

$O(\sqrt{\log n})$ approximation to SPARSEST CUT can be found in $\tilde{O}(n^2)$ time

FOCS 2004 Submission

Sanjeev Arora*

Elad Hazan †

Satyen Kale‡

April 21, 2004

Abstract

We show that the recent results for obtaining $O(\sqrt{\log n})$ -approximation to SPARSEST CUT and BALANCED SEPARATOR problems due to Arora, Rao, and Vazirani (2004) can be used to derive an $\tilde{O}(n^2)$ time approximation algorithm for an n -node graph. The previous best algorithm needed to solve a semidefinite program with $O(n^3)$ constraints.

Our algorithm relies on efficiently finding *expander flows* in the graph. The existence of these flows was established by [ARV].

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 approaches, 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 [26]).

The following are some of the basic problems in this class. In a graph $G = (V, E)$, for any cut (S, \bar{S}) where $|S| \leq |V|/2$, the *edge expansion* of the cut is $|E(S, \bar{S})|/|S|$. In the SPARSEST CUT problem we wish to determine the cut with the smallest edge expansion:

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

A cut (S, \bar{S}) is *c-balanced* if both S, \bar{S} have at least $c|V|$ vertices. In the 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

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

where $E(S)$ denotes the set of edges incident to nodes in S . The above definitions are modified in the obvious way for weighted graphs by using the capacity $c(S, \bar{S})$ of the cut instead of $|E(S, \bar{S})|$.

Efforts to efficiently find good approximate solutions to these NP-hard problems have led to radical progress in theoretical computer science. The earliest algorithms used spectral methods discovered—in the context of Riemannian manifolds—by Cheeger [8] and improved by Alon and Milman [3] and Alon [2].

*Computer Science Department, Princeton University. www.cs.princeton.edu/~arora. Supported by David and Lucile Packard Fellowship, NSF grants CCR 0205594 and CCR 0098180.

†Computer Science Department, Princeton University. www.cs.princeton.edu/~ehazan.

‡Computer Science Department, Princeton University. www.cs.princeton.edu/~satyen.

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.

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 c -BALANCED SEPARATOR. They used a linear programming relaxation of the problem based on multicommodity flow proposed in [25]. This led to approximation algorithms for numerous NP-hard problems, see the surveys [26, 28]. Furthermore, it sparked the emergence of several new fields in theoretical computer science, including fast computations of multicommodity flows and packing-covering linear programs [27, 24, 13], and efficient geometric embeddings of metric spaces [22]; see also [5].

Recently, Arora, Rao and Vazirani [4] showed how to compute a $O(\sqrt{\log n})$ -approximation to the above problems. They use semidefinite programming (SDP), a technique introduced in approximation algorithms by Goemans and Williamson [17]. The paper [4] introduces an intricate, geometric analysis of a particular SDP that uses triangle inequality constraints. The running time of their algorithm is dominated by the solution of this SDP, which takes $\tilde{O}(n^{4.5})$ time using interior point methods. (Here and in the rest of the paper \tilde{O} notation is used to suppress polylogarithmic factors.)

Arora, Rao, and Vazirani outlined an alternative approximation algorithm using *Expander flows*. These are multicommodity flows in the graph whose *demand graph* (i.e., the weighted graph (d_{ij}) where d_{ij} is the flow shipped between nodes i, j) is an expander. Of course, Leighton and Rao had shown how to embed even the complete graph (which in particular contains an expander) in the host graph, so the important issue here is the *edge congestion* (maximum amount of flow using an edge) of the flow. 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 $\Omega(\alpha(G)/\sqrt{\log |G|})$. (Note that determining graph expansion is coNP-complete [7], 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 [4] 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 new $\tilde{O}(n^2)$ -time 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 (Benczur and Karger [6], the last paper in a long line of work). Our algorithm computes an expander flow in a condensed sparse weighted graph that is obtained by Benczur 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 one can use the ideas of Arora-Rao-Vazirani to obtain the $O(\sqrt{\log n})$ -approximate sparsest cut from this metric.

The algorithm can also be used to find expander flows in any weighted graph on m edges in $\tilde{O}(m^2)$ time.

Overview of methodology

As mentioned, we first use the sparsification technique of Benczur-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.

Many papers have described efficient algorithms for packing and covering problems. The LP for finding expander flows has packing and covering constraints, but we found it difficult to fit it into conventional approaches such as [24]. We choose to go with an unconventional choice, the Freund-Schapire [12] method

for approximately solving a 2-person zero-sum game. It is likely that our results can be derived using more standard analyses, especially Young [27] or Garg-Könemann [13] (Young explicitly notes that his ideas can be applied to finding strategies for zero-sum games), but we find the Freund-Schapire setting more natural. We note that the existence proof for expander flows in [4] also used zero-sum games.

One has several choices how to represent expander flows in the zero-sum game, and several obvious ideas 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 the repeatedly playing these strategies causes them to converge to the true 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 Benczur-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 [4].

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. We assume that $\mathbf{M}(i, j) \in [\ell, u]$. Let $w = u - \ell$, this is called the *width* of the \mathbf{M} ; it will affect the running time of the algorithm below. Define $\mu(i, j) = (\mathbf{M}(i, j) - \ell)/w$, so $\mu(i, j) \in [0, 1]$.

The row player maintains a probability distribution \mathbf{P} over \mathcal{R} , which is initially uniform over \mathcal{R} and is refined as the game proceeds. Suppose the row player’s current distribution is \mathbf{P} and the column player — knowing \mathbf{P} — played the move $j \in \mathcal{C}$. Then expected payoff to the column player is $\sum_{i \in \mathcal{R}} \mathbf{P}(i) \mathbf{M}(i, j)$, which we denote by $\mathbf{M}(\mathbf{P}, j)$.

