulman, Vazirani and

[1]The earlier version of this paper, which appeared in FOCS 2004, incorrectly stated the running time of the interior point solver as $\tilde{O}(n^{4.5})$. The correct running time bound, as stated here, is deduced as follows: a typical interior point solver takes $\tilde{O}(\sqrt{n})$ iterations, and in each iteration, the running time is dominated by solving an $m \times m$ system of linear equations, where $m = O(n^3)$ is the number of constraints. This takes $\tilde{O}(n^9)$ time.

Vishnoi [25] to $O(\log n)$-approximation with the same running time.

Overview of methodology

As mentioned, we find sparse cuts by finding expander flows. We first use the sparsification technique of Benczúr-Karger to transform our graph to a sparse weighted graph in which $m = \tilde{O}(n)$. The value of the sparsest cut is essentially unchanged, so we find expander flows in the sparse graph.

The ARV approach to finding expander flows involves solving an LP, which has exponentially many constraints to stipulate the condition that the demand graph is an expander. Specifically, if the amount of flow leaving each node is D, then for each cut $(S, \overline{S})$ one has a constraint stating that the amount of flow crossing that cut (i.e., whose source and sink are on opposite sides) is proportional to $D \cdot \min\{|S|, |\overline{S}|\}$. Though the separation problem for this LP is NP-hard, one can use the eigenvalue approach to design an approximate separation oracle. Thus the ARV algorithm consists of using the this efficient separation oracle and the Ellipsoid method to find a feasible flow.

The main idea in our paper is to use a primal-dual framework instead of the Ellipsoid method. We note that the LP for finding expander flows also has packing and covering constraints and many papers have described efficient algorithms for packing and covering problems. However, we cannot see how to solve it efficiently using conventional approaches such as [26]. We choose to go with an unconventional choice, the Freund-Schapire [14] method for approximately solving a 2-person zero-sum game. It is possible that our results can be derived using more standard analyses, especially Young [30] or Garg-Könemann [15] However, the advantage of the game theoretic setting is that it naturally allows us to introduce nonlinearity in the payoff function, as explained in the few lines around equation (7). Besides, we find the game theoretic setting more natural. (We have a longer survey article on such "Multiplicative Weight Update" algorithms [6] that explains the relationships among these different algorithms, and gives a single meta-algorithm that generalizes all of them.)

One has several choices how to represent expander flows in the zero-sum game, and the obvious ones do not result in quick algorithms (for some failed ideas see Section 3). The correct choice is to ignore the flow and to instead maintain the *demands*. The game is defined so that near-optimal solutions yield demands corresponding to an expander flow — more precisely, a "pseudo-expander flow," which is "almost" an expander flow. The need for considering pseudo-expander flows arises from a lack of precision, which comes from two sources. First, we are using approximate flow algorithms and eigenvalue computations anyway. Second and more importantly, we design the game by a careful attention to its "width" parameter to ensure a quick convergence. For both these reasons, the final solution to the game is fairly coarse, but an expander flow is such a robust object that anything even remotely resembling it can be easily turned into a true expander flow.

We design our algorithm by giving efficient strategies for both players, and proving that repeatedly playing these strategies causes them to converge to the optimum value of the game in $O(\log n)$ rounds. To make our strategies for the players run in $\tilde{O}(n^2)$ time, we use a combination of random sampling (above and beyond the use of Benczúr-Karger at the very start), approximate min-cost concurrent multicommodity computations, and approximate eigenvalue computations. The outline appears in Section 3 and subsequent sections fill in the details. We note that when the game fails to result in an expander flow, then one can produce an approximately optimum sparsest cut; this part relies on [5].

Our algorithm uses some well-known ideas that not explained in the literature in the exact form needed here. Therefore we have included the proofs in the appendix.

2 Freund-Schapire technique

We start by describing Freund and Schapire's method for adaptively solving a two-person zero-sum game. (As they note, the method itself is quite old.) The game has two players, the *row* player and the *column* player, who choose their moves from sets $\mathcal{R}$, $\mathcal{C}$ respectively. We let N denote $|\mathcal{R}|$; the size of $\mathcal{C}$ will play no role. For $i \in \mathcal{R}$, $j \in \mathcal{C}$ let $\mathbf{M}(i,j)$ denote the payoff from the row player to the column player when they play these moves. If the row player chooses his strategy i from a distribution $\mathcal{D} = \{p_1, p_2, \ldots, p_N\}$ over $\mathcal{R}$, the expected payoff to the column player is $\sum_{i \in \mathcal{R}} p_i \mathbf{M}(i,j)$, which we denote by $\mathbf{M}(\mathcal{D}, j)$. We assume that

$\mathbf{M}(i,j) \in [\ell, u]$ for some parameters ℓ, u. Let $\rho = u - \ell$, this is called the *width* of the $\mathbf{M}$; it will affect the running time of the algorithm below. Define $\mu(i,j) = (\mathbf{M}(i,j) - \ell)/\rho$, so $\mu(i,j) \in [0,1]$. Similarly, for a distribution $\mathcal{D} = \{p_1, p_2, \ldots, p_N\}$ on $\mathcal{R}$, define $\mu(\mathcal{D}, j) = \sum_{i \in \mathcal{R}} p_i \mu(i,j)$. Consider the following algorithm:

The Multiplicative Weights algorithm

Fix an $\varepsilon < \frac{1}{2}$. For all $i \in \mathcal{R}$, initialize the weights ${w_i}^{(1)} = 1$. In every round t, for $t = 1, 2, \ldots, T$:

1. The row player chooses strategy $i \in \mathcal{R}$ with probability ${p_i}^{(t)} = {w_i}^{(t)}/\Phi^{(t)}$, where $\Phi^{(t)} = \sum_i {w_i}^{(t)}$. Let this distribution be $\mathcal{D}^{(t)}$.
2. The column player chooses a strategy $j^{(t)} \in \mathcal{C}$ and obtains the expected payoff $\mathbf{M}(\mathcal{D}^{(t)}, j^{(t)})$.
3. The row player updates the weights as:

$$ {w_i}^{(t+1)} \;=\; {w_i}^{(t)} \cdot (1-\varepsilon)^{\mu(i, j^{(t)})}. $$

The following theorem gives guarantees on the total expected loss to the row player in terms of the total expected loss of the best fixed row strategy in hindsight. The proof appears in Appendix D.

Theorem 1
For any $\delta > 0$, let $\varepsilon = \delta/2\rho$, $T = 4\rho^2 \ln(N)/\delta^2$. Then for any distribution $\mathcal{D}$ over $\mathcal{R}$,

$$ \frac{1}{T}\sum_{t=1}^{T} \mathbf{M}(\mathcal{D}^{(t)}, j^{(t)}) \;\le\; \frac{1}{T}\sum_{t=1}^{T} \mathbf{M}(\mathcal{D}, j^{(t)}) + \delta. \tag{3} $$

By von Neumann's min-max theorem, the zero-sum game has an associated game value. Call it λ^*. Let $\mathcal{D}^*$ be the row player's optimum distribution, namely, the $\mathcal{D}$ that minimizes $\max_{j \in \mathcal{C}} \mathbf{M}(\mathcal{D}, j)$. Then the value of the game is $\lambda^* = \max_{q \in \mathcal{C}} \mathbf{M}(\mathcal{D}^*, q)$. By definition, for every distribution $\mathcal{D}$ over $\mathcal{R}$, we have $\max_{j \in \mathcal{C}} \mathbf{M}(\mathcal{D}, j) \ge \lambda^*$. An optimal play for the column player for a given distribution $\mathcal{D}$ for the row player is a $j \in \mathcal{C}$ which achieves this maximum. We have the following easy corollary to Theorem 1:

Corollary 2
If the column player plays optimally in each round then $\frac{1}{T}\sum_{t=1}^{T} \mathbf{M}(\mathcal{D}^t, j^t) \;\in\; [\lambda^*, \lambda^* + \delta]$. This holds even if we assume that the column player plays near-optimally, ie. always ensures that the payoffs are $\ge \lambda^*$.

Proof: The fact that $\frac{1}{T}\sum_{t=1}^{T} \mathbf{M}(\mathcal{D}^{(t)}, j^{(t)}) \ge \lambda^*$ follows because in each round t, the payoff $\mathbf{M}(\mathcal{D}^{(t)}, j^{(t)})$ is at least λ^*. Now, consider the distribution $\mathcal{P}$ on $\mathcal{C}$ which assigns a probability measure of $\frac{1}{T}$ to each $j^{(t)}$ for $t = 1, 2, \ldots, T$ (aggregating measure for repeated $j^{(t)}$'s). By von Neumann's min-max theorem, there is a strategy $i \in \mathcal{R}$ for the row player such that the expected payoff to the column player if he chooses his strategy $j \in \mathcal{C}$ using the distribution $\mathcal{P}$ is at most λ^*, i.e. for some $i \in \mathcal{R}$, $\frac{1}{T}\sum_{t=1}^{T} \mathbf{M}(i, j^{(t)}) \le \lambda^*$. Choosing the distribution $\mathcal{D}$ to be the one that chooses i with probability 1, we get that the right hand side of inequality (3) is at most $\lambda^* + \delta$, as required. □

3 Expander flows and algorithm overview

This section defines expander flows, and outlines the main ideas in our algorithm for finding them.

All weighted graphs in this paper are symmetric, that is $d_{ij} = d_{ji}$ for all node pairs i, j. We call $d_i = \sum_j d_{ij}$ the *degree* of node i. We emphasize that degrees can be fractions (i.e., less than 1).

A *multicommodity flow* in an unweighted graph $G = (V, E)$ is an assignment of a *demand* $d_{ij} \ge 0$ to each node pair $\{i, j\}$ such that we can route d_{ij} units of flow between i and j, and can do this simultaneously for all pairs without violating any edge capacities. We refer to the weighted graph (d_{ij}) as the *demand graph* of the flow. Given a subset $S \subseteq V$ the *demand crossing the cut* $(S, \bar{S})$ is the capacity of the cut $(S, \bar{S})$ in the demand graph, i.e. $d(S, \bar{S}) = \sum_{i \in S, j \in \bar{S}} d_{ij}$.

