Expander Flows and Sparsest Cut Linear Programming

William Lin

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2024-119
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2024/EECS-2024-119.html

May 17, 2024

Acknowledgement

I would like to thank Professor Satish Rao for his guidance and mentorship. Under him I have been exposed to many different algorithmic paradigms and ways of thinking. He has been indispensable in developing my abilities as a researcher and writing my thesis.

I would like to thank my family for supporting me through my time at Berkeley and before. The skills and values I learned at home were just as important as the knowledge I learned in school.

Expander Flows and Sparsest Cut Linear Programming

by William Lin

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Professor Satish Rao
Research Advisor

(Date)

* * * * * * *

Professor Alistair Sinclair
Second Reader

(Date)

Expander Flows and Sparsest Cut Linear Programming

by

William Lin

A thesis submitted in partial satisfaction of the

requirements for the degree of

Masters

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Satish Rao, Chair
Professor Alistair Sinclair

Spring 2024

Expander Flows and Sparsest Cut Linear Programming

Abstract

We discuss a linear programming formulation of expander flows as applied to the sparsest cut problem. The sparsest cut problem looks to find the cut that minimizes the expansion: the ratio of edges to the number of vertices on the smaller side, and is NP-hard. A key idea to getting approximate certificates for this is expander flows, which embeds an expander of known expansion in the graph. This provides a lower bound on the number of edges in cuts in the original graph as significant flow must pass between the sets when embedding an expander. By proving that expander flows are approximately feasible in terms of the original graphs expansion, one can either certify the original graph expands or find a non-expanding cut. Originally, the embeddings required multicommodity flow methods but the cut matching paradigm allowed the use of a few calls to a single commodity flow algorithms to find sparse cuts.

In this thesis, we make the observation that good solutions to the expander flow linear program exist that only use a sequence of maximum single commodity flows. We note that the implementation remains difficult due to the lack of existence of a sufficiently good separation oracle for the expander flow linear program when using the ellipsoid algorithm, and briefly describe previous approaches that do this with worse bounds.

Another observation we make using this setup is that for graphs with small cuts, the approximation can be improved to only depend on the size of the cut rather than a function of the number of vertices.

Acknowledgments

I would like to thank Professor Satish Rao for his guidance and mentorship. Under him I have been exposed to many different algorithmic paradigms and ways of thinking. He has been indispensable in developing my abilities as a researcher and writing my thesis

I would like to thank my family for supporting me through my time at Berkeley and before. The skills and values I learned at home were just as important as the knowledge I learned in school.

Contents

Chapter 1

Introduction

For a given graph $G = (V, E)$, we define the edge expansion of a cut $S \subseteq V, |S| \leq \frac{|V|}{2}$ as

$$\phi(S) = \frac{|E(S, \bar{S})|}{|S|} \tag{1.1}$$

where $E(S, \bar{S})$ is the set of edges with one edge in S and the other in $\bar{S}$. We then define the expansion of the graph to the the minimum edge expansion over all cuts in the graph, and denote it by ϕ. The sparsest cut problem is to find the cut with edge expansion equal to the expansion. There are many variants of the problem centered around minimizing other quantities of expansion, or ensuring each side of the cut is balanced, but they are generally reducible to each other with little overhead.

The problem is $\mathsf{NP}-$hard, so we consider approximation algorithms. One approach is to use Cheeger's inequality, which approximates the edge expansion for d-regular graphs G using the easily calculable smallest nonzero eigenvalue of the normalized graph Laplacian of G, here denoted as λ:

$$\frac{\phi^2}{2} \leq \lambda \leq 2\phi \tag{1.2}$$

which gives an $1/\phi$ approximation which is good when ϕ is large. A different approach introduced in [LR99] was to embed a complete graph into G using multicommodity flow. The result by Leighton and Rao showed that for every graph, it is possible to route $O(\phi/n \log n)$ flow between every pair of vertices, which gives an $O(\log n)$ approximation to the edge expansion of G.

We begin by introducing some relevant methods and ideas for the sparsest cut problem. We proceed to describe a linear programming formulation of embedding an arbitrary expander into the graph which has been used to approximate edge expansion. We will see this as a frame to previous work which developed the cut matching methodology. Finally, we provide an $O(\log k)$ approximation for sparsest cut, where $k = \phi n$ is the number of edges in the sparsest cut by observing that the complexity of the expander flow formulation can be reduced in this case.

1.1 Expander Flows

Expander flows came about as a framework for approximate certification of the expansion of a graph. We have that verifying the expansion is at least α intuitively has no short certificate, as it would require certifying that every cut must have more than α edge expansion. This is known to be coNP complete [BKV+81]. However, given that a graph has α expansion, it was shown that it is possible to certify that all cuts must have expansion of at least $\Omega(\alpha/\log n)$ by embedding a multicommodity flow. The idea has been established over many papers [JS89][LR99][Sin92], and was refined to give a certificate that the expansion is at least $\Omega(\alpha/\sqrt{\log n})$. We give a short summary of this method as in [ARV04].

