CS880: Approximations Algorithms	
Scribe: Tom Watson	**Lecturer:** Shuchi Chawla
Topic: Balanced Cut, Sparsest Cut, and Metric Embeddings	**Date:** 3/21/2007

In the last lecture, we described an $O(\log k \log D)$-approximation algorithm for Sparsest Cut, where k is the number of terminal pairs and D is the total requirement. Today we will describe an application of Sparsest Cut to the Balanced Cut problem. We will then develop an $O(\log k)$-approximation algorithm for Sparsest Cut, due to Linial, London, and Rabinovich [4], using metric embeddings techniques of Bourgain [3].

18.1 Balanced Cut

Recall the Sparsest Cut problem.

Definition 18.1.1 (Sparsest Cut) *Given a graph $G = (V, E)$, edge capacities c_e, and requirements $r_{u,v} \geq 0$ for all $(u, v) \in V \times V$, find a set $S \subseteq V$ minimizing*

$$sparsity(S) = \frac{c(E(S, \overline{S}))}{\sum_{(u,v)\in(S\times\overline{S})\cup(\overline{S}\times S)} r_{u,v}}.$$

The sparsity of a cut is just its capacity divided by the total requirement separated by it. We denote by k the number of *terminal pairs* — pairs (u, v) such that $r_{u,v} > 0$.

We give an application of Sparsest Cut to the following problem.

Definition 18.1.2 (Balanced Cut) *Given a graph $G = (V, E)$, edge capacities c_e, and a balance requirement $\beta \leq 1/2$, find a minimum-capacity cut $(S, \overline{S})$ subject to the constraint that $(S, \overline{S})$ is β-balanced, i.e. $|S|, |\overline{S}| \geq \beta n$ where $n = |V|$.*

In the following, we use C_β to refer both to the minimum capacity of a β-balanced cut and to an optimal cut itself.

The key link between the two problems is that given an instance of Balanced Cut, we can give every pair of nodes a requirement of 1, and then for a given balance β, the capacity of a β-balanced cut is $\Theta(n^2)$ times its sparsity.

We use this link to obtain a *pseudo-approximation algorithm* for Balanced Cut. That is, for a given β and $\beta' < \beta$, we find a cut of balance β' of capacity within some guaranteed factor of C_β. (Note that C_β may be much higher than $C_{\beta'}$.) Specifically, we obtain the following result.

Theorem 18.1.3 *If there exists a ρ-approximation algorithm for Sparsest Cut, then for all $\beta \leq 1/2$ and all $\beta' < \beta$, there exists an algorithm for Balanced Cut that finds β'-balanced cut of capacity at most*

$$\frac{\rho}{(\beta - \beta')(1 - \beta + \beta')} C_\beta,$$

provided $\beta' \leq 1/3$.

Proof: We would like to use our Sparsest Cut algorithm to get information about balanced cuts in G, so we declare every pair of nodes to have requirement 1. Then the total requirement separated by a β-balanced cut is at least $\beta(1-\beta)n^2$, and so

$$sparsity(C_\beta) \leq \frac{C_\beta}{\beta(1-\beta)n^2}.$$

Thus if the hypothesized ρ-approximation algorithm for Sparsest Cut happens to return a β'-balanced cut, then its sparsity is at most

$$\frac{\rho}{\beta(1-\beta)n^2}C_\beta$$

and it separates less than n^2 requirement, so it has capacity at most

$$\frac{\rho}{\beta(1-\beta)}C_\beta$$

and we can output this cut.

If the algorithm doesn't return a β'-balanced cut, then we can repeat this process on the larger side of the cut, taking the smaller side of the cut obtained and combining it with the smaller side of our first cut. One key observation is that since the first cut isn't β'-balanced, the larger side must be a large fraction of the original graph G, and so its optimal sparsity can't be too much larger than that of G. This allows us to argue that the capacity of the final cut we obtain isn't too much larger than C_β. The other key observation is that by iterating this process on the larger side, we end up with a cut that is relatively balanced. These ideas motivate the following algorithm, which we analyze formally next.