Let \mathbf{P}^t denote the distribution of the row player at the start of the t^{th} round (as mentioned, \mathbf{P}^1 is the uniform distribution, assigning probability $1/|\mathcal{R}|$ to each move) and the column player responded with $j^t \in \mathcal{C}$. Since the row player is trying to minimize the payoff to the column player, his next distribution \mathbf{P}^{t+1} tries to lower the probability assigned to strategies i which paid off more than the average strategy of \mathbf{P}^t . Specifically, the row player uses the following *multiplicative update rule*.

$$\mathbf{P}^{t+1}(i) \leftarrow \mathbf{P}^t(i) \cdot (1 - \epsilon)^{\mu(i, j^t)} / Z^t \quad (3)$$

where Z^t is the normalization factor, $Z^t = \sum_{i \in \mathcal{R}} \mathbf{P}^t(i) \cdot (1 - \epsilon)^{\mu(i, j^t)}$.

Freund and Schapire prove the next theorem, and for completeness we include the proof in Appendix C.

THEOREM 1

For any $\delta > 0$, let $\epsilon = \delta/2w$, $T = (4w^2 \ln N)/\delta^2$. Then for any distribution \mathbf{P} over \mathcal{R} ,

$$\frac{1}{T} \sum_{t=1}^T \mathbf{M}(\mathbf{P}^t, j^t) \leq \frac{1}{T} \sum_{t=1}^T \mathbf{M}(\mathbf{P}, j^t) + \delta \quad (4)$$

Suppose \mathbf{P}^* is the row player’s optimum distribution, namely, the \mathbf{P} that minimizes $\max_{j \in \mathcal{C}} \mathbf{M}(\mathbf{P}, j)$. Let $\lambda^* = \max_{j \in \mathcal{C}} \mathbf{M}(\mathbf{P}^*, j)$. By definition, $\max_{j \in \mathcal{C}} \mathbf{M}(\mathbf{P}, j) \geq \lambda^*$ for every \mathbf{P} . Thus we have:

COROLLARY 2

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

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* $d_{ij} \geq 0$ 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 while satisfying capacity constraints.

If the max-degree in the flow is d , then we say it is a β -*expander* if for all subsets S , the flow leaving S satisfies:

$$d(S, \bar{S}) = \sum_{i \in S, j \in \bar{S}} d_{ij} \geq \beta d |S|$$

Clearly, this is a lowerbound on the capacity of the cut (S, \bar{S}) in G , so if we can exhibit such a flow in G , it certifies that the $\alpha(G) \geq \beta d$. Note that a simple eigenvalue computation can check, given a multicommodity flow, whether or not the vector of demands is a $\Omega(1)$ -expander. Thus the flow in the next theorem can be used to certify that the expansion is at least $\alpha(G)/\sqrt{\log |V|}$.

THEOREM 3 ([4])

There is a constant $\beta_0 > 0$ such that every graph $G = (V, E)$ contains a d -regular β_0 -expander flow, where $d = \Omega(\alpha(G)/\sqrt{\log |V|})$.

REMARK 1 The theorem holds for multigraphs, and hence for capacitated graphs as well.

REMARK 2 Our algorithm will result in a flow that is a β -expander where β is some constant less than β_0 .

First we write an LP whose feasibility is implied by the above Theorem. (This LP is mentioned in the introduction of [4] and motivated the results of that paper.) Let d be the degree of interest; think of it as a “constant” in the LP, not as a variable. The capacity of edge e is denoted by c_e . Without loss of generality, by scaling, we assume that the minimum capacity is 1.

For all paths p in the graph, we have a non-negative variable f_p . Let \mathcal{P}_{ij} denote the set of simple paths connecting i, j . The PRIMAL LP is

$$\begin{aligned} \forall i : \sum_j \sum_{p \in \mathcal{P}_{ij}} f_p &\leq d \\ \forall e : \sum_{p: e \in p} f_p &\leq c_e \\ \forall S : \sum_{i \in S, j \in \bar{S}} \sum_{p \in \mathcal{P}_{ij}} f_p &\geq \beta_0 d |S| \end{aligned} \tag{5}$$

Note that we have relaxed the d -regularity condition to only require maximum degree d . This is without loss of generality; see [4]. The DUAL LP is as follows: we have non-negative variables s_i, w_e, z_S corresponding to vertex i , edge e , and subset $S \subseteq V$ respectively.

$$\begin{aligned} \min \quad & d \sum_i s_i + \sum_e c_e w_e - \beta_0 d \sum_S |S| z_S \quad \text{subject to} \\ \forall i, j, \forall p \in \mathcal{P}_{ij} : \quad & s_i + s_j + \sum_{e \in p} w_e - \sum_{S: i \in S, j \in \bar{S}} z_S \geq 0 \end{aligned} \tag{6}$$

The PRIMAL LP can be solved in polynomial time by an Ellipsoid-like method, as outlined in [4]. We wish to solve it more efficiently by associating a zero-sum game with it and using the Freund-Schapire framework. Clearly, 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 $\Omega(n^2m)$, which would be a lowerbound on the running time¹. 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.

Formally, we define a 2-person zero-sum game in which the row player's strategies correspond to node pairs $\{i, j\}$ and at each step he maintains some numbers d_{ij} satisfying $\sum_{i < j} d_{ij} = nd$. Thus $(\frac{d_{ij}}{nd})$ 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 responds with a pure strategy, which consists of picking dual variables $\bar{x} = \langle \bar{s}, \bar{w}, \bar{z} \rangle$ chosen from a polytope \mathbf{Q} , defined as

$$\mathbf{Q} \doteq \left\{ \langle \bar{s}, \bar{w}, \bar{z} \rangle : \sum_S |S| z_S = n; d \sum_i s_i + \sum_e c_e w_e \leq \beta nd \right\}. \quad (7)$$

Here $\beta \ll \beta_0$ is a small positive constant to be chosen later. The payoff for the pure row strategy $\{i, j\}$ and a column strategy $\bar{x} \in \mathbf{Q}$, 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} , define the *payoff function*

$$v(\bar{d}, \bar{x}) \doteq \sum_{ij} d_{ij} f_{ij}(\bar{x}) = \sum_i d_i s_i + \sum_{ij} d_{ij} l_{ij}(\bar{x}) - \sum_S d(S, \bar{S}) z_S \quad (8)$$

The payoff to column player for the mixed strategy $(\frac{d_{ij}}{nd})$ is $v(\bar{d}, \bar{x})/nd$.