We define the edge expansion as before, however here we consider working with weighted graphs for simplicity. In this, instead of the numerator being the number of edges crossing between S and $\bar{S}$, it is the total weight of edges. Then we consider multicommodity flows.

Definition 1.1. *A **multicommodity flow** in a graph $G = (V, E)$ is an assignment of demands $d_{ij} \geq 0$ so that flows along the paths satisfy the demands. It is feasible if we can find a flow f_p such that*

$$\begin{array}{rl} \forall i, j \in V & \sum_{p \in P_{ij}} f_p = d_{ij} \\ \forall e \in E & \sum_{p \ni e} f_p \leq 1 \end{array}$$

It is common to view multicommodity flow as an extension of max flow, where demands between nodes must be of 'different commodities' while on the same edges. We can then view a feasible multicommodity flow as an embedding of a weighted graph G' with weights as the demands from i to j. We can see an equivalent definition as $w_e = \sum_{e \ni p} f_p$. If this is embedded onto G, we see that flow from i to j is represented in the edge weights of G'. If we then have a complete graph with flow f being able to be routed between all i, j pairs, we have in the original graph that the original graph's edges must be able to carry that amount of flow. Because of this, when we consider the edge expansion of $G' = \min_{S:|S| \leq |V|/2} \sum_{e \in E(S,\bar{S})} \frac{w_e}{|S|}$, we see that this gives us a lower bound on the expansion on the original graph as $\sum_{e \in E(S,\bar{S})} w_e = \sum_{e \in E(S,\bar{S})} \sum_{e \ni p} f_p \leq \sum_{e \in E(S,\bar{S})} 1 = |E(S, \bar{S})|$.

Thus, finding a set of feasible flows and demands in a graph provides a lower bound on the expansion, but this can only be done to an approximate factor. If done with the complete graph as in [LR99], the expansion when each flow is f would give $\min_{|S|:|S| \leq |V|/2} \frac{f|S||\bar{S}|}{|S|} = \min_{|S|:|S| \leq |V|/2} f|\bar{S}| = \frac{fn}{2}$. Note that embedding the complete graph may overuse edges in the multicommodity flow, since it demands every vertex be routed to each other. When the edges are overused, this puts a limit on the flow f that we can send from each vertex to the other, making it difficult to get an optimal approximation. An approach helping to resolve this instead chooses to embed expander graphs instead. If we are able to find such expanders that can be routed in the original graph, this would give us nice bounds as well.

Arguments like this show that it may not always be possible to embed a graph with high enough flow to be a valuable lower bound. Structural theorems on such graphs show that we can find productive lower bounds on the expansion.

Theorem 1.2 (Leighton Rao 1999)**.** *For any graph G with edge expansion ϕ, the complete graph associated with a multicommodity flow of value $f = \phi/n \log n$ can be embedded into the graph, which gives an $O(\log n)$ approximation.*

Theorem 1.3 (Arora Rao Vazirani 2004)**.** *For any graph G with edge expansion ϕ, an expander graph associated with a multicommodity flow of value $f = \phi/\sqrt{\log n}$ can be embedded into the graph, which gives an $O(\sqrt{\log n})$ approximation.*

1.2 Cut Matching

Cut matching was introduced in [KRV06] as a framework for finding approximations to the sparsest cut problem. The algorithm functions as a two player game between a cut player and a matching player. Given a graph $G = (V, E)$, the algorithm maintains a graph on the vertex set of G, H. The goal is to use flows to ensure that H becomes an expander that can be embedded in G, thus certifying the expansion of G and providing a cut.

Theorem 1.4 (Khandekar Rao Vaizrani 2007)**.** *Given an undirected graph $G = (V, E)$, there is a randomized algorithm that outputs a cut $(S, \bar{S})$ and a corresponding expander H on V that is embedded in G with congestion at most $O(\log^2 n/\phi(S))$.*

The algorithm proceeds in rounds where in each round, the cut player provides a bisection of the vertices, and the matching player matches vertices using a maximum flow calculation to give a perfect matching that is then added to H. Thus we have that H is the union of matchings across rounds, where the cut player wants to minimize the number of rounds before H is an expander. For the cut player to do so, the cut player looks to find cuts that are nonexpanding in H so when the matching is forced on it, the set now becomes expanding when H is added. This is done in the original paper by taking a random vector and mixing it along the edges of H, which should cause the separate sides of any nonexpanding set to have a differential in average value. So by taking the median and splitting it, this finds a weakly expanding set, and when the matching player routes a flow between the sides, this gives a matching.

The Cut matching framework has been used to design faster approximations for sparsest cut, as it only requires polylogarithmic many single commodity max flow computations, compared to the heavy multicommodity flow methods that came before. We seek to generalize cut matching and provide a linear programming approach for it.

