Lecture 10: Feb 11

Lecturer: Mohammad R. Salavatipour *Scribe: Junfeng Wu*

In the next few lectures we will be talking about several cut problems.

10.1 Multiway cut

Consider the minimum s,t-cut problem:

Definition 10.1 *Given an undirected graph $G(V,E)$ with weights $w(e) \geq 0$ for every edge $e \in E$ and given two terminals $s,t \in V$, find a minimum cut between s,t (i.e. a set of edges whose removal disconnect s,t).*

This problem can be solved by using max-flow algorithms and the following theorem.

Theorem 10.2 (max-flow/min-cut) *The value of max-flow between s and t in G is equal to size of the minimum s,t-cut.*

Now suppose that instead of two terminals, we are given a set $S = \{S_1, ..., S_k\}$ of terminals and want to find a min-cut that separates all the terminials.

Definition 10.3 Multiway cut: *Given an undirected weighted graph $G(V,E)$ and $S = \{s_1, ..., s_k\}$ of terminals, find a minimum weight set C of edges such that all the terminals are disconnected in $G - C$.*

This problem is NP-hard for $k \geq 3$. Here, we are going to present a $(2 - \frac{2}{k})$-approximation algorithm for this problem. The idea is to take a union of cuts each of which separates at least one s_i from the other ones. We can find each of these cuts using max-flow.

Definition 10.4 *A set of edges is called an s_i-cut if removing that set separates s_i from $S - s_i$.*

Note that if we consider all the terminals in $S - s_i$ as one single terminal t then using the max-flow algorithms we can find an s_i-cut in polynomial time.

Multiway Cut:
  for $i \longleftarrow 1$ **to** k **do**
    let C_i be a minimum s_i-cut
  Discard the largest C_i and return the union of the other ones, call it C.

Theorem 10.5 *This is a $2(1 - \frac{1}{k})$-approximation algorithm for multiway cut.*

Proof: First observe that C is a multiway cut: WLOG assume that the discarded cut is C_k, i.e. $C = \bigcup_{i=1}^{k-1} C_i$. Thus C disconnects s_i from all other terminals for each $1 \leq i < k$. Thus s_k must be disconnected from other as well.

Nowe consider an optimal solution A and consider $G - A$. Since A is a multiway cut $G - A$ has $\geq k$ components. Because of the optimality of A, $G - A$ has $\leq k$ components (otherwise we have removed edges that have created components that don't contain any terminal, we can put those edges back). Thus $G - A$ has exactly k components. Let $G_1, ..., G_k$ be the components of $G - A$ and let A_i be the edges between G_i and $G - G_i$, i.e. the cut $(G_i, G - G_i)$. Note that $\bigcup_{i=1}^{k} A_i = A$ and since every edge of A belongs to exactly two of A_i's: $\sum_{i=1}^{k} w(A_i) = 2w(A)$, where $w(.)$ returns the total weights of edges of a set of edges. Since C_i is a minimum cut separateing s_i from the rest of the graph: $w(C_i) \leq w(A_i)$. Thus:

$$\begin{aligned} w(C) &\leq (1 - \frac{1}{k}) \sum_{i=1}^{k} w(C_i) \\ &\leq (1 - \frac{1}{k}) \sum_{i=1}^{k} w(A_i) \\ &\leq 2(1 - \frac{1}{k}) w(A). \end{aligned}$$

■

Here is a tight example for the analysis of this algorithm: Consider a graph on $2k$ vertices $v_1, \ldots, v_k, s_1, \ldots, s_k$ where $v_1, \ldots, v_k$ form a cycle of size k with each edge having weight 1 and each (terminal) s_i is connected (only) to v_i with an edge of weight $2 - \epsilon$ for an arbitrary small $\epsilon > 0$. Then for each terminal s_i: $w(C_i) = 2 - \epsilon$ and therefore $w(C) = (k-1)(2-\epsilon)$. But the optimal solution is to remove all the k edges of the cycle. So $w(OPT) = k$ and $\frac{w(C)}{w(OPT)} = (2 - \varepsilon)(1 - \frac{1}{k})$.

A $\frac{3}{2}$-approximation algorithm: chapter 19, using a clever LP formulation. The best known approximation algorithm for this problem is a 1.3438-approximation algorithm by Karger, Klein, Stein, Thorup, and Young (STOC '99).