It can be shown that if a d -regular β_0 -expander flow exists then the value of the game is $-(\beta_0 - \beta)nd$. 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 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 (8).

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

LEMMA 4

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

¹Note 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 update rule seems to require updating all variables, and indeed we update all n^2 demands at every iteration.

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 nd . Now for every $\bar{x} \in \mathbf{Q}$ we have:

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

The payoff, being negative, only goes down if we scale the demands to sum to exactly nd , so $\lambda^* \leq -(\beta_0 - \beta)nd$.

□

However, 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 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{R} be the polytope of $\bar{x} = \langle \bar{s}, \bar{w}, \bar{z} \rangle$ satisfying:

1. $s_i \leq 1/\varepsilon \quad \forall i$.
2. $z_S = 0$ whenever $|S| < n/10$.

Then the polytope corresponding to allowable pure strategies for the column player will be $\mathbf{P} = \mathbf{Q} \cap \mathbf{R}$, where \mathbf{Q} was defined in (7). Since $z_S > 0$ only for $|S| \geq n/10$, and $\sum_S |S| z_S = n$, we have $\sum_S z_S \leq 10$. Now from the definition of f_{ij} we see that for any $\bar{x} \in \mathbf{P}$, $-10 < f_{ij}(\bar{x}) < 3/\varepsilon$. Thus the width is $10 + 3/\varepsilon = O(1)$ and the Freund-Schapiro game played against a near-optimal column player converges in $T = O(\log n)$ iterations.

The final algorithm of Theorem 7 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 4 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 \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 the payoff to ORACLE is negative (which must happen close to convergence), then \bar{d} "almost" represents an expander flow; this is formalized in Section 4.1 using the notion of *pseudo-expander flow* (see Definition 1). A pseudo-expander flow can be used to find either an expander flow, or a cut of expansion $O(d)$.

THEOREM 5

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 $v(\bar{d}, \bar{x}) \geq 0$ in which exactly one set $S \subseteq V$ with at least $n/10$ 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-Schapiro procedure is run starting with the uniform demand vector where $d_{ij} = d/n$ for all i, j and using ORACLE as column player. (We are basically restating Theorem 1 and Lemma 4.) 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 6

Suppose ORACLE never failed up to round T . Then, for any set of demands \bar{d} , we have:

$$\frac{1}{T} \sum_{t=1}^T v(\bar{d}^t, \bar{x}^t) \leq \frac{1}{T} \sum_{t=1}^T v(\bar{d}, \bar{x}^t) + \delta nd \quad (9)$$

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

Now we are ready to state the main Theorem. This involves an algorithm that uses geometrically increasing values of d and tries to use the above ideas to produce a d -regular expander flow. If it fails to find such a flow it discovers a cut of expansion $O(d\sqrt{\log n})$ instead. If d is the first power of 2 for which this happened, then note that the algorithm has both a $d/2$ -regular expander flow and a cut of expansion $O(d\sqrt{\log n})$, and this implies that the cut is optimal upto $O(\sqrt{\log n})$ factor.

THEOREM 7 (MAIN)

There exists a procedure that given a graph G with edge expansion α , finds (i) a value d , (ii) a d -regular $\Omega(1)$ -expander flow and (iii) a cut of expansion $O(d\sqrt{\log n})$. Furthermore, this procedure runs in time $\tilde{O}(n^2)$.

REMARK 3 Note that (ii) implies $d = O(\alpha(G))$ and so, (iii) implies the cut has expansion $O(\alpha(G)\sqrt{\log n})$.

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]) 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/Alon we can do this using approximate eigenvalue computations on the Laplacian of this sparse graph, which takes $\tilde{O}(n)$ time by repeated powering. (This repeated powering idea has been repeatedly rediscovered, but one reference is [20]). Using the eigenvector and Theorem 16 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 $\tilde{O}(n^2)$.

Finally, the ORACLE procedure performs a min-cost concurrent multicommodity flow computation using the algorithm of Fleischer [11], which also takes time $\tilde{O}(n^2)$ since the number of edges has been reduced to $\tilde{O}(n)$ using Benczur-Karger.

4.1 Pseudo-expander flows

An expander flow corresponds to a demand vector that expands on all cuts. As pointed out in the proof of Theorem 3 in [4], it suffices to find a flow that is basically regular and expands on *balanced* cuts.

DEFINITION 1 *A d -regular multi-commodity flow with demands \bar{d} is called a (γ, β) -pseudo-expander flow if for every γ -balanced cut (S, \bar{S}) we have*

$$d(S, \bar{S}) \geq \beta d |S|. \quad (10)$$

Notice that a d -regular β -expander flow is in particular a d -regular (γ, β) -pseudo expander flow for each γ . In our algorithm, the pseudo-expander flows we produce will have only $\tilde{O}(n)$ non-zero demands. We prove two simple lemmas allowing us to go from a pseudo-expander-flow to an expander flow (and when we fail to so, to obtain a reason for the failure in form of a sparse cut).

LEMMA 8

There is a $\tilde{O}(n^2)$ -time algorithm that given a d -regular (γ, β) -pseudo-expander flow on a graph produces a set T of at least $(1 - \gamma)n$ vertices and a d -regular $\Omega(\beta^2)$ expander flow on the induced graph $G|_T$.