1.3 SDP for Sparsest Cut

Semidefinite programming based methods have achieved success in approximating the sparsest cut. The linear programming approach covered in this paper takes ideas and results from semidefinite programming methods, particularly regarding embedding vertices into euclidian space and the structure behind it. The main idea in the SDP formulation is placing vertices on the sphere in euclidian space such that the average distance between edges is small, but vertices are far apart. This intuitively should try to group closely connected vertices in the graph together. A resulting embedding could work by letting the vectors be somewhat equidistant on the sphere which is too smooth, so to avoid this a triangle inequality constraint is added which forces the angle between any three points to be acute which makes the embedding to be more "cut-like" in a sense.

The SDP method used in [ARV04] (hereafter written as ARV) for the sparsest cut problem looks to place each vertex in R^n, and proves a result known as the ARV Structure Theorem on the embedding returned by the SDP. The structure theorem gives that under the SDP constraints, it is feasible to find two large, sufficiently far apart sets. From this these sets, by randomly projecting and taking a random radius ball around a set, this gives an $O(\sqrt{\log n})$ approximation. The given SDP is

$$\begin{aligned} \min \quad & \frac{1}{4} \sum_{\{i,j\} \in E} |v_i - v_j|^2 \\ \forall i \quad & |v_i|^2 = 1 \\ \forall i, j, k \quad & |v_i - v_j|^2 + |v_j - v_k|^2 \geq |v_i - v_k|^2 \\ & \sum_{i<j} |v_i - v_j|^2 = 1 \end{aligned} \tag{1.3}$$

For background, an SDP is similar to a linear program, except variables are constrained to be entries in a positive semidefinite matrix.

Definition 1.5. *A symmetric matrix $M \in R^{n \times n}$ is* ***positive semidefinite*** *(denoted $M \succeq 0$) if $\forall x \in R^n, x^T M x \geq 0$.*

This is equivalent to having all eigenvalues be greater than or equal to 0. A positive semidefinite matrix M can be decomposed by the Cholesky decomposition into $V^T V$ (we see that $\forall x, x^T V^T V x = \langle Vx, Vx \rangle \geq 0$), and thus with $M = V^T V$, we have that $M_{ij} = \langle v_i, v_j \rangle$, where v_i, v_j are columns i and j of V respectively. Thus, the positive semidefinite constraint in SDPs can be seen as enforcing that we are looking for vectors in R^n, and can put constraints on their inner products. With entries of M representing inner products, we can impose all of the constraints shown in the SDP, as we can see that $|v_i - v_j|^2 = \langle v_i, v_i \rangle - 2\langle v_i, v_j \rangle + \langle v_j, v_j \rangle$, which is linear in entries of M: $M_{ii} - 2M_{ij} + M_{jj}$. The points being on the sphere constraint is satisfied by $|v_i|^2 = 1$, and forcing the vertices to be far comes from the $\sum_{i<j} |v_i - v_j|^2 = 1$ constraint.

Using the SDP, we can find what is defined as an ℓ_2^2 representation: an embedding into euclidian space such that the points satisfy the triangle inequality, and are constant distance apart on average. The ARV structure theorem says that given an ℓ_2^2 representation, we can find two large sets that are far apart from each other. This idea is important as it allows one to conclude the structure of the embedding must induce a form of clusters, which would be relevant for finding nice cuts.

Theorem 1.6 (ARV Structure Theorem)**.** *For every $c > 0$, there are $c', b > 0$ such that every $c-$spread unit ℓ_2^2 representation with n points contains $\Delta-$separated subsets S, T of size $c'n$ where $\Delta = b/\sqrt{\log n}$. Furthermore, there is a randomized polynomial-time algorithm for finding these subsets S, T*

The proof of the structure theorem is involved and beyond the scope of this thesis.

1.4 Linear Program for Expander Flows

To proceed, we formulate the problem of finding a certifying expander flow as an exponentially sized linear program. We begin by defining the linear program, following section 7.3 from [ARV04]. A solution can give us a lower bound on the expansion. The framing is as a feasibility problem: for a graph G and parameter d can one route a flow that sends $\beta d|S|$ flow across every cut $(S, \bar{S})$ with $|S| \leq |V|/2$. For a constant β, this would give a βd lower bound on the expansion. We treat β as the edge expansion and d as the amount of flow from each vertex. Later we will consider modifications of this linear program and use it as a frame to discuss the cut matching game.

The linear program is as follows:

$$\forall i \in V \quad \sum_{j} \sum_{p \in \mathcal{P}_{ij}} f_p \leq d \tag{1.4}$$

$$\forall e \in E \quad \sum_{e \in p} f_p \leq 1 \tag{1.5}$$

$$\forall S \subseteq V, |V|/6 \leq |S| \leq |V|/2 \quad \sum_{i \in S, j \in \bar{S}} \sum_{p \in \mathcal{P}_{ij}} f_p \geq \beta d|S| \tag{1.6}$$

The dual program is as follows.

$$\min \quad \sum_{e} w_e + d \sum_{i} s_i - \beta d \sum_{S} z_S|S| \tag{1.7}$$

$$\forall i, j \in V, \forall p \in P_{ij} \quad \sum_{e \in p} w_e + s_i + s_j \geq \sum_{S: i \in S, j \in \bar{S}} z_S \tag{1.8}$$

$$z_S, w_e, s_i \geq 0 \tag{1.9}$$

The primal is feasible if and only if the dual has no negative valued solution. The ellipsoid algorithm in this case requires a separation oracle, which would give a violated constraint. This can be done using Cheeger's inequality to within a factor of β, since Cheeger's can provide a cut with expansion close to optimal. Note that this works best when β is constant, which is the formulation for such expander flows.

