Advanced Algorithms 2012A
Lecture 5 – flow/cut gap for sparse-cut*

Robert Krauthgamer

1 Concurrent flow and sparse-cut

1.1 Concurrent flow

Consider the same setup as in the multicommodity flow problem, i.e. undirected graph G with edge-capacities and k demand pairs $\{s_i, t_i\}$. In the concurrent flow problem, the goal is to ship λ units of flow between every demand pair, for the largest possible $\lambda > 0$.

The problem can be written as the LP below. We let P_i be the set of all $s_i - t_i$ paths. We have variables for flow paths and also λ.

$$\begin{array}{lll} \text{maximize} & \lambda & \\ \text{subject to} & \sum_{p \in P_i} f_p^i \geq \lambda & \forall i \in [k] \\ & \sum_{i \in [k]} \sum_{p \in P_i : e \in p} f_p^i \leq c_e & \forall e \in E \\ & f_p^i \geq 0 & \forall i \in [k], \forall p \in P_i \end{array} \tag{1}$$

Exer: Write an equivalent program that has a polynomial size.

1.2 Sparse-Cut

In the sparse-cut problem, the input is as above, and the goal is to find a set of edges $E' \subset E$ that minimizes the ratio between capacity(E') and the number of demands that are disconnected in $G \setminus E'$ (which might have many connected components).

Exer: show directly that in every network

maximum concurrent flow $\leq$ minimum sparse-cut,

*These notes summarize the material covered in class, usually skipping proofs, details, examples and so forth, and possibly adding some remarks, or pointers. The exercises are for self-practice and need not be handed in. In the interest of brevity, most references and credits were omitted.

and give an example where the inequality is strict (hint: use the complete bipartite graph $K_{2,3}$).

Exer: Prove that there is always an optimal solution that corresponds to some subset $A \subset V$, namely E' is a cut $(A, \bar{A})$.

By the exercise, it suffices to seek $A \subset V$ that minimizes:

$$\text{sparsity}(A) = \frac{\text{capacity(edges cut)}}{\#\text{(demands separated)}} = \frac{\sum_{uv \in E} c_{uv} 1_{\{|\{u,v\} \cap A|=1\}}}{\sum_{i \in [k]} 1_{\{|\{s_i,t_i\} \cap A|=1\}}} = \frac{\sum_{uv \in E} c_{uv} |1_A(u) - 1_A(v)|}{\sum_{i \in [k]} |1_A(s_i) - 1_A(t_i)|}.$$

1.3 LP relaxation for sparse-cut

The dual LP for (1) has variables $(y_e : e \in E)$ and exponentially many constraints:

$$\begin{array}{|lll|} \hline \text{minimize} & \sum_{e \in E} c_e y_e & \\ \text{subject to} & \sum_{e \in p} y_e \ge y_i & \forall i \in [k], \forall p \in P_i \\ & \sum_{i \in [k]} y_i = 1 & \\ & y_e \ge 0 & \forall e \in E \\ & y_i \ge 0 & \forall i \in [k] \\ \hline \end{array} \tag{2}$$

Observe that the second constraint can be abolished by changing the objective to be the ratio $\frac{\sum_{e \in E} c_e y_e}{\sum_{i \in [k]} y_i}$. Now, we can assume WLOG that y_i is just the shortest-path distance between s_i and t_i according to edge-lengths y_e.

Exer: Prove that this LP is a relaxation of the sparse-cut problem.

1.4 Flow/cut gap

Theorem 1 [Aumann-Rabani and Linial-London-Rabinovich after Leighton-Rao]:

$$\text{minimum sparse-cut} \le O(\log k) \cdot \text{maximum concurrent flow}.$$

Proof: Again, interpret the variables y_e as edge-lengths, and let $d(u,v)$ denote the distance (shortest-path) from u to v according to y_e. Observe that the LP value is at most $\frac{\sum_{uv \in E} c_{uv} d(u,v)}{\sum_{i \in [k]} d(s_i,t_i)}$.

Informally, the next step is to "convert" these arbitrary distance to a "tree metric" with only an $O(\log k)$ factor loss. We then convert the tree distances into a "cut metric" (with no further loss) which is just a cut $(A, \bar{A})$.

Lemma 2 [Probabilistic embedding into trees] [Gupta-Nagarajan-Ravi and Fakcharoenphol-Rao-Talwar after Bartal]: Let $d(.)$ be a metric on a set V of size n, and let $T \subset V$ be a collection of k terminals. Then there exists a randomized tree τ with vertex set $V_\tau \supseteq V$ (in fact the leaves are exactly V) and edge-lengths giving some distance d_τ, such that:

- For all $u, v \in V$ we have $\mathbb{E}[d_\tau(u,v)] \le O(\log k) \cdot d(u,v)$; and
- For all $t, t' \in T$ we have $d_\tau(t,t') \ge d(t,t')$ (with probability 1).