1) Set $r_{u,v} = 1$ for all $(u, v) \in V \times V$.
2) Set $\overline{S_0} = V$, $i = 0$.
3) While $|S_1 \cup \cdots \cup S_i| < \beta' n$,
4) Increment i.
5) Apply the hypothesized Sparsest Cut algorithm to the subgraph $G_{\overline{S_{i-1}}}$ induced on $\overline{S_{i-1}}$ to obtain a cut $(S_i, \overline{S_i})$ where $|S_i| \leq |\overline{S_i}|$.
6) Output $S_1 \cup \cdots \cup S_\ell$, where ℓ is the final value of i.

We argue that the final cut has capacity at most

$$\frac{\rho}{(\beta-\beta')(1-\beta+\beta')}C_\beta.$$

The intuition is that in each iteration, if $|S_i|$ is small, then the cut $(S_i, \overline{S_i})$ does not separate much requirement, but since it has low sparsity, it must have small capacity and so it is safe to cut those edges. More formally, the amount of requirement separated by $(S_i, \overline{S_i})$ is clearly at most $|S_i| \cdot n$, so the capacity of $(S_i, \overline{S_i})$ is at most $\rho \cdot |S_i| \cdot n$ times the optimal sparsity of a cut in $G_{\overline{S_{i-1}}}$. What is that optimal sparsity? It's at most the sparsity of the cut C_β restricted to $G_{\overline{S_{i-1}}}$. Clearly, C_β

cannot have larger capacity in $G_{\overline{S_{i-1}}}$ than it does in G. Since each side of C_β contains at most $(1-\beta)n$ nodes of G, and $|\overline{S_{i-1}}| > (1-\beta')n$, it follows that each side contains at least

$$(1-\beta')n - (1-\beta)n = (\beta - \beta')n$$

nodes of $G_{\overline{S_{i-1}}}$ and thus C_β separates at least

$$(\beta - \beta')n(1 - \beta + \beta')n$$

requirement in $G_{\overline{S_{i-1}}}$. Thus the sparsity of C_β in $G_{\overline{S_{i-1}}}$ is at most

$$\frac{C_\beta}{(\beta - \beta')(1 - \beta + \beta')n^2}.$$

We conclude that the capacity of $(S_i, \overline{S_i})$ is at most

$$\rho \cdot |S_i| \cdot n \cdot \frac{C_\beta}{(\beta - \beta')(1 - \beta + \beta')n^2} = \frac{\rho}{(\beta - \beta')(1 - \beta + \beta')n} \cdot C_\beta \cdot |S_i|.$$

The capacity of the final cut $(S_1 \cup \cdots \cup S_\ell, \overline{S_\ell})$ is at most the sum of the capacities of the cuts found in all the iterations (it could be less since an edge crossing from S_i to $\overline{S_i}$ could have an endpoint in e.g. S_{i+1} and thus not cross the final cut). Furthermore, $|S_1 \cup \cdots \cup S_\ell| = |S_1| + \cdots + |S_\ell| < n$ since the S_i's are disjoint, and so the capacity of the final cut is at most

$$\sum_{i=1}^{\ell} \frac{\rho}{(\beta - \beta')(1 - \beta + \beta')n} \cdot C_\beta \cdot |S_i| < \frac{\rho}{(\beta - \beta')(1 - \beta + \beta')} \cdot C_\beta.$$

All that's left to argue is that this algorithm gives a β'-balanced cut. We know that $|S_1 \cup \cdots \cup S_\ell| \geq \beta' n$ since otherwise the algorithm wouldn't have terminated. We also know that $|S_1 \cup \cdots \cup S_{\ell-1}| < \beta' n$ and so $|\overline{S_{\ell-1}}| > (1-\beta')n$, which implies that $|\overline{S_\ell}| \geq \frac{1-\beta'}{2}n$. In order for this side of the final cut to meet the balance requirement, we just need $\frac{1-\beta'}{2} \geq \beta'$, i.e. $\beta' \leq 1/3$. ■

18.2 Sparsest Cut

18.2.1 Results

In the last lecture, we used an LP relaxation of Sparsest Cut to obtain the following result.