In the remainder of this thesis, we consider a linear programming framing for embedding an expander, that can be seen as a general frame of the cut matching game. In the cut matching game, the union of the matchings for each round will form a $\beta-$expander for $\beta = \Omega(1/\sqrt{\log n})$. The matching player routes the flow through the underlying graph producing the matching, guaranteeing that the matchings can be embedded in the underlying graph to certify the expansion.

By building on the analysis, we further show that for a graph G with edge expansion α there is a solution to the original linear program with $\beta = \Omega(1/\log n)$ and $d = O(\alpha)$ using the ARV structure theorem. Thus the solution certifies that the graph has edge expansion $\Omega(\alpha/\log n)$ as $\beta d|S| = \Omega(\alpha/\log n)|S|$ flow must be routed out of any set $|S|$ and therefore $E(S, \bar{S})$ is at least this large.

This establishes that the solution can be expressed as the convex combination of solutions to a single commodity maximum flow problem when $\beta = O(1/\sqrt{\log n})$. This was not previously written though it was folklore. We further provide observations on how the cut matching game strategy fits into the linear programming frame, and observations on how the expansion in linear programs is given by the structure theorem.

Finally, we consider finding $O(\log k)$ approximations where $k = \alpha n$ is the number of edges in the sparsest cut.

Chapter 2

Linear Programs for Sparsest Cut

2.1 Linear Program for Finding an Expander

To frame cut matching, we examine a simplified linear program that endeavors to construct an expander without embedding it into a graph.

We can then view the cut matching game as a strategy within this simplified framework to show that the cut-matching game is sufficient to build an expander for some values of β. We proceed with expander flows linear program which implements the strategies of the matching player using maximum flow.

In the original cut matching game, there are two players: the cut player and the matching player. The cut player presents a bisection of the vertices that represents a nonexpanding cut, while the matching player provides a matching on the vertices. The cut player can choose a bisection thus to maximize the expansion of the new graph.

We present the process as strategy to solve the following generic linear program to construct an expander graph on vertex set V.

$$\begin{aligned} \forall i \in V \quad & \sum_j x_{ij} \leq d \\ \forall S \subseteq V, |S| \leq |V|/2 & \sum_{i \in S, j \in \bar{S}} x_{ij} \geq \beta d |S| \end{aligned} \tag{2.1}$$

The x_{ij} represents a weighted graph on the vertex set. The constraint ensures that the weight of "edges" across any cut, $(S, V - S)$ (which we typically abbreviate as S), is at least $\beta d|S|$.

In viewing it from the point of view of primal dual 'algorithms', we see a feasible primal solution happens to be the complete graph (with self loops) with weights d/n. In this case, the value of β is $1/2$, as for any cut the weight of edges crossing is $\frac{d|S||\bar{S}|}{n} = \frac{d|S||\bar{S}|}{n} \geq \frac{d|S|}{2}$, as $|V - S| \geq n/2$. This satisfies the inequalities for $\beta = 1/2$.

Indeed, this solution yields the best possible value of β; a random partition cuts half the edges in a graph and everyone can't be better than average.

The cut-matching game is a game that generates some set of other solutions to this linear program in a more constrained fashion. This is useful for the case where the solution is "embedded" into another graph to certify its expansion. For example, a bounded degree expander cannot embed the complete graph, but it is already a solution to the linear program. A matching can be viewed as assigning $x_{ij} = d$ for the pairs (i, j) in the matching, and so works for the linear program.

The dual linear program is:

$$\begin{aligned} \min \quad & \sum_i s_i - \beta d \sum_S z_S |S| \\ \forall i, j \in V \quad & s_i + s_j - \sum_{i \in S, j \in \bar{S}} z_S \geq 0 \end{aligned} \tag{2.2}$$

A feasible dual with negative value implies the primal is infeasible. The cut-matching game maintains a non-optimal feasible dual, infeasible primal pair of solutions. For the cut matching game, the goal is for the primal to eventually become an expander, in which case it would become feasible.

With respect to the linear program, other strategies are possible. For example, for the primal one could modify one of the x_{ij} instead of requiring that they form a matching. For the dual linear programs, the cut player's strategy is basically general.

For this section, we analyse strategy that produces an expander on large sets.[1] That is, the primal is modified as follows.

$$\begin{aligned} \forall i \in V \quad & \sum_j x_{ij} \leq d \\ .\forall S \subseteq V, |V|/6 \leq |S| \leq |V|/2 \quad & \sum_{i \in S, j \in \bar{S}} x_{ij} \geq \beta d |S| \end{aligned} \tag{2.3}$$

Note that the d is just a scaling, but later it will be useful. For now, we assume that $d = 1$.

