CS880: Approximations Algorithms

Scribe: Matt Elder **Lecturer:** Shuchi Chawla
Topic: Sparsest Cut and Balanced Cut **Date:** 3/20/07

17.1 Multicut

First, consider the multicut problem. Given a graph $G = (V, E)$, K pairs of terminal vertices $\{s_i, t_i\}$, and a cost function on the edges $c\colon E \to \mathbb{R}$, the multicut problem asks for a minimum-cost cut of G that separates s_i and t_i for all i. Last time, we gave a $O(\log k)$ approximation for this problem.

The (relaxed) linear program for this problem is as follows; call it "Primal 1".

$$\begin{aligned} \text{minimize} \quad & \sum_{e \in E} c_e d_e \\ \text{where} \quad & d(s_i, t_i) \geq 1 \quad \forall i \\ & d \text{ is a metric} \end{aligned}$$

We can rewrite as follows:

$$\begin{aligned} \text{minimize} \quad & \sum_{e \in E} c_e d_e \\ \text{where} \quad & \mathcal{P}_i = \{\text{All paths from } s_i \text{ to } t_i\} \\ & \sum_{e \in P} d_e \geq 1 \qquad \forall i \, \forall P \in \mathcal{P}_i \\ & d_e \geq 0 \qquad \forall e \end{aligned}$$

The dual of this LP, which we'll call "Dual 1", is as follows:

$$\begin{aligned} \text{maximize} \quad & \sum_i \sum_{P \in \mathcal{P}_i} f_{i,P} \\ \text{where} \quad & \mathcal{P}_i = \{\text{All paths from } s_i \text{ to } t_i\} \\ & \sum_i \sum_{\substack{P \in \mathcal{P}_i \\ P \ni e}} f_{i,P} \leq c_e \qquad \forall e \\ & f_{i,P} \geq 0 \end{aligned}$$

Dual 1 solves the max-sum multi-commodity flow problem: c_e represents the capacity of an edge, and $f_{i,P}$ is the amount of flow directed from s_i to t_i along the path P. The LP tries to maximize the total amount of commodity flow.

Lemma 17.1.1 *Multicut is always larger than the corresponding max-sum multi-commodity flow.*

Lemma 17.1.2 *Multicut is at most $O(\log K)$ times the corresponding max-sum multi-commodity flow.*

Theorem 17.1.3 *When $k = 2$, multicut equals max-sum multi-commodity flow.*

17.2 Maximum Concurrent Multicommodity Flow

A solution to Dual 1 may starve some commodities while routing others. In contrast, max-concurrent multicommodity flow routes equal fractions of all commodities while respecting capacities. Thus, we devise the following LP for max-concurrent multicommodity flow, which we call Dual 2:

$$\begin{aligned} \text{maximize} \quad & t \\ \text{where} \quad & \sum_i \sum_{\substack{P \in \mathcal{P}_i \\ P \ni e}} f_{i,P} \leq c_e && \forall e \\ & \sum_{P \in \mathcal{P}_i} f_{i,P} \geq r_i t && \forall i \\ & f_{i,P} \geq 0 && \forall i, \forall P \in \mathcal{P} \end{aligned}$$

Intuitively, the difference between Dual 1 and Dual 2 is that Dual 1 seeks to maximize the total flow across independent commodities, while Dual 2 seeks to maximize the minimum of a set of weighted flows. This is the *maximum concurrent multicommodity flow* problem.

Primal 2, the dual of the maximum concurrent multicommodity flow problem, is as follows:

$$\begin{aligned} \text{minimize} \quad & \sum_e d_e c_e \\ \text{where} \quad & \sum_{e \in P} d_e \geq y_i && \forall i, \forall P \in \mathcal{P}_i \\ & \sum_i r_i y_i \geq 1 \\ & d_e \geq 0 && \forall e \\ & y_i \geq 0 && \forall i \end{aligned}$$

The costs c_e are constants of the problem instance, so Primal 2 will seek to minimize the values for d_e. Thus, they will be no larger than they are constrained to be, so $y_i = d(s_i, t_i)$, the shortest distance from s_i to t_i where each edge e has length d_e. So, we can devise the following linear program, equivalent to Primal 2:

$$\begin{aligned} \text{minimize} \quad & \sum_e c_e d_e \\ \text{where} \quad & \sum_i r_i d(s_i, t_i) \geq 1 \\ & d \text{ is a metric} \end{aligned}$$

Up to scaling d, this is the same as the following program:

$$\begin{aligned} \text{minimize} \quad & \frac{\sum_e c_e d_e}{\sum_i r_i d(s_i, t_i)} \\ \text{where} \quad & d \text{ is a metric,} \end{aligned}$$

which will tie directly to the sparsest cut problem.

17.3 Sparsest Cut

Let $G = (E, V)$ be some graph. Let T, the set of terminals, be a set of pairs of vertices. For any cut S, a subset of V, let $\alpha(S)$ denote the *sparsity* of S, with

$$\alpha(S) = \frac{\left|E(S, \bar{S})\right|}{\left|(S \times \bar{S}) \cap T\right|}.$$

Thus, $\alpha(S)$ is the size of the cut S divided by the number of terminals that S separates. The *sparsest cut problem* takes G and T, and finds the cut S that minimizes $\alpha(S)$. We can generalize this further, by introducing a cost c on the edges and a weight r_i for each terminal i. Then, our more general $\alpha(S)$ looks like this:

$$\alpha(S) = \frac{c(E(S, \bar{S}))}{\sum_{\{i | s_i \in S \Leftrightarrow t_i \notin S\}} r_i}.$$

Consider the *uniform* version of sparsest cut, where we let the set of terminals be all possible pairs. That is, we let $T = V \times V$ and $r_{u,v} = 1$ for all $u \neq v$. Then, the sparsity is

$$\alpha(S) = \frac{c(E(S, \bar{S}))}{|S||\bar{S}|},$$

which is quite similar to the expansion of a set, from the context of expander graphs. Here, we give a $O(\log K \log D)$-approximation for this problem, where $D = \sum_i r_i$ and $K = |T|$. Next time, we'll find a $O(\log K)$-approximation. The best known efficient approximation is a $O(\sqrt{\log K \log\log K})$-approximation.