PROOF: We use the procedure FINDLARGECUT from Lemma 17 on the demand graph. Since the demand graph expands by βd on every set of size at least γn , the procedure must output a set T of size at least $(1 - \gamma)n$ such that the demands on T form a $\Omega(\beta^2)$ -expander flow on the induced graph $G|_T$. The procedure takes $\tilde{O}(n^2)$ time, as explained above. \square

LEMMA 9

There is a $\tilde{O}(n^2)$ -time algorithm that given a d -regular (γ, β) -pseudo-expander flow on a graph either produces a d -regular $\Omega(\beta^2)$ expander flow, or outputs a cut of expansion at most d .

PROOF: Use Lemma 8 to produce an $\Omega(\beta^2)$ expander flow on a large subgraph on a set $T \subseteq V$ of G . Then the rest of the proof is essentially from [4], who show how to use a single s - t flow computation to extend the flow to be an expander on all the vertices. Briefly, the idea is to set up a up a max s - t flow computation by injecting $k = |T|/|\bar{T}|$ units of flow into all vertices of \bar{T} , and sinking a unit of flow from every vertex of T . All edges have capacity k/d . If a max-flow exists of value $|T|$, then we can combine this flow (scaled by d/k) with the expander flow already existing on T to get an expander flow on the entire graph. Otherwise, there is a min-cut of value $< |T|$, and the intersection of this cut with \bar{T} gives a cut with expansion at most d . The flow computation runs in $\tilde{O}(n^2)$ time using the algorithm of Goldberg-Tarjan [14] since the graph is sparse. \square

5 Implementing ORACLE

In this section we prove Theorem 5. 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 \bar{d} we are only trying to build a response \bar{x} such that the payoff $v(\bar{d}, \bar{x}) \geq 0$. When we fail in that we have to use \bar{d} to exhibit a pseudo-expander flow. The search for \bar{x} goes through the following steps.

Picking the s_i 's

Sort the vertices in decreasing order by d_i . If the largest $\varepsilon_1 \beta n$ degrees account for more than $2\varepsilon_1 n d$ demand, then we can find a $\bar{x} \in \mathbf{P}$ with positive 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. Then

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

Otherwise we discard the top $\varepsilon_1 \beta n$ demands. Now, all the remaining demands are at most $2d/\beta$ - otherwise, the demands we removed are all at least $2d/\beta$, hence between them they account for at least $\varepsilon_1 \beta n \cdot 2d/\beta = 2\varepsilon_1 n d$ demand, which is not possible. Also, the total demand we discard is at most $2\varepsilon_1 n d$. We also discard all demands $< \varepsilon_1 d$. These amount to at most $\varepsilon_1 n d$ total. Then we proceed to the next strategy.

Picking the z_S 's

Assuming the setting of s_i 's didn't already achieve positive payoff, we try to achieve it using an appropriate choice of z_S 's. Let $\beta' = \beta - 30\varepsilon_1$. Next we apply the Benczur-Karger reduction to the 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. We make ε_1 so small that $\beta' \approx \beta$. We use procedure FINDLARGECUT from Lemma 17 with degree bound $2d/\beta$, $\gamma = 1/10$, and expansion bound $\beta\beta'/2$. This runs in $\tilde{O}(n^2)$ time. Suppose it gives $1/10$ -balanced cut S with expansion at most $\beta\beta' \cdot (2d/\beta) = \beta'd$. Even including all the $3\varepsilon_1 n d \leq 30\varepsilon_1 |S|d$ demand we discarded, we get $d(S, \bar{S}) \leq \beta d|S|$. Then we can find a $\bar{x} \in \mathbf{P}$ with positive payoff by setting $z_S = n/|S|$, all $s_i = \beta$, all other variables are 0. The payoff is

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

Otherwise, FINDLARGECUT returns a graph on at least $9n/10$ nodes such that demands on those nodes expand on any set by at least $\Omega((\beta'\beta)^2 \cdot (2d/\beta)) = \Omega(\beta^3 d)$. We discard the demand on the nodes left out. On the entire graph, all $1/5$ -balanced cuts still expand by at least $\Omega(\beta^3 d)$. We then proceed to the next strategy.

Picking the w_e 's

First we perform random sampling on the demands as outlined in appendix A so that the number of nonzero demands is $\tilde{O}(n)$. The sampled demands, denoted by \tilde{d}_{ij} , approximately preserve degrees, expansion of $1/5$ -balanced cuts, and have the additional property that for every $\bar{x} \in \mathbf{P}$, $\sum_{ij} \tilde{d}_{ij} l_{ij}(\bar{x}) > 20nd \Rightarrow$

$\sum_{ij} d_{ij} l_{ij}(\bar{x}) > 2nd$. We will set all $s_i = 0$, and since we truncated the $l_{ij}(\bar{x})$'s, the optimum choice of w_e 's corresponds to solving the following LP:

$$\begin{aligned}
& \text{Maximize } \sum_{ij} \tilde{d}_{ij} l_{ij} \text{ subject to} \\
& \forall ij, \forall p \in \mathcal{P}_{ij} : l_{ij} \leq \sum_{e \in p} w_e \\
& \forall ij : l_{ij} \leq 1/\varepsilon_2 \\
& \sum_e c_e w_e \leq \beta nd
\end{aligned} \tag{11}$$

We show how to approximately solve the above LP by considering the dual, which 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 [11]. 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 $D = 1/(\beta\varepsilon_2 nd)$. 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:

$$\begin{aligned}
& \text{Maximize } t \text{ subject to} & \text{Minimize } \sum_e c_e w'_e + \phi' \text{ subject to} \\
& \forall ij : \sum_{p \in \mathcal{P}_{ij}} y'_p + t'_{ij} \geq \tilde{d}_{ij} t & \forall ij, \forall p \in \mathcal{P}_{ij} : l'_{ij} \leq \sum_{e \in p} w'_e \\
& \forall e : \sum_{p \ni e} y'_p \leq c_e & \forall ij : l'_{ij} \leq D\phi' \\
& D \sum_{ij} t'_{ij} \leq 1 & \sum_{ij} \tilde{d}_{ij} l'_{ij} \geq 1
\end{aligned} \tag{12}$$

We solve it to some error parameter c , say $c = 1$. The algorithm also yields the weights w_e such that $(1+c)t = 2t \geq \sum_e c_e w'_e + \phi'$.