Definition 1 *A demand graph of a set of demands* $\{d_{ij}\}$ *is a* D*-regular* β*-expander if it has maximum degree at most* D *and for any subset* $S \subseteq V$ *such that* $|S| \leq n/2$ *the demand crossing the cut* $(S, \bar{S})$ *satisfies*

$$d(S,\bar{S}) \;\geq\; \beta D|S|.$$

Note that we have relaxed D-regularity and only require maximum degree D; this is without loss of generality [5] since one could add self-loops to raise all degrees to D.

Lemma 3
If a graph G *admits a multicommodity flow whose demand graph is a* D*-regular* β*-expander, then its expansion is at least* βD.

Proof: Let d_{ij} be the demands in the D-regular β-expander flow. Then for any $S \subseteq V$ with $|S| \leq n/2$, the capacity of the cut $(S, \bar{S})$ must be at least the demand crossing it. Thus,

$$E(S,\bar{S}) \;\geq\; d(S,\bar{S}) \;\geq\; \beta D|S|,$$

which implies that $\frac{E(S,\bar{S})}{|S|} \geq \beta D$. Thus, the expansion of G is at least βD. □

We will refer to such a flow as an expander flow for short. Another notion we will need is that of a pseudo-expander flow:

Definition 2 *A demand graph of a set of demands* $\{d_{ij}\}$ *is a* D*-regular* (c, β)*-pseudo expander if it has maximum degree at most* D *and for any subset* $S \subseteq V$ *such that* $cn \leq |S| \leq n/2$ *the demand crossing the cut* $(S, \bar{S})$ *satisfies*

$$d(S,\bar{S}) \;\geq\; \beta D|S|.$$

Notice that a D-regular β-expander flow is in particular a D-regular (γ, β)-pseudo expander flow for each γ. Just like expander flows, pseudo-expander flows can be used to obtain lower bounds on the expansion of balanced cuts:

Lemma 4
If a graph G *admits a multicommodity flow whose demand graph is a* D*-regular* (c, β)*-pseudo expander, then the expansion of all* c*-balanced cuts is at least* βD.

This lemma is proved just as before. The following theorem of Arora, Rao and Vazirani [5] show that the notions of expander flows and pseudo-expander flows allow us to obtain $O(\sqrt{\log n})$ approximations to the expansions of the Sparsest Cut and minimum c-Balanced Separator respectively:

Theorem 5 ([5])
There is a constant $\beta_0 > 0$ *such that every graph* $G = (V, E)$ *admits a* D*-regular* β_0*-expander flow, where* $D = \Omega(\alpha(G)/\sqrt{\log n})$. *Further, every graph* G *admits a* D*-regular* (c, β_0)*-pseudo expander flow, where* $D = \Omega(\alpha_{c'}(G)/\sqrt{\log n})$ *for some* $c' \leq c$.

The essence of our work is to show how to efficiently compute such expander flows. To understand this algorithm it helps to first look at an LP whose feasibility is implied by the above Theorem. Let D be the degree of interest; think of it as a "constant" in the LP, not as a variable.

For all simple paths p in the graph, we have a non-negative variable f_p. Let $\mathcal{P}_{ij}$ be the set of paths connecting node pair $\{i, j\}$, $\mathcal{P}_{i*}$ be the set of paths originating from node i, and $\mathcal{P}_{S,\bar{S}}$ be the set of paths having end points on either side of the cut $S, \bar{S}$. In the following LP, we use the notation "$\forall S$" to only refer to subsets of vertices of size at most $n/2$. The Primal LP is

$$\begin{aligned}
\forall i: \sum_{p \in \mathcal{P}_{i*}} f_p \;&\leq\; D \\
\forall e: \sum_{p:\; e \in p} f_p \;&\leq\; c_e \\
\forall S: \sum_{p \in \mathcal{P}_{S,\bar{S}}} f_p \;&\geq\; \beta_0 D|S| \qquad (4)
\end{aligned}$$

The LP for D-regular (c, β_0)-pseudo expander flows is the same as the one above except that the third set of constraints is only over subsets of vertices S such that $cn \leq |S| \leq n/2$. We give a unified treatment of both LPs, and in the following sections, whenever a subset of vertices S occurs, it is implicitly assumed that $|S| \leq n/2$, and, in the case of the c-Balanced Separator problem, $|S| \geq cn$. We will need to use the constant c even in the context of Sparsest Cut, and in this case we set $c = \frac{1}{2}$.

As outlined in [5], the Primal LP can be solved to near-optimality in polynomial time by an Ellipsoid-like method, using an eigenvalue computation as a separation oracle. To design a better algorithm we aim to associate a zero-sum game with it and use the Freund-Schapire framework. Since the algorithm in that framework maintains a distribution on all pure row strategies, it is important to work with games where the number of pure row strategies is polynomial. In particular, we need a polynomial-size representation of the flow. The standard representation uses variables f_{ije} for each demand pair $(i,j) \in V \times V$ and edge $e \in E$. We do not know how to formulate our algorithm using this representation, and even if we did, the number of variables (i.e., number of pure strategies for the row player) would be $\Omega(n^2 m)$, which would be a lower bound on the running time[2]. The idea instead is to not use any representation of the flows at all, and to maintain only the *demands* d_{ij}. Now the number of variables is $\binom{n}{2}$ and so we at least have a prayer of achieving $\tilde{O}(n^2)$ running time.

The design of the zero-sum game is inspired by the Dual LP to (4). In this Dual LP we have non-negative variables s_i, w_e, z_S corresponding to vertex i, edge e, and subset $S \subseteq V$ respectively.

$$\begin{aligned} \min \quad & D\sum_i s_i + \sum_e c_e w_e - \beta_0 D \sum_S |S| z_S \\ \forall ij, \forall p \in \mathcal{P}_{ij}: \quad & s_i + s_j + \sum_{e \in p} w_e - \sum_{S:\ i \in S, j \in \bar{S}} z_S \geq 0 \end{aligned} \tag{5}$$

The Primal is feasible iff the optimum of the Dual LP is non-negative. Represent the Primal LP in matrix form as $A\bar{f} \leq b, \bar{f} \geq 0$ where $\bar{f}$ is a vector of flow assignments to paths. Then, given a candidate flow $\bar{f}$, one could show that it is *infeasible* if (by Farkas' lemma, if and only if) one demonstrates a vector $\bar{x} = \langle \bar{s}, \bar{w}, \bar{z} \rangle$ of Dual variables such that $\bar{x}^\top A\bar{f} \geq 0$ but $\bar{x}^\top b < 0$. In other words, a candidate flow $\bar{f}$ will be infeasible if one demonstrates Dual variables $\bar{x}$ such that the linear combination $\sum_p f_p(s_i + s_j + \sum_{e \in p} w_e - \sum_{S:\ i \in S, j \in \bar{S}} z_S) \geq 0$ but the Dual objective $D\sum_i s_i + \sum_e c_e w_e - \beta_0 D \sum_S |S| z_S < 0$. The crucial observation is that the set of paths in the constraints of (5) can be restricted to just the *shortest paths* between the $\{i,j\}$ pairs under edge weights w_e. In this case, all the flow between a pair of nodes $\{i,j\}$ can be aggregated into a *demand* d_{ij}.

The above discussion leads naturally to the following 2-person zero-sum game (our algorithm will use a modified version of this game). The row player's strategy set are the node pairs $\{i,j\}$ and he tries to show feasibility of the Primal LP by producing a set of candidate demands d_{ij} corresponding to node pairs $\{i,j\}$, such that $\sum_{i<j} d_{ij} = \frac{1}{2}nD$. Thus $\{\frac{d_{ij}}{\frac{1}{2}nD}\}_{ij}$ is a distribution on pure strategies, which will be updated using the Freund-Schapire rules. As already mentioned, these d_{ij}'s correspond to demands for a multicommodity flow problem; we emphasize that the demands need not correspond to a routable flow, in other words, routing them could require gravely exceeding edge capacities.

The column player tries to foil the row player in his task of showing feasibility by picking dual variables $\bar{x} = \langle \bar{s}, \bar{w}, \bar{z} \rangle$ such that

1. the Dual objective $D\sum_i s_i + \sum_e c_e w_e - \beta_0 D \sum_S |S| z_S < 0$:
   We ensure this by making him choose $\bar{x}$ from the polytope $\mathbf{P}_1$ of $\langle \bar{s}, \bar{w}, \bar{z} \rangle$ satisfying

$$D\sum_i s_i + \sum_e c_e w_e \leq \beta n D; \ \sum_S |S| z_S = n \tag{6}$$

   Here $\beta \ll \beta_0$ is a small positive constant to be chosen later.

[2] Note that it is possible to use random sampling to reduce the number of nonzero demands to $O(n \log n)$; in fact this is done during every iteration of our algorithm while computing the best response for the column player. However, the Freund-Schapire update rule seems to require updating the distribution on all row strategies, and indeed we update all n^2 demands at every iteration.

2. the linear combination $\sum_{ij} d_{ij}(s_i + s_j + \min_{p\in\mathcal{P}_{ij}}\{\sum_{e\in p} w_e\} - \sum_{S:\ i\in S, j\in\bar{S}} z_S) \geq 0$:
   We make this linear combination the payoff of the game, so the column player always tries to get a non-negative payoff. The payoff for the pure row strategy $\{i,j\}$ and a column strategy $\bar{x} \in \mathbf{P}_1$, is $f_{ij}(\bar{x}) \doteq s_i + s_j + l_{ij}(\bar{x}) - \sum_{S:i\in S, j\in\bar{S}} z_S$, where $l_{ij}(\bar{x}) \doteq \min_{p\in\mathcal{P}_{ij}}\{\sum_{e\in p} w_e\}$ is the length of the shortest path from i to j with respect to weight function w_e. Given a set of demands $\bar{d} = \{d_{ij}\}_{ij}$, define the *payoff function*

$$v(\bar{d},\bar{x}) \ \doteq\ \sum_{ij} d_{ij} f_{ij}(\bar{x}) \ =\ \sum_i d_i s_i + \sum_{ij} d_{ij} l_{ij}(\bar{x}) - \sum_S d(S,\bar{S}) z_S. \tag{7}$$

   Note that the payoff function is nonlinear due to the presence of l_{ij}. Thus, the expected payoff to the column player for the mixed strategy $\{\frac{d_{ij}}{\frac{1}{2}nD}\}_{ij}$ is $\frac{v(\bar{d},\bar{x})}{\frac{1}{2}nD}$.