Suppose we solve Primal 2, yielding the metric d. Let $y_i \stackrel{\text{def}}{=} d(s_i, t_i)$. We know that $\sum_i r_i y_i \geq 1$ and that $\forall e, d_e \leq 1$. We need to round d into a cut metric — a metric with only ones and zeroes — without much increasing the sparsity.

Consider the special case where $\forall i, y_i \geq \frac{1}{2} y_{\max}$. Let $y_{\max} \stackrel{\text{def}}{=} \max_i y_i$, $d' \stackrel{\text{def}}{=} d/y_{\min}$, and $y'_i \stackrel{\text{def}}{=} d'(s_i, t_i)$ for all i. Then, $\forall i, y' \geq 1$, and d' is a feasible solution to the multicut LP. So, we can feed d' to the multicut approximation algorithm we saw last lecture. That algorithm will yield a cut of value $O(\log K) \sum_e c_e d'_e$. We deduce:

$$\begin{aligned} O(\log K) \sum_e c_e d'_e &\leq O(\log K) \frac{1}{y_{\min}} \sum_e c_e d_e \\ &\leq O(\log K) \frac{1}{y_{\max}} \sum_e c_e d_e. \end{aligned}$$

The demand this separates is thus $\sum_i r_i \geq (1/y_{\max}) \sum_i r_i y_i$. So, the sparsity of this algorithm, in this special case, is at most

$$O(\log K) \frac{\sum c_e d_e}{\sum r_i y_i}.$$

So, now consider the general case, with an arbitrarily large ratio between $y_{\max}$ and $y_{\min}$. Define I_x, the set of all y_i in a conveniently-defined interval, as:

$$I_x = \left\{ i | y_i \in \left(\frac{y_{\max}}{2^{x+1}}, \frac{y_{\max}}{2^x} \right] \right\}.$$

When x is constrained to the integers, it's clear that every y_i is contained in exactly one I_x. For each I_x, our algorithm will construct a multicut instance as in the special case, but it will scale d by $2^{x+1}/y_{\max}$ instead of $1/y_{\max}$. The sparsity for each of these instances is not too large:

$$\alpha(I_x) \leq O(\log K) \frac{\sum_e c_e d_e}{\sum_{i \in I_x} r_i y_i}.$$

If there exists a constant β and an x such that $\sum_{i \in I_x} r_i y_i \geq \beta^{-1} \sum_i r_i y_i$, then the sparsity of this instance is at least $O(\log K) \beta \sum c_e d_e / (\sum r_i y_i)$.

We claim that we can ignore all i such that $y_i < y_{\max}/D^2$. Again, D is the total demand $\sum r_i$. Define the set W containing wee values of y_i, $W = \left\{ i | y_i < y_{\max}/D^2 \right\}$. Ignoring W can result in the loss of at most $\sum_{i \in W} r_i y_i < \sum_{i \in W} r_i y_{\max}/D^2 \leq y_{\max}/D \leq 1/D$ from the denominator of our algorithm's sparsity. Assuming $D \geq 2$, ignoring W has only a small constant approximation cost. (If $D < 2$, this is an easy boundary case, which we can effeciently handle in an ad-hoc way.)

Thus, we need to consider only those I_x where $2^x < D^2$. There are at most $2 \log D$ such sets, so $\beta \geq 1/(2 \log D)$. This yields a $O(\log K \log D)$-approximation.

17.4 Balanced Cut

Given a graph $G = (V, E)$, the balanced cut problem demands the min-cost cut such that each side has at least αn nodes, for some constant value of $\alpha \leq \frac{1}{2}$. It is known that this problem is inapproximable to $n^{2-\epsilon}/\text{OPT}$ if P $\neq$ NP. This is an absurdly poor approximation.

So, we consider instead a *pseudo-approximation algorithm*, in which we approximate both the objective function of the optimal solution and the parameters of its instance. So, in this case, when asked for a cut with a balance of α, we instead output a cut with a balance α', such that $\alpha' < \alpha$ and $\alpha' \leq 1/3$. If the optimal cut of balance α has cost C_α, our cut will have cost no greater than $O(\log n) C_\alpha / (\alpha - \alpha')$. Notice that, though we have a reasonable bound on the ratio between the size of our cut and C_α, the ratio between the size of our cut and C'_α may be unbounded.

The algorithm employs a direct reduction to the sparsest cut problem, letting $T = V \times V$ and $r_i = 1$. Then, the sparsity of a cut S is, as before, $c(E(S, \bar{S}))/(|S||\bar{S}|)$. We discuss further details next time.

Even though this algorithm is far from optimal, it is actually useful. This pseudo-approximation has applications in divide-and-conquer algorithms on graphs. It ensures that we can always divide a graph into two pieces, each with size roughly linear in the size of the original graph, such that the cut between the pieces isn't too large. This yields log-depth recursion, which divide-and-conquer algorithms demand, while bounding the cost of recombining pieces.