Theorem 18.2.1 *There is an $O(\log k \log D)$-approximation algorithm for Sparsest Cut, where $D = \sum_{u,v} r_{u,v}$.*

Today, we embark on proving the following stronger result. Our algorithm uses metric embeddings technology.

Theorem 18.2.2 *There is an $O(\log k)$-approximation algorithm for Sparsest Cut.*

We only exhibit an $O(\log n)$-approximation algorithm. With a little more work, it can be extended to obtain an $O(\log k)$-approximation.

The following result, due to Arora, Lee, and Naor [1], is the state-of-the-art for Sparsest Cut.

Theorem 18.2.3 *There is an $O(\sqrt{\log k \log\log k})$-approximation algorithm for Sparsest Cut.*

The oldest result on the Sparsest Cut problem is due Leighton and Rao [5]. They obtained an $O(\log n)$-approximation for the uniform version, where every requirement $r_{u,v} = 1$ (as we used in the reduction from Balanced Cut). The state-of-the-art for Uniform Sparsest Cut is the following result of Arora, Rao, and Vazirani [2].

Theorem 18.2.4 *There is an $O(\sqrt{\log n})$-approximation algorithm for Uniform Sparsest Cut.*

We will not explore these results in depth in this course.

18.2.2 Relaxation of Sparsest Cut

We prove Theorem 18.2.2 in two steps. Today, we argue that the Sparsest Cut problem reduces to the problem of finding a good embedding of a metric into Euclidean space under the ℓ_1 norm. In the next lecture, we will show how to find such an embedding.

We start out by recalling the Maximum Concurrent Multicommodity Flow problem.

Definition 18.2.5 (Maximum Concurrent Multicommodity Flow) *For a given graph $G = (V, E)$, edge capacities c_e, and requirements $r_{u,v} \geq 0$ for all $(u, v) \in V \times V$, simultaneously route $f \cdot r_{u,v}$ units of flow from u to v for all (u, v), for f as large as possible.*

A separate commodity is defined for each (u, v) such that $r_{u,v} > 0$, and all commodities must be routed simultaneously while satisfying the capacity constraints. The Maximum Concurrent Multicommodity Flow problem is similar to Maximum Sum Multicommodity Flow, except that the flow values for different commodities must satisfy proportionality requirements. The Maximum Concurrent Multicommodity Flow problem can be exactly captured by an LP, and the dual of this LP turns out to be the following, where $\mathcal{P}_{u,v}$ is the set of all paths from u to v.

$$\begin{array}{rll} \text{minimize} & \sum_e c_e d_e & \\ \text{subject to} & \sum_{e \in p} d_e \geq y_{u,v} & \forall (u,v) \in V \times V,\ \forall p \in \mathcal{P}_{u,v} \\ & \sum_{u,v} r_{u,v} y_{u,v} \geq 1 & \\ & d_e \geq 0 & \forall e \in E \\ & y_{u,v} \geq 0 & \forall (u,v) \in V \times V \end{array} \tag{18.2.1}$$

In what follows, we prove many equivalences of various programs. In these programs, we have a set of variables $d_{u,v}$ for $(u, v) \in V \times V$ constrained to form some sort of metric on V, and we use the notation d_e to refer to the distance between the endpoints of edge e. When we refer to equivalences (or relaxations), we do not necessarily refer to the feasible set of one program being equal to (or a subset of) the feasible set of another program, but rather the ability to take a feasible solution to one and generate a feasible solution to the other with at least as good of an objective value.