We will show that if a D-regular β_0-expander flow exists then the value of the game is $\leq -2(\beta_0 - \beta)$. We will also show that at each step, a near-optimal response for the column player can be computed — using a combination of mincost concurrent flow, eigenvalue computations, and random sampling — in $\tilde{O}(n^2)$ time. The width of the game is $O(n)$, which implies that the game converges to the optimal value after $\tilde{O}(n^2)$ rounds. In fact, we can stop the game as soon as the current demands are such that the column player cannot force a positive payoff, at which point the demands are "close enough" to an expander and can be easily extended to a *bona fide* expander flow (though with expansion somewhat less than β_0). Thus the overall running time would be $\tilde{O}(n^2 \times n^2) = \tilde{O}(n^4)$. This is only slightly better than solving SDPs or LPs.

To achieve a better running time we redesign the game — as well as the column player — carefully so that the width is actually $O(1)$. Then Theorem 1 implies that convergence occurs in $O(\log n)$ rounds, and the running time is $\tilde{O}(n^2)$.

4 Implementing the Game: Details

Now we give a more detailed analysis. To present the $\tilde{O}(n^2)$ algorithm we will need a modified game, but first let us understand the game from the previous section whose payoff is given by (7).

Let the value of this game be $\lambda^* = \min_{\bar{d}} \max_{\bar{x}} \frac{v(\bar{d},\bar{x})}{\frac{1}{2}nD}$. We begin by showing that if a D-regular β_0-expander flow can be embedded in G, then λ^* is highly negative:

Lemma 6
If D-regular β_0-expander flow can be embedded in the graph, then the value of the game is at most $-2(\beta_0 - \beta)$.

Proof: Let the expander flow assign flow $f_p \geq 0$ to every path p. We define $d^*_{ij} = \sum_{p\in\mathcal{P}_{ij}} f_p$. Since the flow is D-regular, the demands sum to at most $\frac{1}{2}nD$. Now for every $\bar{x} \in \mathbf{P}_1$ we have

$$\begin{aligned} v(\bar{d}^*,\bar{x}) &= \sum_i d^*_i s_i + \sum_{ij} d^*_{ij} l_{ij}(\bar{x}) - \sum_S d^*(S,\bar{S}) z_S \\ &\leq \sum_i \sum_{p\in\mathcal{P}_{i*}} f_p s_i + \sum_{ij} \sum_{p\in\mathcal{P}_{ij}} f_p \sum_{e\in p} w_e - \sum_S \sum_{p\in\mathcal{P}_{S,\bar{S}}} f_p z_S \\ &= \sum_i \sum_{p\in\mathcal{P}_{i*}} f_p s_i + \sum_e \sum_{p\ni e} f_p w_e - \sum_S \sum_{p\in\mathcal{P}_{S,\bar{S}}} f_p z_S \\ &\leq \sum_i D s_i + \sum_e c_e w_e - \sum_S \beta_0 D|S| z_S \\ &\leq\ -(\beta_0 - \beta) nD. \end{aligned}$$

The payoff, being negative, reduces even further if we scale the demands to sum to exactly $\frac{1}{2}nd$, so $\lambda^* \leq -\frac{(\beta_0-\beta)nD}{\frac{1}{2}nD} = -2(\beta_0 - \beta)$. □

The same result holds if we consider D-regular (c, β_0)-pseudo expander flows instead, in the corresponding game.

It is easily checked that the width of the game is $O(n)$, so convergence may take $\tilde{O}(n^2)$ rounds. Now we describe a related game in which the payoffs are truncated, and so the width is $O(1)$. Then convergence happens in $O(\log n)$ rounds, but converting the near-optimal d_{ij}'s into an expander flow is slightly more difficult.

To reduce width to $O(1)$, we must truncate the payoffs. Let us first truncate the l_{ij}'s: redefine $l_{ij}(\bar{x}) = \min\{\min_{p \in \mathcal{P}_{ij}} \sum_{e \in p} w_e, 1/\varepsilon\}$, where ε is a small constant defined in Section 5. The definitions of $f_{ij}, v(\bar{d}, \bar{x})$ are the same except that they use this new l_{ij}.

Next, we restrict the strategy set of the column player. Let $\mathbf{P}_2$ be the polytope of $\bar{x} = \langle \bar{s}, \bar{w}, \bar{z} \rangle$ satisfying:

1. $s_i \le 1/\varepsilon \qquad \forall i.$
2. $z_S = 0$ whenever $|S| < cn$.

Then the polytope corresponding to allowable pure strategies for the column player will be $\mathbf{P} = \mathbf{P}_1 \cap \mathbf{P}_2$, where $\mathbf{P}_1$ was defined in (6). Since $z_S > 0$ only for $|S| \ge cn$, and $\sum_S |S| z_S = n$, we have $\sum_S z_S \le 1/c$. Now from the definition of f_{ij} we see that for any $\bar{x} \in \mathbf{P}$, $-1/c \le f_{ij}(\bar{x}) \le 3/\varepsilon$. Thus the width is $1/c + 3/\varepsilon = O(1)$ and the Freund-Schapire game played against a near-optimal column player converges in $T = O(\log n)$ iterations.

The final algorithm of Theorem 9 will use binary search on D, and so for the next few paragraphs assume that D is such that a D-regular β_0-expander flow exists. Since the truncation of the previous paragraph can only decrease payoffs, Lemma 6 still holds for the truncated game, and the value of the game is very negative.

This leaves the issue of describing a near-optimal column player that runs in $\tilde{O}(n^2)$ time and to show that near-optimal row strategy $\bar{d}$ can be used to construct an expander flow. Both issues are addressed in the next theorem (proved in Section 5), which describes a specific column player, Oracle. If $\bar{d}$ ever is such that Oracle cannot enforce non-negative payoff (which must happen close to convergence), then $\bar{d}$ "almost" represents an expander flow; in fact we obtain a pseudo-expander flow. A pseudo-expander flow can be used to find either an expander flow, or a cut of expansion $O(D)$.

Theorem 7

There is a randomized procedure, Oracle, *which given a set of demands $\bar{d}$, runs in $\tilde{O}(n^2)$ time and*

1. *either produces an $\bar{x} \in \mathbf{P}$ with $v(\bar{d}, \bar{x}) \ge 0$ in which exactly one set $S \subseteq V$ with at least cn vertices has a non-zero z_S,*
2. *or else,* Oracle *fails. In this case, a pseudo-expander flow can be constructed from $\bar{d}$.*

Now we formally state what happens when the Freund-Schapire procedure is run starting with the uniform demand vector where $d_{ij} = \frac{1}{2} nD / \binom{n}{2}$ for all pairs $\{i, j\}$ and using Oracle as column player. Let $\bar{d}^t$ be the demand vector produced in the t^{th} iteration of the algorithm. Let $\bar{x}^t \in \mathbf{P}$ be the corresponding vector produced by Oracle.

Lemma 8

Suppose Oracle *never failed up to round $T = O(\log n)$. Then, for any set of demands $\bar{d}$, we have*

$$\frac{1}{T} \sum_{t=1}^{T} v(\bar{d}^t, \bar{x}^t) \;\le\; \frac{1}{T} \sum_{t=1}^{T} v(\bar{d}, \bar{x}^t) + \frac{\delta}{2} nD. \tag{8}$$

Also, if a D-regular β_0-expander flow can be routed in G, then the Oracle *must fail in $O(\log n)$ iterations.*

Proof: Inequality (8) can be obtained simply by scaling the inequality of Theorem 1 by a factor of $\frac{1}{2} nD$. As for the second part, suppose the Oracle never aborted. Substitute $\bar{d} = \bar{d}^*$ of Lemma 6 in Lemma 8. Then $v(\bar{d}, \bar{x}_t) \le -(\beta_0 - \beta) nD$ for any t, so for $\frac{\delta}{2} < \beta_0 - \beta$, the RHS of (8) is negative. On the other hand, the Oracle always returns an $\bar{x}$ which has non-negative payoff for the given set of demands, so the LHS is non-negative, a contradiction. □

The implementation of Oracle will ensure that when it fails, the demands d_{ij} must be pseudo-expander flow, which can be turned into an expander flow or a cut of low expansion. We will also show that if inequality (8) holds, then we can efficiently obtain a cut of low expansion. Thus, we have the following theorem:

Theorem 9 (Main)
For some universal constant β, there is a procedure, that given a graph G and a value D, finds either

1. *a D-regular β-expander flow, or*
2. *a cut of expansion at most $O(\sqrt{\log n} \cdot D)$.*

Furthermore, this procedure runs in time $\tilde{O}(n^2)$.

If $\bar{f}$ is a D-regular β-expander flow, then it is easy to check that for any $D' \leq D$, $\frac{D'}{D} \cdot \bar{f}$ is a D'-regular β-expander flow. Thus, by starting with a low value of D and doubling it every time, we can find a value D^* such the algorithm finds a D^*-regular β-expander flow, and a cut of expansion at most $O(\sqrt{\log n} \cdot D^*)$. Thus, by Lemma 3 we have $D^* \leq \alpha(G) \leq O(\sqrt{\log n} \cdot D^*)$, and so we have the desired $O(\sqrt{\log n})$ approximation to the Sparsest Cut.

Theorem 9 makes use of the following analogous theorem concerning pseudo-expander flows. This theorem enables us to obtain a $O(\sqrt{\log n})$ pseudo approximation to the minimum c-Balanced Separator just as before.

Theorem 10
For some universal constant β, there is a procedure, that given a graph G and a value D, finds either

1. *a D-regular (c, β)-pseudo expander flow, or*
2. *a c' balanced cut of expansion at most $O(\sqrt{\log n} \cdot D)$, for some $c' \leq c$.*

Furthermore, this procedure runs in time $\tilde{O}(n^2)$.

The algorithm of Theorem 9 actually runs the algorithm of Theorem 10 with $c = \frac{1}{2}$. At any point where it obtains a D-regular (c, β)-pseudo expander flow, it runs the algorithm given by the following lemma (the stipulation on the number of non-zero demands will be met by random sampling), which shows that we can pass from a pseudo-expander-flow to an expander flow (and when we fail to so, we can obtain a reason for the failure in form of a sparse cut). This lemma is proved in Appendix B.

