Lecture notes on multiway cuts and k-cuts

Consider a connected, undirected graph $G = (V, E)$ with a weight function $w : E \to \mathbb{R}^+$. A *cut* is defined by a partition of V into disjoint sets V_1' and V_2', and consists of all edges $E' \in E$ which have one vertex in V_1' and one vertex in V_2'. The *weight* of this cut is defined as $w(E') = \sum_{e \in E'} w(e)$.

The most simple problem involving cuts is that of finding the minimum-cost cut which separates two nodes s and t (we call these nodes *terminals*). This problem is the dual of the maximum flow problem, which is solvable in polynomial time. However, there are two NP-hard generalizations of minimum cut which yield interesting approximation algorithms.

Multiway cut: Given a set of terminals $S = \{s_1, s_2, \ldots, s_k\} \subseteq V$, find the minimum-weight set of edges $E' \subseteq E$ whose removal separates each pair of terminals.

Minimum k-cut: Given an integer k, find the minimum-weight set of edges $E' \subseteq E$ whose removal divides the graph into k connected components.

Although the multiway cut problem is equivalent to minimum cut and thus polynomial time solvable when $k = 2$, it becomes NP-hard for any fixed $k > 2$. The minimum k-cut problem, on the other hand, is polynomial time solvable for any *fixed* k, but when k is given in the input it becomes NP-hard. We will give $(2 - \frac{2}{k})$ approximations for each.

1 Multiway cut

We can achieve our approximation for multiway cuts using minimum-weight *isolating cuts*. An isolating cut for a terminal s_i is a set of edges which disconnects s_i from each of the other terminals in S.

Finding the minimum-weight isolating cut for a terminal s_i is easily reducible to finding a minimum cut. We create a new graph in which a supernode s_i' replaces the remaining terminals $S - \{s_i\}$, where the edges leaving s_i' correspond to the edges leaving any of the terminals in $S - \{s_i\}$ in the original graph. Now we find a minimum cut between s_i and s_i' in this new graph, which represents the minimum isolating cut for s_i in G.

In our algorithm for multiway cuts, we will first compute the minimum-weight isolating cut C_i for each terminal s_i. Aggregating any $k - 1$ of these k isolating cuts will serve as a multiway cut in our graph, because such a cut would isolate each of the first $k - 1$ terminals and thus it would isolate the last terminal as well. We will show that the total weight of the $k - 1$ lightest isolating cuts is at most $(2 - \frac{2}{k})$ times that of the optimal multiway cut, giving

our desired bound.

Theorem 1 *The total weight of the $k-1$ lightest isolating cuts for the set of terminals S is at most $(2-\frac{2}{k})$ times the weight of the optimal multiway cut.*

Proof: Let A be the optimal multiway cut for S. If we remove the set of edges A from G, we must be left with k connected components $V_1, V_2, \ldots, V_k$, where V_i contains s_i as its only terminal. Let A_i be the set of edges in G between V_i and $V - V_i$, so that A_i represents an isolating cut for s_i.

Note that every edge $e \in A$ must pass between two components, call them V_i and V_j, so that $e \in A_i$ and $e \in A_j$. Thus each term in the sum of weights in $w(A)$ appears twice in the sum of weights in $w(A_1) + w(A_2) + \ldots + w(A_k)$. It follows that

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

Assuming without loss of generality that the cut A_k has the maximum weight of all the cuts A_i, we can therefore state that

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

Now recall that each A_i is an isolating cut for one of the terminals s_i. However, since we found the minimum-weight isolating cuts for each terminal, we necessarily have $w(C_i) \leq w(A_i)$ for each i. Using equation 1 gives

$$\sum_{i=1}^{k-1} w(C_i) \leq \sum_{i=1}^{k-1} w(A_i) \leq (2-\frac{2}{k})w(A). \tag{3}$$

Noting that $\sum_{i=1}^{k-1} w(C_i)$ is at least as large as the sum of the $k-1$ lightest isolating cuts, this concludes our proof. $\triangle$

2 Gomory-Hu Trees

A Gomory-Hu tree for $G = (V, E)$ is a tree T on the same set of vertices V. The edges of T are not necessarily in the set E and have a new weight function w' associated with them. In addition, each edge $e \in T$ which partitions T into components S and $T - S$ is said to represent the cut associated with separating S and $T - S$ in G.