Lemma 18.2.6 *Program (18.2.1) is equivalent to the following program.*

$$\begin{array}{rl} \text{minimize} & \sum_e c_e d_e \\ \text{subject to} & \sum_{u,v} r_{u,v} d_{u,v} \geq 1 \\ & d \text{ is a metric} \end{array} \tag{18.2.2}$$

Proof: A feasible solution to Program (18.2.2) immediately yields a feasible solution to Program (18.2.1) with the same objective value by setting $y_{u,v} = d_{u,v}$. Given a feasible solution to Program (18.2.1), the $y_{u,v}$'s might not form a metric, but for $\{u, v\} \notin E$ we can change $y_{u,v}$ to be the shortest path distance between u and v under edge lengths d_e, without changing the feasibility or objective value. Setting $d_{u,v} = y_{u,v}$ then gives a feasible solution to Program (18.2.2) with no worse objective value. ■

Lemma 18.2.7 *Program (18.2.2) is a relaxation of the Sparsest Cut problem.*

Proof: Given a cut $S \subseteq V$, let $d'_{u,v} = 1$ if u and v are on opposite sides of the cut, and $d'_{u,v} = 0$ otherwise (i.e. d' is the cut metric associated with S). The total requirement separated by the cut is $\sum_{u,v} r_{u,v} d'_{u,v}$ and hence

$$sparsity(S) = \frac{\sum_e c_e d'_e}{\sum_{u,v} r_{u,v} d'_{u,v}}.$$

It follows that if we set $d_{u,v} = d'_{u,v} / \sum_{u,v} r_{u,v} d'_{u,v}$ for all u, v, then we get a feasible solution to Program (18.2.2) having objective value

$$\sum_e c_e d_e = sparsity(S).$$

■

Corollary 18.2.8 *The optimum objective value for Program (18.2.2) is a lower bound on the minimum sparsity of a cut in G.*

Corollary 18.2.9 *The sparsity of every cut in G upper bounds the value f of every concurrent multicommodity flow in G.*

Proof: This follows from Corollary 18.2.8, Lemma 18.2.6, and weak duality, but it is also easy to see directly. For every cut and every concurrent multicommodity flow, all the flow for the commodities separated by the cut must flow through the cut, and the total amount of such flow is bounded by the capacity of the cut. Since this flow would have to be proportioned according to the $r_{u,v}$'s, the value f is bounded by the sparsity of the cut. ■

Our proof of Theorem 18.2.2 is an LP-rounding algorithm — we solve relaxation (18.2.2) via LP (18.2.1) and round the solution to obtain a cut. We next describe some equivalent characterizations of Sparsest Cut that help us accomplish this.

18.2.3 Characterizations of Sparsest Cut

Recall that a cut metric is one obtained by choosing a cut $S \subseteq V$ and setting the distance between two nodes to be 1 if they are on opposite sides of the cut and 0 otherwise. In the next result we show that if in Program (18.2.2) we require d to be a scalar multiple of a cut metric (note that $d_{u,v}$

refers to the cut metric distance between u and v, *not* to the shortest path distance when the edge lengths are d_e), then we actually get an exact characterization of the Sparsest Cut problem.

Lemma 18.2.10 *Sparsest Cut is equivalent to the following program.*

$$\begin{array}{rl} \text{minimize} & \sum_e c_e d_e \\ \text{subject to} & \sum_{u,v} r_{u,v} d_{u,v} \geq 1 \\ & d \text{ is a nonnegative scalar multiple of a cut metric} \end{array} \tag{18.2.3}$$

Proof: We already argued in Lemma 18.2.7 that a cut gives a feasible solution with objective value at most the sparsity of the cut. Conversely, a feasible solution can be assumed to satisfy $\sum_{u,v} r_{u,v} d_{u,v} \geq 1$ with equality, which implies that edges crossing the corresponding cut have length $1/\sum_{u,v} r_{u,v} d_{u,v}$ and thus the cut has sparsity equal to the objective value of the feasible solution. ■

Lemma 18.2.11 *Program (18.2.3) is equivalent to the following program.*

$$\begin{array}{rl} \text{minimize} & \sum_e c_e d_e \\ \text{subject to} & \sum_{u,v} r_{u,v} d_{u,v} \geq 1 \\ & d \text{ is linear combination of cut metrics, with nonnegative coefficients} \end{array} \tag{18.2.4}$$