Lemma 11
Let f_p be a D-regular (c, β)-pseudo-expander flow on a graph G. Assume that the flow has non-zero demand on only $O(n \log n)$ pairs of vertices. Then, there is a procedure that in time $\tilde{O}(n^2)$, finds either

1. *a D-regular $\frac{\beta^2}{130}$ expander flow,*
2. *or, a cut of expansion at most $\frac{1}{c}D$.*

A note on running time: We make a few remarks on the $\tilde{O}(n^2)$ running time, which occurs many times in the paper and in particular in the implementation of Oracle. First, one can reduce the number of nonzero demands to $\tilde{O}(n)$ by random sampling. This is a known technique from existing sparsest cut implementations (see eg [19], [8]) though we occasionally need to add a few simple ideas.

In many places we need to find cuts $(S, \bar{S})$ where the demand graph fails to expand (i.e. $d(S, \bar{S}) = o(nD)$) and the cut is large, namely $|S| = \Omega(n)$. Using the well-known results of Cheeger and Alon we can do this using approximate eigenvalue computations on the Laplacian of this sparse graph, which takes $\tilde{O}(n)$ time by repeated matrix-vector products. (This idea has been repeatedly rediscovered, but one reference is [20]). Using the eigenvector and Theorem 17 we can find cuts (if any exist) where the demand graph does not expand. Repeating the eigenvector method $O(n)$ times we try to aggregate these small cuts to have size $\Omega(n)$. If this aggregation fails to produce any large cuts that do not expand, then we can throw away $o(n)$ of the graph such that in the remaining graph all cuts expand well. (In other words, we have a pseudo-expander flow already.) Thus the total time is $\tilde{O}(n^2)$.

The Oracle procedure performs a min-cost concurrent multicommodity flow computation using the algorithm of Fleischer [13], which also takes time $\tilde{O}(n^2)$ since the number of demands has been reduced to $\tilde{O}(n)$ by random sampling.

Finally, we repeat the algorithm of Theorem 10 for successively doubling values of D. Thus overall, the algorithm for approximating Sparsest Cut takes $\tilde{O}(n^2 \cdot \log(\frac{U}{L}))$ time, where $[L, U]$ is a range of values for $\alpha(G)$.

We can bound $\frac{U}{L}$ by $O(n)$ as follows. Let the global min-cut value in the graph G be C (this value can be approximated to a constant factor in $O(m+n)$ time using Matula's algorithm [23]). Then, for any cut $(S, \bar{S})$ in the graph, $\frac{E(S,\bar{S})}{|S|} \geq \frac{C}{n}$, so $\alpha(G) \geq \frac{C}{n}$. On the other hand, the expansion of the min-cut is at most C, so $\alpha(G) \leq C$. Thus, we can take the range of $\alpha(G)$ to be $[\frac{C}{n}, C]$, so that the $O(\sqrt{\log n})$ approximation algorithm for Sparsest Cut takes $\tilde{O}(n^2)$ time overall.

Similarly, for c-Balanced Separator, by removing minimum cuts recursively as long as the total size of the removed subgraph is at most $c'n$, we can obtain a factor $O(n)$ approximation to $\alpha_{c'}(G)$, as shown by Leighton and Rao [21]. Since we may have to aggregate $O(n)$ minimum cuts, the total amount of time needed to obtain the $O(n)$ approximation is $O(mn) = \tilde{O}(n^2)$. Thus, the $O(\sqrt{\log n})$ pseudo-approximation algorithm for the c-Balanced Separator takes $\tilde{O}(n^2)$ time as well.

5 Implementing Oracle

In this section we prove Theorem 7. Let $\varepsilon_1, \varepsilon_2$ be suitably chosen small constants, set $\varepsilon = \min\{\varepsilon_1, \varepsilon_2\}$ for the truncated payoff. Recall that given the demand vector (i.e., distribution on row strategies) $\bar{d}$ we are only trying to build a response (i.e., pure column strategy) $\bar{x}$ such that the payoff $v(\bar{d}, \bar{x}) \geq 0$. When we fail to build such a response we have to use $\bar{d}$ to exhibit a pseudo-expander flow.

In each of the following cases, Oracle attempts to exploit certain characteristics of the demands to find a suitable $\bar{x}$, failing which, execution falls through to the next case. To facilitate the search, Oracle may temporarily neglect part of the demands, however, the final $\bar{x}$ it finds gives payoff ≥ 0 even with the original demands.

Case 1: Many large degrees

Sort the vertices in decreasing order by d_i.
Case 1a: If the largest $\varepsilon_1 \beta n$ degrees account for more than $\varepsilon_1 nD$ demand, then we can find a $\bar{x} \in \mathbf{P}$ with non-negative payoff by setting $s_i = 1/\varepsilon_1$ for all these vertices. Set $z_S = 2$ for any S with $n/2$ vertices. All other variables are 0. Since $d(S, \bar{S}) \leq \frac{1}{2}nd$, we have

$$v(\bar{d}, \bar{x}) \;\geq\; \varepsilon_1 nD \cdot (1/\varepsilon_1) - d(S, \bar{S}) \cdot 2 \;\geq\; nD - nD \;\geq\; 0.$$

Case 1b: Otherwise, Oracle modifies the demand graph. Vertices with the top $\varepsilon_1 \beta n$ degrees have their demands set to 0. The remaining degrees must be at most $\frac{D}{\beta}$: otherwise, the removed demands summed up to at least $\varepsilon_1 \beta n \cdot \frac{D}{\beta} = \varepsilon_1 nD$, which we assumed is not the case. The total demand discarded is at most $\varepsilon_1 nD$. Execution falls through to the next case.

Case 2: A large non-expanding cut

First, Oracle applies the Benczúr-Karger reduction to the (modified) demand graph to reduce it to a set of $O(n \log n)$ non-zero demands such that *all* cuts (in particular, degrees too) are approximately preserved. Let G_d be the demand graph obtained this way. Oracle runs the procedure FindLargeCut(G_d, $\frac{D}{\beta}$, $\frac{c}{2}$, $\frac{\beta^2}{2}$) (see Lemma 20 in Appendix A) This runs in $\tilde{O}(n^2)$ time since there are only $O(n \log n)$ non-zero demands in G_d.
Case 2a: Suppose it gives $\frac{c}{2}$-balanced cut S with expansion at most $\frac{\beta^2}{2} \cdot \frac{D}{\beta} = \frac{\beta}{2}D$. The demand discarded in Case 1 is at most $\varepsilon_1 nD \leq \frac{2\varepsilon_1}{c}|S|D = \frac{\beta}{2}D|S|$, if we set $\varepsilon_1 = \frac{\beta c}{4}$. Even including this discarded demand we have $d(S, \bar{S}) \leq \beta D|S|$. Oracle constructs a $\bar{x} \in \mathbf{P}$ with non-negative payoff by setting $z_S = n/\,|S|$, all $s_i = \beta$, all other variables are 0. The payoff is

$$v(\bar{d}, \bar{x}) \;\geq\; nD \cdot \beta - d(S, \bar{S}) \cdot (n/|S|) \;\geq\; \beta nD - \beta nD \;\geq\; 0.$$

Case 2b: Otherwise, FINDLARGECUT returns a graph on at least $(1-\frac{c}{2})n$ nodes such that the induced demand graph has expansion least $\frac{\beta^4}{32}\cdot\frac{D}{\beta}=\frac{\beta^3}{32}D$. The demand on the nodes left out is discarded. On the entire graph, all c-balanced cuts still expand by at least $\beta_1 D$ for $\beta_1=\frac{\beta^3}{64}$. Execution falls through to the next case.

Case 3: Unroutable demands

First, ORACLE performs random sampling on the demands so that the number of nonzero demands is $\tilde{O}(n)$. In Appendix C, we prove the following lemma via simple applications of the Chernoff-Hoeffding bounds:

LEMMA 12
We can randomly sample the demands to produce new demands, $\tilde{d}_{ij}$, of which at most $O(n\log^2 n)$ are non-zero, so that for any $\delta>0$, with probability at least $1-n^{-\Omega(\log n)}$, we have:

$$\begin{aligned}
\forall i: &\quad \tilde{d}_i \;\le\; d_i + D\\
\forall S,\ n/2\ge|S|\ge cn: &\quad \tilde{d}(S,\bar{S}) \;\ge\; (1-\delta)d(S,\bar{S})\\
\forall \bar{x}\in\mathbf{P}: &\quad \sum_{ij} d_{ij}l_{ij}(\bar{x}) \;<\; nD \implies \sum_{ij}\tilde{d}_{ij}l_{ij}(\bar{x}) \;<\; 7nD
\end{aligned}$$

Now, ORACLE sets all $s_i=0$, and since $l_{ij}(\bar{x})$'s are truncated, the optimum choice of w_e's corresponds to solving the following LP:

$$\begin{aligned}
&\text{Maximize}\ \sum_{ij}\tilde{d}_{ij}l_{ij}\ \text{ subject to}\\
\forall ij,\forall p\in\mathcal{P}_{ij}: &\quad l_{ij} \;\le\; \sum_{e\in p} w_e\\
\forall ij: &\quad l_{ij} \;\le\; 1/\varepsilon_2\\
&\quad \sum_e c_e w_e \;\le\; \beta nD
\end{aligned} \tag{9}$$

We show how to approximately solve the above LP by considering the dual. This can be thought of as a min-cost max-concurrent flow problem, which can be solved in sparse graphs in $\tilde{O}(n^2)$ time using the algorithm of Fleischer [13]. Consider the following instance of a min-cost max-concurrent flow problem: for every pair $\{i,j\}$ we associate demand $\tilde{d}_{ij}$. We also associate a pseudo-edge between every pair $\{i,j\}$ with infinite capacity and cost $b=1/(\beta\varepsilon_2 nD)$. Any real edge e has cost 0 and its original capacity c_e. We impose the budget constraint 1 on the total cost of the flow. We get the following LP and its dual:

$$\begin{array}{ll}
\max t & \min \sum_e c_e w'_e + \phi'\\
\forall ij: \sum_{p\in\mathcal{P}_{ij}} y'_p + t'_{ij} \ge \tilde{d}_{ij}t & \forall ij:\ l'_{ij}\le b\phi'\\
\forall e: \sum_{p\ni e} y'_p \le c_e & \forall ij,\forall p\in\mathcal{P}_{ij}:\ l'_{ij}\le\sum_{e\in p} w'_e\\
b\sum_{ij} t'_{ij}\le 1 & \sum_{ij}\tilde{d}_{ij}l'_{ij}\ge 1
\end{array} \tag{10}$$

ORACLE solves this LP using Fleischer's algorithm to within a constant multiplicative factor, say 2. The algorithm runs in $\tilde{O}(n^2)$ time since there are only $O(n\log^2 n)$ non-zero demands.