10.1.1 Min Steiner k-cut and min k-cut

Definition 10.6 *Given a connected weighted undirected graph $G(V, E)$, find a minimum weight set C of edges that $G - C$ has k components.*

Unlike multiway cut, this problem belongs to P for any fixed k, but it remains NP-Complete for arbitrary k. There is a common generalization of both multiway cut and min k-cut, called Steiner k-cut.

Definition 10.7 Steiner k-cut: *given a connected undirected weighted graph $G(V, E)$, a set $X \subseteq V$ of terminals, and integer k; find a minimum weight cut that creates k components, $V_1, ..., V_k$ such that $V_i \cap X \neq \phi$ for $1 \leq i \leq k$.*

If $|X| = k$ then we have the multiway cut problem. If $X = V$ then we have the min k-cut problem. We present a $2(1 - \frac{1}{k})$-approximation algorithm for Steiner k-cut.

Let $T(V, E_T)$ be a tree on V (but may contain edges that are not in E). For each $uv \in E_T$, $T - uv$ has two components on vertex sets S and $V - S$. Consider the cut $(S, V - S)$ in G. We call this cut the cut associated with uv. If T satisfies the following two properties then we call it a *Gomory-Hu* tree:

1. $w(uv)$ in T is the weight of the cut associated with uv,
2. $\forall u, v \in G$, the minimum u,v-cut in G has the same weight as the minimum u,v-cut in T.

It can be proved that:

Lemma 10.8 *we can compute a Gomory-Hu tree in polytime.*

Steiner k-cut Algorithm:
Compute a G-H tree
For $k-1$ iteration do:
pick the smallest edge in T that separates a pair of terminals (in X) that are not already separated.
Return the union of the cuts associated with these edges, call it C.

Lemma 10.9 *The algorithm returns a Steiner k-cut.*

Proof: Clearly each component generated has at least one terminal. Also, each cut (corresponding to an edge of T) increases the number of components by 1. Since the algorithm has $k-1$ iterations, there will be k connected components at the end. ■

Theorem 10.10 *This is a $2(1-\frac{1}{k})$-approximation algorithm for Steiner k-cut.*

Proof: Assume that A is an optimal solution and let $V_1, ..., V_k$ be the components of $G - A$. Define $A_i = (V_i, V - V_i)$. Each V_i has at least one vertex of X. Choose a terminal from each each V_i and call it t_i, $1 \le i \le k$. Without loss of generality, assume that $w(A_1) \le w(A_2) \le \ldots \le w(A_k)$. We show that there are $k-1$ cuts defined by the edges of T whose weights are dominated by the weights of $A_1, \ldots, A_{k-1}$. Since each edge of A belongs to exactly two A_i's:

$$\sum_{i=1}^{k} w(A_i) = 2w(A).$$

Let $T' \subseteq T$ be the set of edges of T that correspond to the cuts $A_1, \ldots, A_k$. Consider the graph on vertex set V and edges set T'. Now shrink each V_i $(1 \le i \le k)$ into a single vertex t_i. We obtain a connected graph (because T was origianlly connected). Delete extra edges until we are left with a tree on $t_1, \ldots, t_k$, call it B. Note that the $k-1$ edges that remain in B belong to T, too. Put directions on the edges of B such that each edge is directed toward t_k. This helps in defining a correspondence between the edges of B and sets $V_1, \ldots, V_{k-1}$: each edge of B corresponds to one set V_i $(1 \le i \le k-1)$, i.e. the one which has the terminal that the edge is coming out of in the rooted tree. Consider an edge uv of B corresponding to a leaf, say t_i. By property 2 of G-H trees: $w(uv)$ is the weight of a minimum u,v-cut in G. Therefore, the weight of this edge is at most $w(A_i)$. We can now remove this vertex and edge from the tree and do the same argument. Since we pick the $k-1$ lightest edges of T (that separate terminals from X), this implies that:

$$\sum_{i=1}^{k-1} w(C_i) \le \sum_{i=1}^{k-1} (A_i).$$

Therefore:

$$w(C) \le \sum_{i=1}^{k-1} w(C_i) \le \sum_{i=1}^{k-1} w(A_i) \le (1-\frac{1}{k}) \sum_{i=1}^{k} w(A_i) \le 2(1-\frac{1}{k}) w(A).$$

■