Proof: A feasible solution to Program (18.2.3) is trivially a feasible solution to this program. Now consider a feasible solution to Program (18.2.4), corresponding to some sets S and nonnegative real coefficients β_S, and let d_S denote the cut metric corresponding to S. The solution can be assumed to satisfy

$$\sum_{u,v} r_{u,v} d_{u,v} = \sum_S \beta_S \sum_{u,v} r_{u,v} (d_S)_{u,v} = 1,$$

and so the objective value is

$$\frac{\sum_S \beta_S \sum_e c_e (d_S)_e}{\sum_S \beta_S \sum_{u,v} r_{u,v} (d_S)_{u,v}}.$$

We leave it as an exercise to show that the smallest ratio

$$\frac{\sum_e c_e (d_S)_e}{\sum_{u,v} r_{u,v} (d_S)_{u,v}}$$

is at most that objective value. Thus, selecting the sparsest cut S^* among those used to define d and rescaling so that $\sum_{u,v} r_{u,v} (d_{S^*})_{u,v} = 1$ gives a feasible solution to Program (18.2.3) with at least as good of an objective value.

For our application, we are given the linear combination of cut metrics that comprises d. ■

Before giving our last characterization, we need to define some standard normed metrics.

Definition 18.2.12 *For a positive integer p, the ℓ_p distance between $(x_1, \ldots, x_m) \in \mathbb{R}^m$ and $(y_1, \ldots, y_m) \in \mathbb{R}^m$ is defined to be*

$$\Big(|x_1 - y_1|^p + \cdots + |x_m - y_m|^p \Big)^{1/p}.$$

The ℓ_∞ distance is defined to be

$$\max_i |x_i - y_i|.$$

A metric is said to be an ℓ_p metric if its points can be mapped to $\mathbb{R}^m$ for some m in such a way that the ℓ_p distance between each pair of points is the same as their distance in the original metric.

The ℓ_1 metric is also known as the Manhattan metric, and the ℓ_2 metric is also known as the Euclidean metric. Interestingly, the ℓ_2 metric is the only one of the above metrics where the distance between two points does not depend on the particular coordinate system used. While it is possible to solve linear programs subject to the constraint that the variables specify a metric, it is NP-hard to optimize over ℓ_p metrics for certain values of p, e.g. $p = 1$. Indeed, the following result implies that being able to optimize over ℓ_1 metrics would allow us to solve Sparsest Cut exactly.

Lemma 18.2.13 *Program (18.2.4) is equivalent to the following program.*

$$\begin{array}{ll} \text{minimize} & \sum_e c_e d_e \\ \text{subject to} & \sum_{u,v} r_{u,v} d_{u,v} \geq 1 \\ & d \text{ is an } \ell_1 \text{ metric} \end{array} \tag{18.2.5}$$

Proof: Left as a homework problem. ■

Our strategy for approximating Sparsest Cut is to solve LP (18.2.1), or equivalently Program (18.2.2), and then "round" the solution to an ℓ_1 metric. Then the characterizations given by Lemmas 18.2.10, 18.2.11, and 18.2.13 allow us to construct a sparse cut. The precise notion of rounding is what we discuss next.

18.2.4 Metric Embeddings

Definition 18.2.14 *Given metric spaces (V, μ) and (V', μ') where μ and μ' are metrics on V and V' respectively, an embedding from (V, μ) to (V', μ') is a mapping $f : V \to V'$. The embedding is said to have expansion α if for all $x, y \in V$,*

$$\mu'(f(x), f(y)) \leq \alpha \mu(x, y).$$

The embedding is said to have contraction β if for all $x, y \in V$,

$$\mu'(f(x), f(y)) \geq \frac{1}{\beta} \mu(x, y).$$

If the embedding has expansion α and contraction β, then it is said to have distortion $\alpha \cdot \beta$.

Observe that rescaling either of the metrics by a fixed factor changes the expansion and contraction by that same factor, but the distortion remains constant.

We can now formalize the connection between Sparsest Cut and metric embeddings.