The algorithm also yields the weights w_e such that $2t\ge\sum_e c_e w'_e+\phi'$. We also get a flow y_p with congestion C by setting $C=1/t$ and scaling all y'_p and t'_{ij} by C to get y_p and t_{ij}. This routes all but $\sum_{ij} t_{ij}$ of the demands with congestion C.

Next, we get a feasible solution w_e and l_{ij} for LP (9): let $k=\beta nD/(\sum_e c_e w'_e+\phi')\ge\beta nD\cdot C/2$, and scale up the w'_e, l'_{ij}, ϕ' by k to get w_e, l_{ij}, ϕ. Since $\sum_e c_e w_e+\phi=\beta nD$, $\sum_e c_e w_e\le\beta nD$ and $\phi\le\beta nD$; so

$b\phi \leq 1/\varepsilon_2$ as needed. Also, $\sum_{ij} \tilde{d}_{ij} l_{ij} = \sum_{ij} \tilde{d}_{ij} l'_{ij} \cdot k \geq \beta n DC/2$.

Case 3a: If $C > 14/\beta$ then $\sum_{ij} \tilde{d}_{ij} l_{ij} > 7nD$, so $\sum_{ij} d_{ij} l_{ij} \geq nD$. Then ORACLE constructs an $\bar{x} \in \mathbf{P}$ by using the given settings for w_e and l_{ij} and assigning $z_S = 2$ for some S with $n/2$ vertices. Other variables are all 0. Then

$$v(\bar{d}, \bar{x}) \;\geq\; nD - d(S, \bar{S}) \cdot 2 \;\geq\; nD - nD \;\geq\; 0$$

Case 3b: Otherwise $C \leq 14/\beta$, then the ORACLE fails. We then get a pseudo-expander flow as explained below.

5.1 Finding a Pseudo-Expander flow

The flow y_p obtained by ORACLE just before it failed routes all but $\sum_{ij} t_{ij}$ of the total demand with congestion at most $14/\beta$. We discard the t_{ij} demands, which amount to at most $\sum_{ij} t'_{ij}/t \leq (1/b) \cdot C \leq 14\varepsilon_2 nD$. The remaining demands are $\frac{D}{\beta} + D$ regular. If we choose $\delta = \frac{1}{2}$ in Lemma 12, then after random sampling, all c-balanced cuts have expansion at least $\frac{\beta_1}{2} D$. By setting $\varepsilon_2 = \frac{\beta_1 c}{56}$, the total demand discarded is at most $\frac{\beta_1 c}{4} nD$. Thus, all c-balanced cuts still have expansion at least $\frac{\beta_1}{4} D$. We then scale the flow by $\frac{\beta}{14}$ so that the congestion becomes 1, all degrees are at most D, and all c-balanced cuts have expansion at least $\beta_2 D$ for $\beta_2 = \frac{\beta_1 \beta}{56}$. Thus, we end up with a D-regular (c, β_2)-pseudo-expander flow.

6 Finding a cut of expansion within $O(\sqrt{\log n})$ of optimal

In this section we finish the proof of Theorem 10 (as noted earlier, Theorem 9 follows using Lemma 11). Recall that the idea is to do binary search on degree D and try to find a D-regular pseudo-expander flow by solving the game. We already proved in the previous section that if the oracle fails within $O(\log n)$ rounds, then we can find a D-regular (c, β_2)-pseudo-expander flow.

Thus to finish the proof of Theorem 9 we show that if ORACLE does not fail in $T = O(\log n)$ rounds (in particular, no D-regular β_0-expander flow exists, as proved in Lemma 8) then we can find a c'-balanced cut of expansion $O(\sqrt{\log n} \cdot D)$, for some $c' \leq c$. Let $\bar{x}^* = \frac{1}{T} \sum_t \bar{x}^t$ be the vector obtained by averaging ORACLE's responses for all T rounds. Let its elements be s_i^*, w_e^*, z_S^*. Then we have the following lemmas.

LEMMA 13
For any $\bar{d}$, $v(\bar{d}, \bar{x}^*) \geq -\frac{\delta}{2} nD$

PROOF: We show $v(\bar{d}, \bar{x}^*) \geq \frac{1}{T} \sum_t v(\bar{d}, \bar{x}^t)$. Then the lemma follows from Lemma 8, because for all t, the ORACLE ensures that the payoff $v(\bar{d}, \bar{x}^t) \geq 0$. The only non-linear part in $v(\bar{d}, \bar{x})$ is $l_{ij}(\bar{x})$, so it suffices to prove that $l_{ij}(\bar{x}^*) \geq \frac{1}{T} \sum_t l_{ij}(\bar{x}^t)$. If $l_{ij}(\bar{x}^*) = 1/\varepsilon$, then since $1/\varepsilon$ is an upper bound on all $l_{ij}(\bar{x}^t)$, the inequality holds trivially. Otherwise, $l_{ij}(\bar{x}^*)$ is the length of shortest path from i to j under the corresponding w_e's. Let this path be p. The length of p under any $\bar{x}^t$ is at least $l_{ij}(\bar{x}^t)$. Averaging the lengths of p under all $\bar{x}^t$ we get exactly $l_{ij}(\bar{x}^*)$, which is thus at least $\frac{1}{T} \sum_{ij} l_{ij}(\bar{x}^t)$. □

LEMMA 14
We can construct a unit vectors $v_1, v_2, \ldots, v_n$ such that $\frac{1}{4} \sum_{ij} \|v_i - v_j\|^2 = c(1-c)n^2$ and for all pairs i, j, $\frac{1}{4} \|v_i - v_j\|^2 = \sum_{S:\ i \in S, j \in \bar{S}} \frac{|S|}{n} z_S^*$

PROOF: First we note that $\sum_S \frac{|S|}{n} z_S = 1$ for any $\bar{x} \in \mathbf{P}$. There are $N = O(\log n)$ sets S with non-zero z_S^*. We construct vectors in $\mathbb{R}^N$, with a coordinate for each such set S. For any vertex i, construct vector v_i by setting $v_i(S) = \pm\sqrt{\frac{|S|}{n} z_S^*}$ depending on whether $i \in S$ or $i \in \bar{S}$. Note that $\|v_i\|^2 = \sum_S \frac{|S|}{n} z_S = 1$. Also, for any pair i, j, the vector $v_i - v_j$ has non-zero coordinates only for S such that $i \in S, j \in \bar{S}$. Thus, $\frac{1}{4} \|v_i - v_j\|^2 = \sum_{S:\ i \in S, j \in \bar{S}} \frac{|S|}{n} z_S^*$. So,

$$\frac{1}{4} \sum_{ij} \|v_i - v_j\|^2 \;=\; \sum_{ij} \sum_{S:\ i \in S, j \in \bar{S}} \frac{|S|}{n} z_S^* \;=\; \sum_S \frac{|S|}{n} z_S^* \cdot \left[\sum_{ij:\ i \in S, j \in \bar{S}} 1 \right] \;=\; \sum_S \frac{|S|}{n} z_S^* \cdot |S||\bar{S}|.$$

Since $z_S^* \neq 0$ only if the cut $(S, \bar{S})$ is c-balanced, we have $|S||\bar{S}| \geq c(1-c)n^2$ for all such S, and hence

$$\frac{1}{4}\sum_{ij} \|v_i - v_j\|^2 \;\geq\; \sum_S \frac{|S|}{n} z_S^* \cdot c(1-c)n^2 \;=\; c(1-c)n^2.$$

□

We need the following Theorem from [5]:

THEOREM 15 ([5])
Let $v_1, v_2, \ldots, v_n$ be vectors of length at most 1, such that $\frac{1}{4}\sum_{ij}\|v_i - v_j\|^2 \geq c(1-c)n^2$. Let w_e be weights on edges and nodes and let $\alpha := \sum_e c_e w_e$. Then there is an algorithm which runs in $\tilde{O}(mn)$ time, and finds a cut of value C which is c'-balanced for some constant $c' \leq c$, such that there exists a pair of nodes i, j with the property that the graph distance between i and j is at most $O(\sqrt{\log n} \cdot \frac{\alpha}{C})$ and $\|v_i - v_j\|^2 \geq s$ where s is a constant. Furthermore, this is true even if any fixed set of τn nodes are prohibited from being i or j, for some small constant τ. The constants c', s, τ only depend on c.

THEOREM 16
If ORACLE *fails then we can find a cut with expansion at most $O(\sqrt{\log n} \cdot D)$ in $\tilde{O}(n^2)$ time.*

PROOF: Since for any $\bar{d}$, $v(\bar{d}, \bar{x}^*) \geq -\frac{\delta}{2}nD$, in particular, for any pair $\{i, j\}$, if we choose the demands $d_{ij} = \frac{1}{2}nD$ and $d_{k\ell} = 0$ if $\{k, \ell\} \neq \{i, j\}$, we conclude that

$$s_i^* + s_j^* + l_{ij}(\bar{x}^*) - \sum_{S:\; i \in S, j \in \bar{S}} z_S^* \geq -\delta$$

$$\Longrightarrow\; s_i^* + s_j^* + \min_{p \in \mathcal{P}_{ij}} \left\{ \sum_{e \in p} w_e^* \right\} \geq \sum_{S:\; i \in S, j \in \bar{S}} \frac{|S|}{n} z_S^* - \delta$$

For convenience, let $\Delta(i, j) = \min_{p \in \mathcal{P}_{ij}} \{\sum_{e \in p} w_e^*\}$. Construct the unit vectors $v_1, v_2, \ldots, v_n$ of Lemma 14. Then for all pairs $\{i, j\}$, we have $s_i^* + s_j^* + \Delta(i, j) \geq \frac{1}{4}\|v_i - v_j\|^2 - \delta$. Let c', s, τ be the constants given by Theorem 15. Note that $\alpha = \sum_e c_e w_e^* \leq \beta n D$.