We now show how to convert these solutions to our problem: first, we 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} . With these settings, we route all but $\sum_{ij} t_{ij}$ of the demand with congestion C .

Next, we get w_e and l_{ij} for our original LP (11) as follows: set $s = \beta nd / (\sum_e c_e w'_e + \phi') \geq \beta nd \cdot C/2$, and scale up the w'_e, l'_{ij}, ϕ' by s to get w_e, l_{ij}, ϕ . Since $\sum_e c_e w_e + \phi = \beta nd$, so $\sum_e c_e w_e \leq \beta nd$ and $\phi \leq \beta nd$, so $D\phi \leq 1/\varepsilon_2$ as needed. Also, $\sum_{ij} \tilde{d}_{ij} l_{ij} = \sum_{ij} \tilde{d}_{ij} l'_{ij} \cdot s \geq \beta nd C/2$.

If $C > 40/\beta$ then $\sum_{ij} \tilde{d}_{ij} l_{ij} > 20nd$, so $\sum_{ij} d_{ij} l_{ij} > 2nd$. Then we get 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

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

Otherwise $C \leq 40/\beta$, 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 $40/\beta$. We discard the t_{ij} demands, which amount to at most $\sum_{ij} t'_{ij}/t \leq 1/D \cdot C \leq 40\varepsilon_2 nd$. The remaining demands are also $3d/\beta$ regular (where we are assuming that the random sampling may have increased degree from $2d/\beta$ by at most 1.5 times). Since all $1/5$ -balanced cuts expand by $\Omega(\beta^3 d)$ after random sampling, by making $\varepsilon_2 \ll \beta^3$, we can ensure that in spite of the demand thrown away all $1/5$ -balanced cuts still expand by $\Omega(\beta^3 d)$. We then scale the flow by $\beta/40$ so that the congestion becomes 1, all degrees are at most d , and all $1/5$ -balanced cuts expand by $\Omega(\beta^4 d)$. We get a d -regular $(\frac{1}{5}, \Omega(\beta^4))$ -pseudo-expander flow.

6 Finding an $O(\sqrt{\log n})$ -approximate sparsest cut

In this section we finish the proof of Theorem 7. Recall that the idea is to do binary search on degree d and try to find a d -regular 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 pseudo-expander flow. Using the ideas of Section 4.1, a pseudo-expander flow can either be extended to d -regular expander flow (with expansion at least $\Omega(\beta^4)$) or be used to find a cut of expansion $O(d)$.

Thus to finish the proof of Theorem 7 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 6) then we can find a cut of expansion $O(d\sqrt{\log n})$. 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 10

For any \bar{d} , $v(\bar{d}, \bar{x}^*) \geq -\delta nd$

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 6. The definition of $v(\bar{d}, \bar{x})$ shows that the only non-linear part in it 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 the average of all $l_{ij}(\bar{x}^t)$. \square

LEMMA 11

We can construct a unit- ℓ_2^2 -representation v_1, v_2, \dots, v_n that is $1/10$ -spread (cf. [4]) for some constant and which satisfies, for all 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 $\bar{x} \in \mathbf{P}$. Now, let there be N sets S with non-zero z_S^* (there are only $O(\log n)$ such). We construct vectors in \mathbb{R}^N , where we have a coordinate for each such set S . For any vertex i , we construct vector v_i by setting $v_i(S) = \pm \sqrt{\frac{|S|}{n} z_S^*}$ depending on whether $i \in S$ or $i \in \bar{S}$. These vectors are $1/10$ -spread because any set S with non-zero z_S^* is of size at least $n/10$. It is easily verified that all the other properties hold. \square

We need the following Theorem from [4]:

THEOREM 12 ([4])

For every $c < 1/4$ there are constants γ, C, τ such that the following is true. Given a graph G with a unit- ℓ_2^2 -representation v_1, v_2, \dots, v_n that is c -spread, we can find pair of vertices i, j and a cut of expansion α_{obs} such that their graph distance is $\leq C\sqrt{\log n}/\alpha_{\text{obs}}$ and $|v_i - v_j|^2 \geq \gamma$. Furthermore, this property holds even if we forbid some τn vertices from playing the role of i, j .

REMARK 4 (ARV) do not specify the running time of this procedure but it uses only breadth-first search and random sampling. When the vectors are in $\mathbb{R}^{O(\log n)}$, as ours are, the whole algorithm runs in $\tilde{O}(n^2)$ time.

THEOREM 13

We can find a cut with expansion at most $O(d\sqrt{\log n})$ in $\tilde{O}(n^2)$ time.

PROOF: Since for any \bar{d} , $v(\bar{d}, \bar{x}^*) \geq -\delta nd$, in particular, by concentrating all the nd on each pair ij , we conclude that

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

For convenience, let $\Delta(i, j) = \min_{p \in \mathcal{P}_{ij}} \left\{ \sum_{e \in p} w_e^* \right\}$. Construct the $1/10$ -spread unit- ℓ_2^2 -representation v_i of Lemma 11. Then we have for all ij , $s_i^* + s_j^* + \Delta(i, j) \geq \frac{1}{4}|v_i - v_j|^2 - \delta$. Let γ, C, τ be the constants given by Theorem 12 for $c = 1/10$. Let $W = \sum_e c_e w_e \leq \beta nd$.