We will show that for some $\beta = \Theta(1/\sqrt{\log n})$, there exists a matching that gives a proof that any feasible dual has positive value. Duality then suggests that a set of matchings (appropriately weighted) will give a solution to the primal for this value of β.

Note that in general a proof would have an arbitrary structure in terms of the variables x_{ij}, but we look at the cut matching structure for now.

We restate the structure theorem from again [ARV04] in fundamental terms.

[1] A reasonably easy argument shows that deleting small non-expanding sets in such a graph will yield an expander with slightly lower value of β. This is argued in [ARV04]

Theorem 1.7 (ARV Structure Theorem)**.** *For a set of vectors, $v_1, \ldots, v_n$, with $\|v_i\| \leq 1$ and $d(i,j) = \|v_i - v_j\|^2$. If $d(i,j)$ satisfies the triangle inequality, i.e.,*

$$\forall i,j,k \in [n], d(i,k) + d(k,j) \leq d(i,j),$$

and

$$\frac{1}{n^2} \sum_{i,j} d(i,j) = \Omega(1),$$

there are sets A, B, with $|A|, |B| = \Omega(n)$, where

$$\min_{i \in A, j \in B} d(i,j) = \Omega(\frac{1}{\sqrt{\log n}}).$$

This is the same as the original statement of the structure theorem: we define the ℓ_2^2 representation as the condition that the points are embedded such that the squared distances obey the triangle inequality and are on average the points are far apart. We thus have there are two large sets that are somewhat far apart (at least $\Omega(1/\sqrt{\log n})$ apart.)

We proceed by using a dual solution to produce an embedding of the vertices so that we can apply the structure theorem. We want the embedding to give us a vector v_i for each $i \in V$ and that is an ℓ_2^2 representation.

Given a dual solution to 1.4, we embed the vertices into space such that distances are given by the cut metric. We use the variables z_S for $S \subseteq V$, and scale the dual solution to have $\sum_S z_S = 1$. The vector v_i has a coordinate for each S where $v_i = \sqrt{z_S}$ if $i \in S$ and $v_i = -\sqrt{z_S}$ otherwise. We see that by scaling it, the norm for each vector becomes $\sum_S (\pm\sqrt{z_S})^2 = \sum_S z_S = 1$, giving us that it is unit norm.

We make the observations that

1. For two vectors v_i and v_j, that $d(i,j) = \|v_i - v_j\|^2 = \sum_{i \in S, j \in \bar{S}} z_S$.
2. The distance $d(i,j)$ forms a metric: Each coordinate has two possible values and thus any three points share a value which implies that the triangle inequality holds for every coordinate. Since $\|v_i - v_j\|^2$ is a sum over the coordinates which obey the triangle inequality the overall distance obeys the triangle inequality
3. For all $i \in V$, $\|v_i\| = 1$.
4. And

$$\sum_{i,j} \|v_i - v_j\|^2 = \sum_S \sum_{i \in S, j \in \bar{S}} z_S = \sum_S |S||V - S| z_S = \Omega(n^2) \sum_S z_S,$$

where the last inequality is due our assumption that $z_S > 0$ only on sets of size $\geq |V|/6$.

This gives us that the points obey the conditions of theorem 1.7 and there are sets A, B of size $\Omega(n)$ where all pairs are $\Omega(1/\sqrt{\log n})$ apart under the distance metric $d(i,j) = \sum_{i \in S, j \in \bar{S}} z_s$.

For any partial matching, M, with $\Omega(n)$ pairs, (i,j) where $i \in A$, and $j \in B$, we can thus sum over corresponding constraints in the dual linear program 2.2.

$$\sum_{(i,j) \in M} s_i + s_j - \sum_{i \in S, j \in \bar{S}} z_S \geq 0.$$

Applying the fact that $d(i,j) = \Omega(1/\sqrt{\log n})$, $|P| = \Omega(n)$, and any vertex only appears in one term, we obtain the following inequality:

$$\sum_i s_i - \Omega(\frac{1}{\sqrt{\log n}}) \geq 0.$$

This is the objective function of the dual when $\sum_S z_S = 1$, and for some $\beta = \Theta(1/\sqrt{\log n})$. Thus, we have proven the following lemma:

Lemma 1.8. *There is a feasible solution to the expander linear program for some $\beta = O(\frac{1}{\sqrt{\log n}})$ which is a convex combination of arbitrary matchings between sets of size $\Omega(n)$.*

A remaining question is how many matchings does one need to form this solution.

Khandekar et. al. [KRV06] established that for $\beta = \Theta(1/\log^2 n)$, one could use $O(\log^2 n)$ matchings. Orecchia et. al. [OSVV08] improved β to be $\Theta(1/\log n)$. Arora and Kale [AK07] used a semidefinite programming approach to match this result.

One barrier in all these approaches is that finding a violating constraint (or a small cut) in the graph defined by a solution in the x_{ij} variables is difficult. One can use eigenvalue methods to approximately find a violating a constraint to within a factor β, which holds well for the linear program for the general graph.