The conditions for a Gomory-Hu tree are as follows:

1. For every pair of vertices u and v, the weight of the minimum cut between u and v is the same in both G and T.

2. For each edge $e \in T$, $w'(e)$ is the weight of the cut represented by e in G.

We can compute the Gomory-Hu tree on a graph G by maintaining a tree T on a collection of vertex sets $S_1, S_2, \ldots, S_t$, beginning with the single set $S_1 = V$. At each step, we take one of these sets S_i for which $|S_i| > 1$, and choose $u, v \in S_i$.

After rooting T at S_i, we create a new graph by starting with G and then collapsing each subtree of S_i into a single supernode. We then run a minimum cut between u and v in this new graph, dividing V into V_1 containing u and V_2 containing v. Our graph T is now modified by breaking S_i into $S_{i1} = S_i \cap V_1$ and $S_{i2} = S_i \cap V_2$, with an edge between them representing the minimum cut we just calculated. As for the subtrees of S_i, we connect a subtree to S_{i1} if its supernode was in the same partition as u in the minimum cut, and connect it to S_{i2} otherwise.

Performing this step $k-1$ times will result in a tree in which each node in T represents a single vertex from G. This final tree will satisfy the properties of the Gomory-Hu tree.

3 Minimum k-cut

We will achieve our approximation for minimum k-cut using T, the Gomory-Hu tree for G, with the associated weight function w'. Our approach will be to take the $k-1$ lightest cuts from the $n-1$ cuts associated with the edges in T. We can prove that removing these cuts from G will leave us with at least k connected components.

Theorem 2 *Removing ℓ cuts associated with ℓ edges in the Gomory-Hu tree for G results in a new graph G' with at least $\ell + 1$ connected components.*

Proof: Let $V_1, V_2, \ldots, V_{\ell+1}$ be the connected components which are left in T after the removal of the ℓ edges. For any $u \in V_i$ and $v \in V_j$ with $i \neq j$, we must have removed some edge in T which disconnects u and v. Recall that this edge has a cut associated with it in G which must disconnect u and v in G as well. Thus removing the edges in T must disconnect each set V_i from each of the other sets V_j, and it follows that there are at least $\ell+1$ connected components in G. $\triangle$

Now as with our proof for the multiway cut, let A be the optimal k-cut for G, let $V_1, V_2, \ldots, V_k$ be the components which A divides V into, and for each V_i let A_i be the set of edges between V_i and $V - V_i$. Thus each edge in A is contained in two of the A_i sets, and again we have

$$\sum_{i=1}^{k} w(A_i) = 2w(A). \tag{4}$$

Consider modifying T by shrinking the vertices corresponding to each V_i into a supernode, then removing edges until the graph becomes a new tree T'. Assume without loss of generality that A_k has the highest weight of all the cuts, and root T' at the the supernode V_k.

Now we can put every supernode V_i (except for V_k) in correspondence with the edge between V_i and its parent. Let this edge be (u_i, v_i) for some supernode V_i. Recall that since the edge (u_i, v_i) was in T, it represents the minimum cut between u_i and v_i. Therefore, since A_i represents a cut between u_i and v_i as well, we must have

$$w(A_i) \geq w'(u_i, v_i). \tag{5}$$

Thus

$$\sum_{i=1}^{k-1} w'(u_i, v_i) \leq \sum_{i=1}^{k-1} w(A_i) \leq 2(\frac{k-1}{k})w(A) = (2 - \frac{2}{k})w(A). \tag{6}$$

Note that (u_i, v_i) are arbitrary edges in T, so $\sum_{i=1}^{k-1} w'(u_i, v_i)$ is at least as large as the sum of the $k-1$ lightest edges in T and our proof is complete.

References

[1] Vazirani, Vijay. "Approximation Algorithms." New York: Springer, 2003.