Since $\sum_i s_i^* \leq \beta/n$, at most $\tau n/2$ nodes have $s_i^* > 2\beta/\tau$. Let all such vertices form set U . Next, we get a new unweighted graph G' by transforming every edge e into a path of length of $\lceil \frac{\epsilon c_e w_e n}{W} \rceil$, for some small constant $\epsilon < \tau/2$. By doing this, we introduce at most $\sum_e \lceil \frac{\epsilon c_e w_e n}{W} \rceil \leq \epsilon n$ new nodes. Let all new nodes form set V . These new nodes inherit a natural $1/10$ -spread unit- ℓ_2^2 -representation - for any edge $\{i, j\}$ in G , assign vector v_i for the first half of the vertices on the path joining i to j in G' and vector v_j for the latter.

We apply Theorem 12 to G' with $U \cup V$ being the τn forbidden vertices. We thus get a pair of vertices i, j and a cut with expansion α_{obs} such that (i) i, j are original vertices from G , (ii) $s_i^*, s_j^* \leq 2\beta/\tau$, (iii) the graph distance of i, j in G' is $\leq C\sqrt{\log n}/\alpha_{\text{obs}}$ and (iv) $|v_i - v_j|^2 \geq \gamma$. Note that the graph distance of i, j in G' is at most $\frac{\epsilon \Delta(i, j)n}{W}$, because $c_e \geq 1$. Further, let us choose β and δ to be small enough constants so that $\frac{1}{4}\gamma - 4\beta/\tau - \delta = \eta$ where $\eta > 0$ is a small constant. Then we conclude that $\Delta(i, j) \geq \eta$, and hence we have

$$\frac{C\sqrt{\log n}}{\alpha_{\text{obs}}} \geq \frac{\epsilon \eta n}{W} \implies \alpha_{\text{obs}} \leq O\left(\frac{\sqrt{\log n} W}{n}\right) = O(d\sqrt{\log n})$$

□

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 [4] but is likely to be 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 Benczur-Karger and Fleischer (or Garg-Könemann) that are used in black-box fashion here.

Acknowledgements

We thank Russell Impagliazzo, 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. Optimization*, **5**:13–51, 1995.
- [2] N. Alon. Eigenvalues and expanders. *Combinatorica* **6**:83–96, 1986.
- [3] N. Alon and V. Milman. λ_1 , isoperimetric inequalities for graphs and superconcentrators. *J. Combin. Theory B* **38**:73–88, 1985.
- [4] S. Arora, S. Rao and U. Vazirani. Expander flows, geometric embeddings, and graph partitioning. *To appear in ACM STOC 2004*.
- [5] Y. Aumann and Y. Rabani. An $O(\log k)$ approximate min-cut max-flow theorem and approximation algorithms. *SIAM J. Comp*
- [6] A. Benczur and D. Karger. Approximating $s - t$ Minimum Cuts in $\tilde{O}(n^2)$ Time, in *Proc. 28th Annual Symp. on Theory of Comp 1996*, pages 47–55.
- [7] M. Blum, R. Karp, O. Vornberger, C. Papadimitriou, M. Yannakakis. The complexity of testing whether a graph is a superconcentrator. *Inf. Proc. Letters* **13**:164-167, 1981.
- [8] J. Cheeger. A lower bound for the smallest eigenvalue of the Laplacian, in *Problem in Analysis*, 195-199, Princeton Univ. Press, (1970),

- [9] F. Chung. Spectral graph theory. *CBMS Regional Conference Series in Mathematics*, 92, American Mathematical Society, 1997.
- [10] U. Feige and R. Krauthgamer. A polylogarithmic approximation of the minimum bisection. In *IEEE FOCS 2001* pp 105–115.
- [11] L. Fleischer. Approximating Fractional Multicommodity Flows Independent of the Number of Commodities, *SIAM J. Discrete Math.* 13 (2000), no. 4, 505-520.
- [12] Y. Freund and R. Schapire. Adaptive game playing using multiplicative weights. In *Games and Economic Behavior 1999* pp 29:79–103.
- [13] N. Garg and J. Könemann. Faster and Simpler Algorithms for Multicommodity Flow and other Fractional Packing Problems. In *IEEE FOCS 1997*.
- [14] A. Goldberg and R. Tarjan. A new approach to the maximum flow problem. In *Journal of the ACM 1988*. pp 35:921–940
- [15] N. Garg and V. V. Vazirani and M. Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. *SIAM J. Computing*, **25**(2):235–251, 1996. *Prelim. version in Proc. ACM STOC'93*.
- [16] M.X. Goemans. Semidefinite programming in combinatorial optimization. *Math. Programming*, **79**:143–161, 1997.
- [17] M.X. Goemans and D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *JACM*, **42**(6):1115–1145, 1995.
- [18] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric algorithms and combinatorial optimization*. Springer-Verlag, 1993.
- [19] P. Klein and S. Plotkin and C. Stein and É. Tardos. Faster Approximation Algorithms for the Unit Capacity Concurrent Flow Problem With Applications to Routing and Finding Sparse Cuts *SIAM Journal on Computing* (1994), 466–487.
- [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.*, 15 (1994), pp. 672-691
- [21] T. Leighton and S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *JACM* **46** 1999. *Prelim. version in ACM STOC 1988*.
- [22] N. Linial, E. London, and Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica* **15**:215–246, 1995.
- [23] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University. 1995.
- [24] S. Plotkin and D. B. Shmoys and E. Tardos. Fast approximation algorithms for fractional packing and covering problems. *Math. Operations Res.* **20**:257-301, 1995. *Prelim. version IEEE Foundations of Computer Science, 1991, 495-504*.
- [25] F. Shahrokhi and D.W. Matula. The maximum concurrent flow problem. *Journal of the ACM*, 37:318–334, 1990.
- [26] D. S. Shmoys. Cut problems and their application to divide and conquer. *Approximation Algorithms for NP-hard problems*, D.S. Hochbaum (ed.), PWS Publishing, 1995.
- [27] N. Young. Randomized Rounding without Solving the Linear Program. *Sixth ACM-SIAM symposium on Discrete Algorithms*, 1995.
- [28] V. Vazirani. *Approximation algorithms*. Springer Verlag, 2002.