Indeed, the approaches in [KRV06, OSVV08, AK07] all directly bound the eigenvalue rather than the minimum cut size. In a sense, they work with the following convex program:

$$\begin{aligned} \forall i \in V \quad & \sum_j x_{ij} \leq d \\ \lambda(L_x) \geq \beta^2 & \end{aligned} \tag{2.4}$$

In this program $\lambda(L_x)$ is the smallest non-zero eigenvalue of the Laplacian of the graph with edge weights x_{ij}.

Through the Cheeger inequalitie (1.2), this implies that every cut has expansion β^2, but one can only find a cut with expansion at most β. Thus, the method gives an approximate separation oracle with approximation factor β.

2.2 Cut/Matching to Embeddings

We return to the original linear program for embedding an expander flow and modify the previous analysis can be applied when adding in flow paths. The analysis follows the same frame as before. We restate the primal here:

$$\forall i \in V \quad \sum_j \sum_{p \in \mathcal{P}_{ij}} f_p \leq d \tag{2.5}$$

$$\forall e \in E \quad \sum_{e \in p} f_p \leq 1 \tag{2.6}$$

$$\forall S \subseteq V, |V|/6 \leq |S| \leq |V|/2 \quad \sum_{i \in S, j \in \bar{S}} \sum_{p \in \mathcal{P}_{ij}} f_p \geq \beta d |S| \tag{2.7}$$

Comparing to the generic cut matching program, we have flow path variables which, when added together for all paths between vertices i and j correspond to the x_{ij} variables. This is useful for considering the expander flow embedding, since we add the constraint that no edges is overused by flow paths and thus obtain a routable embedding of the graph defined by x_{ij} in our graph G.

The dual now adds variables w_e that correspond to the capacity constraints on edges. We combine the analysis on generic cut matching which builds an expander and apply it to these linear programs which embed that expander an underlying graph.

Here, we take $d = \alpha$ and $\beta = \Theta(1/\sqrt{\log n})$ where G is an α-expander, and thus its cuts should be able to support a $\beta\alpha|S|$ flow across any cut.

Here as in lemma 1.8, the z_S variables from the dual solution can be used to create the embedding as before and produce large sets A and B where every $d(i,j) = \Omega(1/\sqrt{\log n})$. In this case, we compute a single maximum flow from A to B where at most d units of flow are from each source in A and at most d units of flow are routed to each source in B. This is possible since G is an α-expander by assumption.[2] We thus obtain a setting of flow path variables f_p that allow us to sum the constraints in the dual.

We associate each path from i to j with variables f_p for $p \in \mathcal{P}_{ij}$, where $\mathcal{P}_{ij}$ is the set of paths from i to j. We can now sum over the flow paths for the correspond dual constraints.

$$\sum_p f_p(\sum_{e \in p} w_e + s_i + s_j - \sum_{i \in S, j \in \bar{S}} z_S)) \geq 0 \tag{2.8}$$

To analyze the result, we consider the effect of summing each part of the constraint using the paths. Since each $i \in A$ sends at most d flow using the setting of f_p variables, each $\sum_p f_p s_i = d s_i$ for $i \in A$. Similarly we have $\sum_p f_p s_i = d s_i$ for $i \in B$ due to each vertex in B receiving d flow. Also, recall that for any equation above we have $\sum_{i \in S, j \in \bar{S}} z_S \geq \frac{1}{\sqrt{\log n}}$.

[2] One need not assume G is an alpha expander as if the flow is not feasible, a sparser cut will be found. So one can view this as a way to find cuts of sparsity α or produce a solution to the LP.

And because $\sum_{f_p} f_p = \Omega(dn)$, this gives us $\sum_p f_p \sum_{i \in S, j \in \bar{S}} z_S)) = \Omega(\frac{dn}{\sqrt{\log n}})$. Finally, due to the f_ps forming a feasible flow, we can conclude that $\sum_p f_p w_e \leq w_e$, since the edges in the graph only have unit capacity.

Thus, from equation 2.8 we can conclude that

$$\sum_e w_e + d \sum_i s_i - \frac{cdn}{\sqrt{\log n}} \geq 0$$

for some constant c. Considering the dual from the original linear program 1.7, we see this has the form of the objective function, so with $\beta \geq c/\sqrt{\log n}$ ensures that the objective is positive for any feasible dual solution. Thus, the primal is feasible demonstrating the following lemma.

Lemma 1.9. *For an α-expander, the linear program is feasible for $d = \alpha$ and some $\beta = O(\frac{1}{\log n})$.*

We note that if the procedure fails to prove infeasibility, the maximum flow that was computed in the argument above would produce a cut with expansion at less than α. Thus, this lemma could be phrased as given an dual solution with negative value that the algorithm produces a cut of expansion at most α.

Finally, observing that the linear program with $\beta = \Omega(\frac{1}{\sqrt{\log n}})$ provides an $d\beta = \Omega(\frac{\alpha}{\sqrt{\log n}})$ lower bound on the expansion of G, allows us to view this an an approximation algorithm.