It is instructive to think of the case $T = V$ (thus $k = n$).

Proof of lemma: Below. The idea is to use algorithm CKR (from last week) recursively.

By applying Lemma 2 to a solution to LP (2) and terminals $T = \{s_1, t_1, \ldots, s_k, t_k\}$, we obtain a randomized tree τ such that:

$$\frac{\mathbb{E}_\tau[\sum_{uv\in E} c_{uv} d_\tau(u,v)]}{\sum_{i\in[k]} d_\tau(s_i,t_i)} \le O(\log k) \cdot \frac{\sum_{uv\in E} c_{uv} d(u,v)}{\sum_{i\in[k]} d(s_i,t_i)} \le O(\log k) \cdot \frac{\sum_{e\in E} c_e y_e}{\sum_{i\in[k]} y_i}$$

Fix henceforth a tree τ for which $\sum_{uv\in E} c_{uv} d_\tau(u,v)$ is no more than its expectation.

Lemma 3 [Extracting a cut from a tree metric]: Given a tree τ, there is $A \subset V_\tau$, i.e. a cut $(A, V_\tau \setminus A)$, such that

$$\frac{\sum_{uv\in E} c_{uv} |1_A(u) - 1_A(v)|}{\sum_{i\in[k]} |1_A(s_i) - 1_A(t_i)|} \le \frac{\sum_{uv\in E} c_{uv} d_\tau(u,v)}{\sum_{i\in[k]} d_\tau(s_i,t_i)}$$

To understand the lemma, it is instructive to think of the tree τ as a path, and then the cut A will be some "prefix" of the path.

Proof of lemma: Below. Basically an averaging argument over the tree's edges.

Using Lemma 3, we get a set $A \subset V_\tau$, and WLOG we may assume $A \subset V$ (because vertices of $V_\tau \setminus V$ do not really appear in the lemma), such that:

$$\frac{\sum_{uv\in E} c_{uv} |1_A(u) - 1_A(v)|}{\sum_{i\in[k]} |1_A(s_i) - 1_A(t_i)|} \le \frac{\sum_{uv\in E} c_{uv} d_\tau(u,v)}{\sum_{i\in[k]} d_\tau(s_i,t_i)} \le O(\log k) \cdot \frac{\sum_{e\in E} c_e y_e}{\sum_{i\in[k]} y_i}$$

i.e., a sparse-cut whose value is within factor $O(\log k)$ of the LP.

Theorem 2.1 follows using strong duality. QED.

Remark: It's not hard to verify that this gives a polynomial-time $O(\log k)$ approximation algorithm for the sparse-cut problem, which is NP-hard.

1.5 Proof of Lemma 3 (sketch)

Let E_τ be the set of edges in the tree τ, and let $\ell(.)$ be the edge lengths. Just like in every tree, removing a tree-edge separates the tree into two connected components. Thus, every tree-edge $xy \in E_\tau$ defines a partition $V_\tau = A_{xy} \cup A_{yx}$. Observe that we can write

$$d_\tau(u,v) = \sum_{xy\in E_\tau} \ell(xy) |1_{A_{xy}}(u) - 1_{A_{xy}}(v)|.$$

As seen in class, the lemma follows by using this formula together with the simple inequality: $\min_i\{\frac{c_i}{d_i}\} \le \frac{c_1+\cdots+c_n}{d_1+\cdots+d_n}$.

1.6 Proof of Lemma 2 (sketch)

The tree τ will correspond to a hierarchical decomposition (recursive partitioning) of V, as described below. Assume WLOG the minimum interpoint distance is 4, and set $\delta = \log \operatorname{diam}(V) + 2$.

Partition V using algorithm CKR (from last week) with $R = 2^\delta$, then compute a new partition of V using algorithm CKR with $R = 2^{\delta-1}$, and so forth using $R = 2^i$ for $i = \delta, \delta - 1, \ldots, 1, 0$. At each stage, "force" the partition of level i partition to be a refinement of all the previous partitions (by breaking level i clusters according to all higher level partitions). The result of this forced nesting is that now every level i cluster is completely contained in some level $i+1$ cluster.

The tree τ is the natural representation of this hierarchical decomposition, with the root of the tree representing the vertex-set V, its children represent the clusters at level δ, and so forth, until the leaves of the tree which represent the clusters for $R = 1$. Edges between a tree node at level i and its parent are given length 2^{i+2}. Ordinarily, the clusters at the leaves of the tree represent a cluster of size 1 (single vertex of V), but not always because CKR algorithm has a "leftover" cluster V_0. In this last case we add under this leaf $|V_0|$ children, each representing a single vertex of V_0, connected with zero edge lengths. It follows that the leaves of V_τ can be thought of as V.

The rest of the analysis (bounds on d_τ) was seen in class, and uses the important remark about how algorithm CKR depends on the term $O(\log \frac{|B_T(u,3\cdot 2^i)|}{|B_T(u,2^i/2)|})$.