A 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:

$$\begin{aligned} \forall i : d_i &\geq \varepsilon_1 d \\ \forall S, |S| > n/5 : d(S, \bar{S}) &\geq \Omega(\beta^3 d |S|) \geq \Omega(\beta^3 \gamma n d) \end{aligned}$$

For the purposes of the random sampling, we scale down all demands by their sum so that they sum up to 1. Clearly, the degrees and expansion scale down by the same factor. We reuse notation and continue to refer to these demands as d_{ij} . Now choose the multiset S by sampling independently m times, $\{i, j\}$ with probability d_{ij} . Set indicator random variable $X_{ij}^s = 1$ 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

$$\tilde{d}_{ij} = \frac{\sum_s X_{ij}^s}{|S|}$$

We use the following version of the Chernoff-Hoeffding bounds from [23]:

LEMMA 14

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 $\delta < 1$ for the second bound), and $b(\delta) = (1 + \delta) \ln(1 + \delta) - \delta$. Then

$$\begin{aligned} \Pr[X/m \geq (1 + \delta)\mu] &< e^{-mb(\delta)\mu} \\ \Pr[X/m \leq (1 - \delta)\mu] &< e^{-m\delta^2\mu/2} \end{aligned}$$

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 15

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 the following properties hold with very high probability, for some small error parameter $\delta > 0$:

$$\begin{aligned} \forall i : \tilde{d}_i &\leq (1 + \delta)d_i \\ \forall S, |S| > n/5 : \tilde{d}(S, \bar{S}) &\geq (1 - \delta)d(S, \bar{S}) \\ \forall \bar{x} \in \mathbf{P} \sum_{ij} \tilde{d}_{ij} l_{ij}(\bar{x}) > 20 &\implies \sum_{ij} d_{ij} l_{ij}(\bar{x}) > 2 \end{aligned}$$

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

- Fix any i . Define, for all $1 \leq s \leq m$, $X_s = \sum_j X_{ij}^s$. All $X_s \in [0, 1]$ and have expectation d_i . Define $X = \sum_s X_s$. Then $X/m = \tilde{d}_i$. By Lemma 14 above, we conclude that

$$\Pr[\tilde{d}_i \geq (1 + \delta)d_i] < e^{-mb(\delta)d_i} \leq e^{-mb(\delta)\varepsilon_1/n}$$

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

$$\Pr[\exists i : \tilde{d}_i \geq (1 + \delta)d_i] < n^{-\Omega(\log n)}$$

- Fix any S . Define, for all $1 \leq s \leq m$, $X_s = \sum_{i \in S, j \in \bar{S}} X_{ij}^s$. All $X_s \in [0, 1]$ and have expectation $d(S, \bar{S})$. Define $X = \sum_s X_s$. Then $X/m = \tilde{d}(S, \bar{S})$. By Lemma 14 above, we conclude that

$$\Pr[\tilde{d}(S, \bar{S}) \geq (1 + \delta)d(S, \bar{S})] < e^{-m\delta^2 d(S, \bar{S})} \leq e^{-\Omega(m)}$$

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

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

- Fix an $\bar{x} \in \mathbf{P}$. With some abuse of notation, let $l_{ij} = l_{ij}(\bar{x})$. Define, for all $1 \leq s \leq m$, $X_s = \varepsilon_2 \sum_{ij} X_{ij}^s l_{ij}$. All $X_s \in [0, 1]$ (as $l_{ij} \leq \varepsilon_2^{-1}$) and have expectation $\mu = \varepsilon_2 \sum_{ij} d_{ij} l_{ij}$. Define $\bar{X} = \frac{1}{m} \sum_s X_s$. Then $\bar{X} = \varepsilon_2 \sum_{ij} \tilde{d}_{ij} l_{ij}$.

We discretize the space of all possible weight functions $\{w_e\}$. Round the w_e values downwards to a multiple of $n^{-\Omega(1)}$. Then the number of possible such metrics is bounded by $(n^{O(1)})^{O(n \log n)} = 2^{O(n \log^2 n)}$. Also, the difference in $\sum_{ij} d_{ij} l_{ij}$ value before and after discretization is at most $n^{-\Omega(1)}$.

To prove our claim, it suffices to show that if $\mu < 2\varepsilon_2$ then $\bar{X} < 20\varepsilon_2$ with high probability. Let $k = 20\varepsilon_2$, $1 + \Delta = \frac{k}{\mu} \geq 10$. By Lemma 14 above, we conclude that,

$$\Pr[\bar{X} \geq 20\varepsilon_2] = \Pr[\bar{X} \geq (1 + \Delta)\mu] < e^{-mb(\Delta)\mu}$$

We now bound the quantity $b(\Delta)\mu$ as follows:

$$b(\Delta)\mu = \left[\frac{k}{\mu} \ln \frac{k}{\mu} - \frac{k}{\mu} + 1 \right] \cdot \mu = k \ln \frac{k}{\mu} - k + \mu \geq k \ln 10 - k + \mu \geq k + \mu \geq 20\varepsilon_2$$

Hence, the previous probability bound implies $\Pr[\bar{X} \geq (1 + \Delta)\mu] < e^{-20\varepsilon_2 m}$

By application of the union bound to all possible discretized metrics, we obtain that sampling $m = \Omega(n \log^2 n)$ times guarantees that the probability that in *all* discretized metrics $\sum_{ij} \tilde{d}_{ij} l_{ij} > 20 \implies \sum_{ij} d_{ij} l_{ij} > 2$ is at least:

$$1 - 2^{n \log^2 n} e^{-\Omega(n \log^2 n)} \mapsto 1$$

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

We scale up the \tilde{d}_{ij} 's by $\sum_{ij} d_{ij}$. These demands will be used in the edge strategy.