Theorem 18.2.15 *If there exists an efficiently computable ρ-distortion embedding from arbitrary metrics into ℓ_1, then there exists a ρ-approximation algorithm for Sparsest Cut.*

Proof: Suppose we have a ρ-distortion ℓ_1 embedding algorithm. We can solve LP (18.2.1) and obtain a metric d on V as in Lemma 18.2.6. We know from Lemma 18.2.7 that $\sum_e c_e d_e$ is a lower

bound on the optimal sparsity of a cut in G. We apply the ρ-distortion embedding algorithm to obtain an ℓ_1 metric. By rescaling, we may assume this embedding has contraction 1 and distortion ρ. Now we have a feasible solution to Program (18.2.5) with objective value at most a factor of ρ higher than our lower bound on the optimal sparsity. Lemmas 18.2.13, 18.2.11, and 18.2.10 then allow us to construct a cut of sparsity within a factor ρ of the optimum. ■

A partial converse to Theorem 18.2.15 is known. We omit the proof, which is involved.

Theorem 18.2.16 *If the integrality gap of LP (18.2.1) is ρ, then for every metric there exists a ρ-distortion embedding into ℓ_1.*

We now just need to obtain a low-distortion ℓ_1 embedding algorithm. We will show the following result in the next lecture.

Theorem 18.2.17 *There exists an efficiently computable $O(\log n)$-distortion embedding for arbitrary metrics into ℓ_1.*

Corollary 18.2.18 *There exists an $O(\log n)$-approximation algorithm for Sparsest Cut.*

As we mentioned before, with a little more work one can improve Corollary 18.2.18 to obtain Theorem 18.2.2.

It can be shown using the expander graph example from a previous lecture that Theorem 18.2.17 is tight (up to constant factors), i.e. there exist metrics such that no ℓ_1 embedding achieves distortion $o(\log n)$.

Note that this Sparsest Cut algorithm uses an embedding where the distortion guarantee applies to *every* distance, which is more than enough to prove the approximation guarantee. It is conceivable that we can do better using an embedding that only has low distortion *on average*. It turns out that for the uniform version, where every requirement is exactly 1, this approach works. The seminal result of Leighton and Rao [5] uses a low-average-distortion embedding to achieve an $O(\log n)$ approximation for Uniform Sparsest Cut. While their result is superceded by the present result of Linial, London, and Rabinovich [4], the best known result for Uniform Sparsest Cut, due to Arora, Rao, and Vazirani [2], uses low-average-distortion embeddings and achieves an $O(\sqrt{\log n})$ approximation, which is better than the best known approximation for Sparsest Cut, which is $O(\sqrt{\log n \log\log n})$. The $O(\sqrt{\log n})$ result uses a different relaxation, optimizing over ℓ_2^2 metrics, which can be done efficiently using semi-definite programming. It then uses a low-average-distortion embedding of ℓ_2^2 metrics into ℓ_1 metrics. This embedding can be made to have low distortion for *every* pair of points at the cost of an $O(\sqrt{\log\log n})$ factor loss in the approximation factor.

In the next lecture, we will describe how to obtain an $O(\log n)$-distortion embedding of an arbitrary metric on n points into ℓ_1. We will also start discussing the use of semi-definite programming in approximation algorithm design.

References

[1] S. Arora, J. Lee, A. Naor. Euclidean Distortion and the Sparsest Cut. In *STOC*, 2005, pp. 553-562.

[2] S. Arora, S. Rao, and U. Vazirani. Expander Flows, Geometric Embeddings and Graph Partitioning. In *STOC*, 2004, pp. 222-231.

[3] J. Bourgain. On Lipschitz Embedding of Finite Metric Spaces in Hilbert Spaces. In *Israeli J. Math.*, 52, 1985, pp. 46-52.

[4] N. Linial, E. London, and Y. Rabinovich. The Geometry of Graphs and Some of Its Algorithmic Applications. In *Combinatorica*, 15, 1995, pp. 215-245.

[5] T. Leighton and S. Rao. Multicommodity Max-Flow Min-Cut Theorems and Their Use in Designing Approximation Algorithms. In *Journal of the ACM*, 46, 1990, pp. 259-271.