Since $\sum_i s_i^* \leq \beta n$, at most τn nodes have $s_i^* > \beta/\tau$. Let all such vertices form the set A. We apply Theorem 15 to G with A being the τn forbidden vertices. We thus get a cut of value C such that there is a pair of vertices i, j with the following properties:

1. $s_i^*, s_j^* \leq \beta/\tau$,
2. the graph distance of i, j in G is at most $O(\sqrt{\log n} \cdot \alpha/C) = O(\sqrt{\log n} \cdot nD/C)$, and
3. $\|v_i - v_j\|^2 \geq s$.

Further, we choose $\beta = \frac{s\tau}{32}$ and $\delta = \frac{s}{16}$ so that $\frac{s}{4} - \frac{2\beta}{\tau} - \delta = \frac{s}{8}$. Then we conclude that $\Delta(i, j) \geq \frac{s}{8}$. Hence, we have $O(\sqrt{\log n} \cdot nD/C) \geq \frac{s}{8}$. Thus $\frac{C}{c'n} \leq O(\sqrt{\log n} \cdot D)$. This implies that the expansion of the cut found is at most $O(\sqrt{\log n} \cdot D)$, as required. Since we have only $O(n \log n)$ edges in the graph, this procedure runs in $\tilde{O}(n^2)$ time. □

7 Conclusions

Though our approximation ratio is $O(\sqrt{\log n})$, the constants inside the $O(\cdot)$ are not great. The game's definition uses a constant β_0 that is unspecified in [5] but is likely to be quite small, less than 0.1. The use of the Alon-Cheeger inequality degrades the constants further. We plan to explore whether the constants can be improved — either theoretically or in practice.

Though our running time of $\tilde{O}(n^2)$ seems tough to improve (in particular it arises in several places in the paper), one could conceivably get $\tilde{O}(m)$, that is, near-linear. This would involve looking inside the various results such as Benczúr-Karger and Fleischer (or Garg-Könemann) that are used in black-box fashion here. A challenging open problem is to obtain a poly-logarithmic approximation to the SPARSEST CUT problem in near-linear time. Currently, the fastest such algorithms appear in [4] and [25] and obtain a $O(\log n)$ approximation, in $\tilde{O}(m + n^{1.5})$ time.

Acknowledgements

We thank Russell Impagliazzo, Lisa Fleischer, Satish Rao, and Robert Schapire for helpful conversations.

References

[1] F. Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM J. Optim.*, 5(1):13–51, 1995.

[2] N. Alon. Eigenvalues and expanders. *Combinatorica*, 6(2):83–96, 1986.

[3] N. Alon and V. Milman. Lambda$_1$, isoperimetric inequalities for graphs, and superconcentrators. *J. Comb. Theory, Ser. B*, 38(1):73–88, 1985.

[4] S. Arora and S. Kale. A combinatorial, primal-dual approach to semidefinite programs. In *STOC*, pages 227–236, 2007.

[5] S. Arora, S. Rao, and U. V. Vazirani. Expander flows, geometric embeddings and graph partitioning. In *STOC*, pages 222–231, 2004.

[6] Sanjeev Arora, Elad Hazan, and Satyen Kale. Multiplicative weights method: a meta-algorithm and its applications. (a survey).

[7] Y. Aumann and Y. Rabani. An $O(\log n)$ approximate min-cut max-flow theorem and approximation algorithm. *SIAM J. Comput.*, 27(1):291–301, 1998.

[8] A. A. Benczúr and D. R. Karger. Approximating $s-t$ minimum cuts in $\tilde{O}(n^2)$ time. In *STOC*, pages 47–55, 1996.

[9] M. Blum, R. M. Karp, O. Vornberger, C. H. Papadimitriou, and M. Yannakakis. The complexity of testing whether a graph is a superconcentrator. *Inf. Process. Lett.*, 13(4/5):164–167, 1981.

[10] S. Chawla, A. Gupta, and H. Räcke. Embeddings of negative-type metrics and an improved approximation to generalized sparsest cut. In *SODA*, pages 102–111, 2005.

[11] J. Cheeger. A lower bound for the smallest eigenvalue of the laplacian. *Problems in Analysis*, pages 195–199, 1970.

[12] F. Chung. *Spectral graph theory*. Number 92 in CBMS Regional Conference Series in Mathematics. AMS, 1997.

[13] L. Fleischer. Approximating fractional multicommodity flow independent of the number of commodities. *SIAM J. Discrete Math.*, 13(4):505–520, 2000.

[14] Y. Freund and R. E. Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29:79–103, 1999.

[15] N. Garg and J. Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In *FOCS*, pages 300–309, Los Alamitos, CA, November8–11 1998. IEEE Computer Society.

[16] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, 1995.

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

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

[19] P. N. Klein, S. A. Plotkin, C. Stein, and É. Tardos. Faster approximation algorithms for the unit capacity concurrent flow problem with applications to routing and finding sparse cuts. *SIAM J. Comput.*, 23(3):466–487, 1994.

[20] J. Kuczynski and H. Wozniakowski. Estimating the largest eigenvalue by the power and lanczos algorithms with a random start. *SIAM J. Matrix Anal. Appl.*, 13(4):1094–1122, 1992.

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

[22] N. Linial, E. London, and Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 1995.

[23] D. W. Matula. A linear time $2 + \varepsilon$ approximation algorithm for edge connectivity. In *SODA*, pages 500–504, 1993.

[24] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

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

[26] S. A. Plotkin, D. B. Shmoys, and É. Tardos. Fast approximation algorithms for fractional packing and covering problems. In *FOCS*, pages 495–504, 1991.

[27] F. Shahrokhi and D. W. Matula. The maximum concurrent flow problem. *J. ACM*, 37(2):318–334, 1990.

[28] D. S. Shmoys. Cut problems and their application to divide and conquer. *Approximation Algorithms for NP-hard problems*, 1995.

[29] V. Vazirani. *Approximation algorithms.* Springer Verlag, 2002.

[30] N. E. Young. Randomized rounding without solving the linear program. In *SODA*, pages 170–178, 1995.

A Sparse cuts via eigenvalue computations

For a weighted graph G where the weight for pair $\{i, j\}$ (j could possibly the same as i, to allow for self-loops) is d_{ij}, the Laplacian $\mathcal{L}$ of G is the $n \times n$ symmetric matrix with rows and columns indexed by nodes in G, such that $\mathcal{L} = D^{-1/2}(D - A)D^{-1/2}$, where D is the diagonal matrix of (weighted) node degrees, and A is the (weighted) adjacency matrix of the graph G [3]. The smallest eigenvalue of $\mathcal{L}$ is 0 corresponding to the eigenvector $\langle\sqrt{d_1}, \sqrt{d_2}, \ldots, \sqrt{d_n}\rangle^\top$. The following well-known theorem that arises from the work of Alon and Cheeger (for a proof see [12]) shows that the second smallest eigenvalue of $\mathcal{L}$ gives us useful information about the conductance of G:

Theorem 17 (Alon-Cheeger)
Let the conductance of a weighted graph G with n vertices and m edges be $c(G)$. Let the Laplacian of the graph be $\mathcal{L}$, and denote its second smallest eigenvalue by $\lambda_{\mathcal{L}}$. Then:

$$2c(G) \geq \lambda_{\mathcal{L}} \geq \frac{c(G)^2}{2}$$

Furthermore, suppose we are given a vector x such that $\sum_i \sqrt{d_i} x_i = 0$, where d_i is the degree of node i. Let $\lambda := \frac{x^\top \mathcal{L} x}{x^\top x}$. Then there is a procedure SweepCut(G, x), *that in time $\tilde{O}(m+n)$ finds a cut with conductance at most $\sqrt{2\lambda}$.*

[3] We assume here that no node has zero degree.

The procedure SweepCut(G, x) operates as follows. Assume that the coordinates of x are ordered in increasing value, $x_1 \le x_2 \le \dots x_n$. For $1 \le k \le n-1$, let $S_k = \{1, 2, \dots, k\}$. Then the theorem shows that one of the cuts $(S_k, \bar{S}_k)$ has conductance at most $\sqrt{2\lambda}$, and thus can be found in time $\tilde{O}(m+n)$.

The optimal x for this procedure is an eigenvector belonging to the second smallest eigenvalue of $\mathcal{L}$. Computing this eigenvector may be computationally expensive if $\lambda_{\mathcal{L}}$ is very close to 0. However, for our application, we only need to find a cut of conductance less than some pre-specified constant $\beta > 0$. In this case, it suffices to find a vector x such that its λ value is at most $\frac{\beta^2}{2}$. For this, we can use the power method, as explained in the following lemma.

Lemma 18
Let $\lambda > \lambda_{\mathcal{L}}$ be a given parameter. Then we can find a vector x such that $\sum_i \sqrt{d_i} x_i = 0$ and $\frac{x^\top \mathcal{L} x}{x^\top x} \le \lambda$ using $O(\frac{\log n}{\lambda - \lambda_L})$ matrix vector products with the matrix $\mathcal{L}$.