2.3 Few Matchings, Worse β in KRV.

We can now examine the algorithm from [KRV06] with the view that there is a good solution to the linear program using just matchings. The algorithm of [KRV06] produces an expander by maintaining and repeatedly adding matchings which are computed using a maximum flow computation, which can be seen as forming a convex combination of matchings in the linear program.

They maintain a vector which starts as random vector, v, where $\sum_i v_i = 0$, use the vector to compute a cut, and then a matching is computed. Then one average the endpoints to produce a new vector, and the vector's norm gets smaller. The vector norm getting smaller intuitively corresponds to a random walk mixing well.

To formally quantify this, they consider the potential function consisting of the sum over each vertex v of the norm of the distribution of the random walk starting at vertex v. In each step, the potential decreases in expectation by a factor of $1 - \frac{1}{\log n}$ and thus in $\Theta(\log^2 n)$ time essentially reaches the uniform distribution for all vertices.

Thus, the set of matchings must cross every cut, S, $\Omega(|S|)$ times to send enough distribution into S, and the resulting degree is $d = O(\log^2 n)$.

In a sense, this cut-matching game works by using an embedding that is not an ℓ_2^2 metric and does not obey the structure theorem established in ARV. This is part of the reason the approximation factor is worse.

As observed above, the expander flow linear program does have good solutions in the cut-matching framework. However, this structure has not yet been exploited to obtain an $O(\sqrt{\log n})$ approximation factor.

2.4 Constant β from the Proof of the Structure Theorem.

The theorem 1.7 states that for an appropriate point set there are two large sets that $O(1/\sqrt{\log n})$ apart. This translates as argued above into the statement that there is a feasible solution to the program with $\beta = \Theta(1/\sqrt{\log n})$.

In [ARV04], the main formulation allows for $\beta = O(1)$ while having $d = O(\alpha/\sqrt{\log n})$, giving a $\sqrt{\log n}$ approximation, however their proof involves computing a computationally heavy multicommodity flow. This is done to produce $\Omega(n)$ pairs of points that are each $\Omega(1)$ apart as pairs, but not as two large sets. Thus, the pairs of points must be routed as separate commodities.

Sherman [She09] implemented the proof itself using a sequence of $O(n^\epsilon)$ maximum flows to find $\Omega(n^{1-\epsilon})$ pairs of points that a far. His construction is quite complicated but attains an $O(\sqrt{\log n/\epsilon})$ approximation.

2.5 Improved bounds for very sparse cuts.

In this section, we give a method to give approximations that depend on the size of a sparse balanced cut rather than in terms of the number of vertices in the graph. This works well when the size of the cut is significantly less than the number of vertices, and has been recently had advances in [ALLS24]. We note that [FM06] also considered clustering for approximating the sparsest cut problem.

The idea is that for a graph with k edges in a sparse balanced cut, $B \subseteq V$, we can (easily) cluster the graph into $t = O(k)$ roughly equal sized clusters such that at most k clusters are split between S and $V - S$. We have that a sparse cut is $b-$balanced if $\min(|B|, |\bar{B}|) \geq bn$. More precisely, given a clustering (partition) of V, $\mathcal{C} = \{C_1, \dots, C_t\}$ and a cut B, we say a cluster C_i *is split by* B if it contains vertices from both B and $\bar{B}$. We say the $\mathcal{C}$ is c-balanced with respect to B if the total number of vertices in split clusters is at most cn.

We can modify the linear program from 1.4, to route flows between clusters instead of vertices. In the following, let $\mathcal{P}_{ij}$ be the set of paths from *any* vertex in cluster i to cluster j. We index clusters using an index set $I = [t]$.

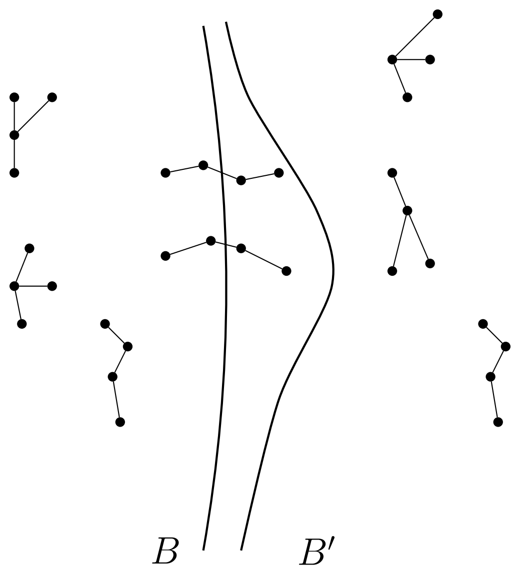

Figure 2.1: The cut B splits the clusters. The cut B' does not split any clusters. The balance of B' is different from B by the total size of the split clusters. The new linear program only optimizes over cuts like B' which does not split any clusters and for every small cut B there is a B' which is as nearly as balanced as B.

$$\begin{aligned} \forall i \in I \quad & \sum_j \sum_{p \in P_{ij}} f_p \leq d|C_i| \\ \forall e \in E \quad & \sum_{e \in p} f_p \leq 1 \\ \forall I \subseteq [t], S = \cup_{i \in I} C_i, |V|/6 \leq |S| \leq |V|/2, \quad & \sum_{i \in I, j \in \bar{I}} \sum_{p \in P_{ij}} f_p \geq \beta d|S| \end{aligned} \tag{2.9}$$

The first constraint says the flow out of a cluster to each other cluster must be less than $d|C_i|$, the second maintains the flow path constraint on the edges, and the final one ensures the expansion. Note that S is a set of vertices from the clusters, and we maintain the size constraint on S.