B Finding sparse cuts using Eigenvalue computations

In the main body we used the following well-known theorem that arises from the work of Alon and Cheeger; for a proof see [9].

THEOREM 16 (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, there exists a randomized procedure, $\text{FINDCUT}(G)$, that in time $\tilde{O}(m + n)$ finds a cut with conductance at most $\sqrt{2c(G)}$.

One can iterate the above procedure to find γ -balanced separators; this is also standard but we prove it for completeness.

LEMMA 17

There is a procedure that, given a weighted graph G with degrees bounded by d , a fraction $\gamma > 0$, and an expansion bound β , uses Alon-Cheeger $O(n)$ times and produces either a cut of size γn with edge expansion less than βd , or a graph on at least $(1 - \gamma)n$ vertices on which every cut has expansion at least $\Omega(\beta^2 d)$.

PROOF: We always augment the graphs we get with self-loops to make it d -regular. Let $\lambda_d(G)$ denote the second eigenvalue of the Laplacian of the augmented G . The following procedure repeatedly uses FINDCUT to get cuts of expansion less than βd and unions them together, the resulting set also has expansion less than βd . When the procedure can no longer find cuts of expansion less than β then every cut in the remaining graph has expansion $\Omega(\beta^2 d)$ by the previous theorem.

Procedure FINDLARGECUT(G, d, γ, β)

Initialization: $S \leftarrow \phi$

while $\lambda_d(G \setminus S) < \beta^2/2$

$T \leftarrow \text{FINDCUT}(G \setminus S)$

$S \leftarrow S \cup T$

end while

if $|S| \geq \gamma n$ **then** return the set S

else return the graph $(G \setminus S)$

□

C Proof of Freund-Schapiro Theorem (Theorem 1)

PROOF:(Theorem 1) The proof is exactly on the lines of the proof of the main theorem in [Freund-Schapiro]. We show that that relative entropy between \mathbf{P} and \mathbf{P}^t , $\mathbf{RE}(\mathbf{P} \parallel \mathbf{P}^t) \doteq \sum_{i \in \mathcal{R}} \mathbf{P}(i) \ln(\mathbf{P}(i)/\mathbf{P}^t(i))$ goes down with every iteration. We have

$$\mathbf{RE}(\mathbf{P} \parallel \mathbf{P}^{t+1}) - \mathbf{RE}(\mathbf{P} \parallel \mathbf{P}^t) = \sum_{i \in \mathcal{R}} \mathbf{P}(i) \ln \frac{\mathbf{P}^t(i)}{\mathbf{P}^{t+1}(i)} = \sum_{i \in \mathcal{R}} \mathbf{P}(i) \ln \frac{Z^t}{(1-\epsilon)^{\mu(i, j^t)}} = \ln \frac{1}{1-\epsilon} \sum_{i \in \mathcal{R}} \mathbf{P}(i) \mu(i, j^t) + \ln Z^t$$

Next, we observe that $\mu(i, j^t) \in [0, 1]$. We use the bound $(1-\epsilon)^x \leq (1-\epsilon x)$ for $x \in [0, 1]$ to conclude

$$\ln Z^t \leq \ln \left[\sum_{i \in \mathcal{R}} \mathbf{P}^t(i) (1 - \epsilon \mu(i, j^t)) \right] = \ln [1 - \epsilon \sum_{i \in \mathcal{R}} \mathbf{P}^t(i) \mu(i, j^t)] \leq -\epsilon \sum_{i \in \mathcal{R}} \mathbf{P}^t(i) \mu(i, j^t)$$

The last inequality uses the fact that $\ln(1-x) \leq -x$ for any $x < 1$. We use the bound $-\ln(1-\epsilon) \leq \epsilon(1+\epsilon)$ for small ϵ to conclude

$$\mathbf{RE}(\mathbf{P} \parallel \mathbf{P}^{t+1}) - \mathbf{RE}(\mathbf{P} \parallel \mathbf{P}^t) \leq \epsilon(1+\epsilon) \sum_{i \in \mathcal{R}} \mathbf{P}^t(i) \mu(i, j^t) - \epsilon \sum_{i \in \mathcal{R}} \mathbf{P}^t(i) \mu(i, j^t)$$

Finally, we sum up from $t = 1$ to T . The sum telescopes; we use the fact that $\mathbf{RE}(\mathbf{P} \parallel \mathbf{P}^T) \geq 0$, and $\mathbf{RE}(\mathbf{P} \parallel \mathbf{P}^1) \leq \ln N$ because \mathbf{P}^1 is uniform. After rearranging the terms, and dividing by ϵ we conclude

$$\sum_{t=1}^T \sum_{i \in \mathcal{R}} \mathbf{P}^t(i) \mu(i, j^t) \leq (1+\epsilon) \sum_{t=1}^T \sum_{i \in \mathcal{R}} \mathbf{P}_{ij} \mu(i, j^t) + \frac{\ln N}{\epsilon}$$

Expanding out $\mu(i, j^t) = (\mathbf{M}(i, j^t) - \ell)/w$, dividing by T , and rearranging terms we get

$$\frac{1}{T} \sum_{t=1}^T \mathbf{M}(\mathbf{P}^t, j^t) \leq \frac{1}{T} \sum_{t=1}^T [\mathbf{M}(\mathbf{P}, j^t) + \epsilon(\mathbf{M}(\mathbf{P}, j^t) - \ell)] + \frac{w \ln N}{\epsilon T}$$

We bound $(\mathbf{M}(\mathbf{P}, j^t) - \ell) \leq w$, and then using the specified values of ϵ and T , we get (4). □