Proof: An eigenvector of $\mathcal{L}$ belonging to the 0 eigenvalue is $y = \langle \sqrt{d_1}, \sqrt{d_2}, \dots, \sqrt{d_n} \rangle^\top$. Furthermore, the largest eigenvalue of $\mathcal{L}$ is at most 2. Thus, the matrix $2I - \mathcal{L} - \frac{1}{y^\top y} y y^\top$ is positive semidefinite with largest eigenvalue $2 - \lambda_{\mathcal{L}}$. To get a vector x with Rayleigh quotient at least $2 - \lambda$, we can use the power method with a random start. The analysis of [20] indicates that the method succeeds with constant probability in $O(\frac{\log n}{\lambda - \lambda_L})$ iterations. □

Lemma 19
There is a randomized procedure, FindCut(G, λ), *which finds a cut of conductance at most $\sqrt{2\lambda}$ in G, or with constant probability, concludes correctly that G has conductance at least $\frac{\lambda}{4}$. The procedure requires $O(\frac{\log n}{\lambda})$ matrix vector products with the Laplacian of G.*

Proof: Let $\lambda_{\mathcal{L}}$ be the second smallest eigenvalue of the Laplacian of G. If $\lambda_{\mathcal{L}} \le \frac{1}{2}\lambda$, then with constant probability, $O(\frac{\log n}{\lambda})$ iterations of the power method, as described in Lemma 18, suffice to find the desired x. Once x is found, we can run SweepCut(G, x) to find a cut of conductance at most $\sqrt{2\lambda}$. Otherwise if $\lambda_{\mathcal{L}} \ge \frac{1}{2}\lambda$, then by Theorem 17, G has conductance at least $\frac{1}{2}\lambda$. □

One can iterate the FindCut procedure to find c-balanced separators:

Lemma 20
There is a procedure that, given a weighted graph G with degrees bounded by D, a fraction $c > 0$, and a expansion bound $\beta > 0$, uses FindCut *$O(n)$ times and produces either a cut of size cn with edge expansion less than βD, or a graph on at least $(1-c)n$ vertices on which every cut has expansion at least $\frac{\beta^2}{8} D$.*

Proof: For a graph G, let G_D be the graph G with each node augmented with (weighted) self-loops to make the weighted degree exactly D. This ensures that a cut of conductance ϕ in G_D has expansion at least ϕD. We repeatedly use FindCut to get cuts of expansion less than βD and aggregate them. The resulting set also has expansion less than βD. When FindCut can no longer find cuts of expansion less than βD, then with constant probability, the second smallest eigenvalue of the Laplacian of the remaining (augmented) graph is at least $\frac{\beta^2}{4}$. Thus, by Theorem 17, every cut in the remaining graph has expansion $\frac{\beta^2}{8} D$. This procedure, FindLargeCut, is given below. The success probability can be boosted up using standard repetition techniques. Here, $G \setminus S$ is the subgraph induced on the vertex set $V \setminus S$.

Procedure FindLargeCut(G, D, c, β)
// Returns a cut of expansion of expansion at most βD or
// a subgraph of G of size at least $(1-c)n$ with expansion at least $\frac{\beta^2}{8} D$
Initialization: $S \leftarrow \phi$
while FindCut$((G \setminus S)_D, \frac{\beta^2}{2})$ finds a cut $(T, \bar{T})$ of conductance at most β
  $S \leftarrow S \cup T$
end while
if $|S| \ge cn$ **then** return the set S
else return the graph $(G \setminus S)$

□

B Pseudo-Expander Flows

In this section we show how, given a pseudo-expander flow, we can either convert it into an expander flow or find a sparse cut.

DEFINITION 3 *A D-regular multi-commodity flow (f_p) is called a (c,β)-***pseudo-expander flow** *if every c-balanced cut expands well. Formally:*

$$\forall S,\ n/2 \geq |S| \geq cn: \quad \sum_{i\in S, j\in \bar{S}} \sum_{p\in \mathcal{P}_{ij}} f_p \ \geq\ \beta D|S|$$

LEMMA 21
Let f_p be a D-regular (c,β)-pseudo-expander flow on a graph G. Assume that the flow has non-zero demand on only $O(n\log n)$ pairs of vertices. Then, there is a procedure that in time $\tilde{O}(n^2)$, finds either

1. *a D-regular $\frac{\beta^2}{130}$ expander flow,*
2. *or, a cut of expansion at most $\frac{1}{c}D$.*

PROOF: Let G_f be the demand graph for the given pseudo-expander flow. Let $\mathcal{D}$ be its Laplacian, and let $\lambda_{\mathcal{D}}$ be the second smallest eigenvalue of $\mathcal{D}$.

First, we run FINDLARGECUT$(G_f, D, c, \frac{\beta}{2})$. Since G_f is the demand graph of a (c,β)-pseudo-expander flow, the procedure cannot return a cut $(S,\bar{S})$ in G_f that is c balanced with expansion less than $\frac{\beta}{2}D$. Thus, it returns a subset of vertices S of size at most cn such that the induced subgraph $G_f \setminus S$ has expansion at least $\frac{\beta^2}{32}$.

Now, let L be the set S and $cn-|S|$ arbitrarily removed nodes of $\bar{S}$, and let $R = V\setminus L$. Note that $|L| = cn$ and $R = (1-c)n$. Let $k = \frac{|R|}{|L|} \leq \frac{1}{c}$. We form a flow network by connecting all nodes in L to a single (artificial) source with edges of capacity kD and all nodes in R to a single (artificial) sink with edges of capacity D, and compute the max flow in the network (with all original graph edges in G retaining their capacities). The flow computation runs in $\tilde{O}(n^2)$ time using the algorithm of Goldberg and Tarjan [17] since the graph is sparse.

Suppose the flow does not saturate all source edges. Then its value is less than $kD|L|$. Let $(T,\bar{T})$ be the min-cut found, with T being the side of the cut containing the source. Let $n_\ell = |T\cap L|$ and let $n_r = |\bar{T}\cap R|$. Then the capacity of the original graph edges cut is at most $kD|L| - kD(|L|-n_\ell) - D(|R|-n_r) = D[kn_\ell + n_r - |R|]$. The smaller side of the cut contains at least $\min\{n_\ell, n_r\}$ nodes, and since $kn_\ell \leq k|L| = |R|$ and $n_r \leq |R|$, the expansion of the cut found is at most $kD \leq \frac{1}{c}D$.

Otherwise, suppose that the flow does saturate all source edges. Let $\bar{g}$ be this flow. Consider the flow $\bar{h} = \frac{1}{2}\bar{f} + \frac{1}{2k}\bar{g}$. Then $\bar{h}$ is a D-regular flow. We claim that it is an $\Omega(c^2)$ expander flow.

Let $(T,\bar{T})$ be a cut in the graph, with $|T| \leq |\bar{T}|$. Let $x = |T\cap L|$ and $y = |T\cap R|$. Now we have two cases:

1. $x \geq \frac{\beta^2}{64}y$:
   Since $\bar{g}$ pumps kD flow into every node in $T\cap L$, which eventually goes into the sink, at least kDx flow must cross the cut $(T,\bar{T})$. Thus, in the flow $\bar{h}$, at least $\frac{1}{2}Dx$ flow crosses the cut $(T,\bar{T})$. Thus, the expansion of the cut $(T,\bar{T})$ is at least

   $$\frac{\frac{1}{2}Dx}{x+y} \ \geq\ \frac{\frac{1}{2}Dx}{x+\frac{64}{\beta^2}x} \ \geq\ \frac{\beta^2}{130}D$$

   since $\beta \leq 1$.

2. $x \leq \frac{\beta^2}{64}y$:
   Since $T\cap R \subseteq \bar{S}$, and since the subgraph of G_f induced by $\bar{S}$ has expansion at least $\frac{\beta^2}{32}D$, at least $\frac{\beta^2}{32}Dy$ flow in $\bar{f}$ must leave the set $T\cap R$. Since $\bar{f}$ is D-regular, at most Dx of this flow can terminate

in nodes in $T \cap L$. Thus, since $x \leq \frac{\beta^2}{64} y$, at least $\frac{\beta^2}{64} Dy$ flow crosses the cut $(T, \bar{T})$. So in $\bar{h}$, at least $\frac{\beta^2}{128} Dy$ flow crosses the cut $(T, \bar{T})$. Thus, the expansion of the cut $(T, \bar{T})$ is at least

$$\frac{\frac{\beta^2}{128} Dy}{x+y} \;\geq\; \frac{\frac{\beta^2}{128} Dy}{y + \frac{\beta^2}{64} y} \;\geq\; \frac{\beta^2}{130} D$$

since $\beta \leq 1$.

□

C Random sampling on the demands

We now describe how to randomly sample the demands to reduce the number of non-zero demands to $O(n \log^2 n)$ while still preserving degrees, expansion of large cuts, and the values of $l_{ij}(x)$'s.

Recall that we only perform the random sampling if the steps corresponding to choosing the s_i's and the z_S's do not result in a $\bar{x}$ that has positive payoff. Therefore:

$$\begin{aligned} \forall i: \quad & d_i \;\leq \frac{D}{\beta} \\ \forall S,\ n/2 \geq |S| \geq cn: \quad & d(S, \bar{S}) \;\geq\; \beta_1 D|S| \;\geq\; \beta_1 cnD \end{aligned}$$

For random sampling, we choose the probability distribution $\mathcal{D}$ over pairs of nodes, where the probability of $\{i,j\}$ is $p_{ij} = d_{ij}/Z$ where $Z = \sum_{k\ell} d_{k\ell} \leq \frac{1}{2} nD$. Now we form the multiset S by choosing m independent samples $\{i,j\}$ from $\mathcal{D}$. Thus, in each of the m rounds, we choose only 1 pair, for a total of m pairs. Set indicator random variable $X^s_{ij} = 1$ or 0 depending on whether we choose $\{i,j\}$ on the s^{th} trial, $1 \leq s \leq m$. Finally, set the new sampled demands to be

$$\tilde{d}_{ij} \;=\; \frac{Z \sum_s X^s_{ij}}{|S|}$$

We use the following version of the Chernoff-Hoeffding bounds from [24]:

LEMMA 22
Let $X = \sum X_s$ be the sum of m independent identically distributed random variables in $[0,1]$ such that $\mathbf{E}[X_s] = \mu$. Let $\delta > 0$ be any small error parameter , and $b(\delta) = (1+\delta)\ln(1+\delta) - \delta$. Then

$$\mathbf{Pr}[X/m \;\geq\; (1+\delta)\mu] \;<\; e^{-mb(\delta)\mu}.$$

If $\delta > 2e - 1$, the upper bound above can be replaced by $2^{-m(1+\delta)\mu}$. For $\delta < 1$, we have

$$\mathbf{Pr}[X/m \;\leq\; (1-\delta)\mu] \;<\; e^{-m\delta^2\mu/2}$$

Now we show the sampled demands approximate the original ones well with high probability, if the number of samples is $\Omega(n \log^2 n)$.

LEMMA 23
We can randomly sample the demands to produce new demands, $\tilde{d}_{ij}$, of which at most $O(n \log^2 n)$ are non-zero, so that for any $\delta > 0$, with probability at least $1 - n^{-\Omega(\log n)}$, we have:

$$\begin{aligned} \forall i: \quad & \tilde{d}_i \;\leq\; d_i + D \\ \forall S,\ n/2 \geq |S| \geq cn: \quad & \tilde{d}(S, \bar{S}) \;\geq\; (1-\delta) d(S, \bar{S}) \\ \forall \bar{x} \in \mathbf{P}: \quad & \sum_{ij} d_{ij} l_{ij}(\bar{x}) \;<\; nD \implies \sum_{ij} \tilde{d}_{ij} l_{ij}(\bar{x}) \;<\; 7nD \end{aligned}$$

PROOF: Let $m = \Omega(n \log^2 n)$ be the number of samples.