Note that if the clustering is c-balanced with respect to B which is b-balanced and has k edges between B and $V - B$, then there is some set $I \subseteq [t]$ with $S = \cup_{i \in I} C_i$ and $|S \cap B| \geq (b-c)n$. If we make c small then, S would be close to B. Thus, the linear program is only feasible for $\beta d(b-c)n \leq k$, as compared to the original linear program which is only feasible when $\beta dbn \leq k$.

We have that the linear program "collapses" the clusters in terms of the demands, but does not collapse the graph in that the flow paths are still paths in G and are still required to not overuse any edge in G as indicated in the second inequality above. Vertices are no longer routed to vertices in the same cluster, and can route to any vertex in the other clusters.

We also have that in the dual program that each variable z_S corresponds to a partition of the clusters rather than an arbitrary partition of the vertices of C. Thus, mapping a dual solution from variables z_S to an embedding of the vertices as before would assign the same vector to every vertex in a cluster. We can see the embedding only contains t points and is an ℓ_2^2 embedding as it is a cut metric. The ARV structure theorem then ensures that there are two large subsets $|A|, |B|$, of the points that are $\Omega(1/\sqrt{\log t})$ far in the embedding.

For completeness, the clustering method in the next section provides $t = O(k)$ clusters such that the total number of vertices in split clusters is at most $c \leq .04$

Lemma 1.10. *There is a linear time algorithm to find a set of clusters* $\{C_1, \ldots, C_{100k}\}$ *where each cluster has at most* $2n/100k$ *vertices and at most* $2k$ *clusters are split by at* B *for any cut* B *with fewer than* k *edges in the cut.*

Proof: The algorithm to form clusters is for a parameter $p = n/100k$. Recall that in a depth first search traversal of the graph consists of $2(n-1)$ edges where each of $n-1$ edges is traversed in either direction. The traversal is then divided into segments of length p. Note there are $2n/t$ segments. Each segment is connected and thus at most $2k$ segments contains an edge in the cut B. Any set of vertices in any other segment is not split by B. For each vertex arbitrarily assign it to a cluster corresponding to one of the segments that it is contained in. The clusters have size at most t, and there are at most $2n/p = 50k$ clusters at most $2k$ of which are split. Thus at most 4% of the vertices are in split clusters. □

2.6 Conclusion

We hope this thesis introduces readers to the fundamental ideas in expander flows and how linear programs are used for them. We hope this also provides another understanding for cut matching in terms of linear programming.

Bibliography

[AK07] Sanjeev Arora and Satyen Kale. A combinatorial, primal-dual approach to semidefinite programs. In David S. Johnson and Uriel Feige, editors, *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 227–236. ACM, 2007.

[ALLS24] Aditya Anand, Euiwoong Lee, Jason Li, and Thatchaphol Saranurak. Approximating small sparse cuts. *CoRR*, abs/2403.08983, 2024.

[ARV04] Sanjeev Arora, Satish Rao, and Umesh V. Vazirani. Expander flows, geometric embeddings and graph partitioning. In László Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 222–231. ACM, 2004.

[BKV+81] Manuel Blum, Richard M. Karp, Oliver Vornberger, Christos H. Papadimitriou, and Mihalis Yannakakis. The complexity of testing whether a graph is a superconcentrator. *Inf. Process. Lett.*, 13(4/5):164–167, 1981.

[FM06] Uriel Feige and Mohammad Mahdian. Finding small balanced separators. In Jon M. Kleinberg, editor, *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 375–384. ACM, 2006.

[JS89] Mark Jerrum and Alistair Sinclair. Approximating the permanent. *SIAM J. Comput.*, 18(6):1149–1178, 1989.

[KRV06] Rohit Khandekar, Satish Rao, and Umesh V. Vazirani. Graph partitioning using single commodity flows. In Jon M. Kleinberg, editor, *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 385–390. ACM, 2006.

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

[OSVV08] Lorenzo Orecchia, Leonard J. Schulman, Umesh V. Vazirani, and Nisheeth K. Vishnoi. On partitioning graphs via single commodity flows. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 461–470. ACM, 2008.

[She09] Jonah Sherman. Breaking the multicommodity flow barrier for o(vlog n)-approximations to sparsest cut. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*, pages 363–372. IEEE Computer Society, 2009.

[Sin92] Alistair Sinclair. Improved bounds for mixing rates of marcov chains and multicommodity flow. *Comb. Probab. Comput.*, 1:351–370, 1992.