1. Fix any i. Define, for all $1 \le s \le m$, $X_s = \sum_j X_{ij}^s$. All $X_s \in \{0,1\}$ and have expectation d_i/Z. Define $X = \sum_s X_s$. Then $X/m = \tilde{d}_i/Z$. Set $\delta = \frac{D}{d_i}$. Now we have two cases:

   (a) $\delta \le 2e-1$:
   Then $d_i \ge \frac{D}{2e-1} \ge D/6$, and hence $d_i/Z \ge 1/3n$. By Lemma 22, we conclude that

   $$\mathbf{Pr}[\tilde{d}_i \ge d_i + D] \;=\; \mathbf{Pr}[X/m \;\ge\; (1+\delta)(d_i/Z)] \;<\; e^{-mb(\delta)d_i/Z} \;\le\; e^{-mb(\beta)/3n}$$

   since $\delta = \frac{D}{d_i} \ge \beta$, and so $b(\delta) \ge b(\beta)$.

   (b) $\delta \ge 2e-1$:
   Then $(1+\delta)d_i/Z > D/Z \ge 2/n$. By Lemma 22, we conclude that

   $$\mathbf{Pr}[\tilde{d}_i \ge d_i + D] \;=\; \mathbf{Pr}[X/m \;\ge\; (1+\delta)(d_i/Z)] \;<\; 2^{-m(1+\delta)d_i/Z} \;<\; 2^{-2m/n}.$$

   If $m = \Omega(n \log^2 n)$, then any such event happens with probability $< n^{-\Omega(\log n)}$. By the union bound over all n nodes,

   $$\mathbf{Pr}[\exists i:\; \tilde{d}_i \;\ge\; d_i + D] \;<\; n^{-\Omega(\log n)}.$$

2. Fix any S. Define, for all $1 \le s \le m$, $X_s = \sum_{i \in S, j \in \bar{S}} X_{ij}^s$. All $X_s \in \{0,1\}$ and have expectation $d(S,\bar{S})/Z$. Define $X = \sum_s X_s$. Then $X/m = \tilde{d}(S,\bar{S})/Z \ge 2\beta_1 c$. By Lemma 22 above, we conclude that

   $$\mathbf{Pr}[\tilde{d}(S,\bar{S}) \ge (1-\delta)d(S,\bar{S})] \;<\; e^{-m\delta^2 d(S,\bar{S})/2Z} \;\le\; e^{-\delta^2\beta_1 cm}.$$

   If $m = \Omega(n \log^2 n)$, then any such event happens with probability $< e^{-\Omega(n \log^2 n)}$. By the union bound over at most e^n choices of S,

   $$\mathbf{Pr}[\exists S,\; |S| > n/5:\; \tilde{d}(S,\bar{S}) \;\ge\; (1-\delta)d(S,\bar{S})] \;<\; e^{-\Omega(n \log^2 n)}$$

3. Fix an $\bar{x} \in \mathbf{P}$. Let w_e be the weight function on edges specified by $\bar{x}$. Note that since we truncate all path lengths to be at most $1/\varepsilon_2$, we may assume that all $w_e \le 1/\varepsilon_2$. We discretize the space of all possible weight functions on edges as follows. Let the number of edges be $N = O(n \log n)$. We round the w_e values downwards to the closest multiple of $1/N$, to obtain the point $\tilde{x} \in \mathbf{P}$. The number of possible such discretized weight functions is bounded by $(N/\varepsilon_2)^N = e^{O(n \log^2 n)}$.

   With some abuse of notation, let $l_{ij} = l_{ij}(\bar{x})$ and $\tilde{l}_{ij} = l_{ij}(\tilde{x})$. The discretization ensures that $|l_{ij} - \tilde{l}_{ij}| \le 1$. Since case 1. holds with high probability, we may assume that all $\tilde{d}_i \le d_i + D$. Thus, $|\sum_{ij} \tilde{d}_{ij} l_{ij} - \sum_{ij} \tilde{d}_{ij}\tilde{l}_{ij}| \le \sum_{ij} \tilde{d}_{ij} \le nD$.

   Define, for all $1 \le s \le m$, $X_s = \varepsilon_2 \sum_{ij} X_{ij}^s \tilde{l}_{ij}$. All $X_s \in [0,1]$ (as $\tilde{l}_{ij} \le \varepsilon_2^{-1}$) and have expectation $\mu = \varepsilon_2 \sum_{ij} d_{ij}\tilde{l}_{ij}/Z$. Define $X = \sum_s X_s$. Then $X/m = \varepsilon_2 \sum_{ij} \tilde{d}_{ij}\tilde{l}_{ij}/Z$. Now, if $\sum_{ij} d_{ij} l_{ij} < nD$, then $\sum_{ij} d_{ij}\tilde{l}_{ij} < nD$, and so $\mu < \varepsilon_2 nD/Z$. Let $1+\delta = \frac{6\varepsilon_2 nD/Z}{\mu}$. Note that $\delta > 2e-1$, and $(1+\delta)\mu \ge 12\varepsilon_2$. By Lemma 22 above, we conclude that,

   $$\mathbf{Pr}[\sum_{ij} \tilde{d}_{ij} l_{ij} > 7nD] \;\le\; \mathbf{Pr}[\sum_{ij} \tilde{d}_{ij}\tilde{l}_{ij} > 6nD] \;\le\; \mathbf{Pr}[X/m > (1+\delta)\mu] \;<\; 2^{-m(1+\delta)\mu} \;<\; 2^{-12\varepsilon_2 m}.$$

   Applying the union bound to all the $e^{O(n \log^2 n)}$ possible discretized metrics, we obtain that if $m = \Omega(n \log^2 n)$ then

   $$\mathbf{Pr}[\exists \bar{x} \in \mathbf{P}:\; \sum_{ij} d_{ij} l_{ij} < nD \text{ but } \sum_{ij} \tilde{d}_{ij} l_{ij} > 7nD] \;<\; 2^{O(n \log^2 n)} e^{-\Omega(n \log^2 n)} \;=\; e^{-\Omega(n \log^2 n)}.$$

Finally, the union bound over all three cases implies that the stipulation of the lemma holds with probability at least $1 - n^{-\Omega(\log n)}$. □

D Proof of Theorem 1

PROOF:(Theorem 1) We use the value $\Phi^{(t)} = \sum_{i \in \mathcal{R}} w_i{}^{(t)}$ as a potential function, and track changes in it as t increases. We have, for $t \geq 1$:

$$
\begin{aligned}
\Phi^{(t+1)} &= \sum_{i \in \mathcal{R}} w_i{}^{(t+1)} \\
&= \sum_{i \in \mathcal{R}} w_i{}^{(t)} \cdot (1-\varepsilon)^{\mu(i, j^{(t)})} \\
&\leq \sum_{i \in \mathcal{R}} w_i{}^{(t)} \cdot (1 - \varepsilon\mu(i, j^{(t)})) && \because (1-\varepsilon)^x \leq 1 - \varepsilon x \text{ for } x \in [0,1] \\
&= \Phi^{(t)}[1 - \varepsilon\mu(\mathcal{D}^{(t)}, j^{(t)})] && \because \mathcal{D}^{(t)} = \{w_1{}^{(t)}/\Phi^{(t)}, \ldots, w_N{}^{(t)}/\Phi^{(t)}\} \\
&\leq \Phi^{(t)} \exp(-\varepsilon\mu(\mathcal{D}^{(t)}, j^{(t)})) && \because (1-x) \leq \exp(-x)
\end{aligned}
$$

Thus, by induction, we get that

$$\Phi^{(T+1)} \leq \Phi^{(1)} \exp(-\varepsilon {\textstyle\sum_{t=1}^{T}} \mu(\mathcal{D}^{(t)}, j^{(t)})) = N \exp(-\varepsilon {\textstyle\sum_{t=1}^{T}} \mu(\mathcal{D}^{(t)}, j^{(t)})).$$

On the other hand, we have for any $i \in \mathcal{R}$,

$$\Phi^{(T+1)} \geq w_i{}^{(T+1)} = (1-\varepsilon)^{\sum_{t=1}^{T} \mu(i, j^{(t)})}.$$

Putting these together, and taking logarithms and simplifying using the fact that $-\ln(1-\varepsilon) \leq \varepsilon(1+\varepsilon)$ for $\varepsilon < \frac{1}{2}$, we get that for any $i \in \mathcal{R}$,

$$\sum_{t=1}^{T} \mu(\mathcal{D}^{(t)}, j^{(t)}) \leq (1+\varepsilon) \sum_{t=1}^{T} \mu(i, j^{(t)}) + \frac{\ln N}{\varepsilon}.$$

Since this holds for all $i \in \mathcal{R}$, given any distribution $\mathcal{D}$ over $\mathcal{R}$, by summing up the inequalities for all i with weights given by $\mathcal{D}$, we get that

$$\sum_{t=1}^{T} \mu(\mathcal{D}^{(t)}, j^{(t)}) \leq (1+\varepsilon) \sum_{t=1}^{T} \mu(\mathcal{D}, j^{(t)}) + \frac{\ln N}{\varepsilon}.$$

Substituting $\mu(\mathcal{D}^{(t)}, j^{(t)}) = (\mathbf{M}(\mathcal{D}^{(t)}, j^{(t)}) - \ell)/\rho$, we get

$$\frac{1}{T} \sum_{t=1}^{T} \mathbf{M}(\mathcal{D}^{(t)}, j^{(t)}) \leq \frac{1}{T} \sum_{t=1}^{T} [\mathbf{M}(\mathcal{D}, j^{(t)}) + \varepsilon(\mathbf{M}(\mathcal{D}, j^{(t)}) - \ell)] + \frac{\rho \ln N}{\varepsilon T}.$$

We bound $(\mathbf{M}(\mathcal{D}, q^t) - \ell) \leq \rho$, and then using the specified values of ε and T, we get (3).
□