CMPUT 675: Approximation Algorithms **Winter 2018**

Lecture 15 (March 6, 2018): Cut Problems

Lecturer: Mohammad R. Salavatipour *Scribe: Han Wang*

15.1 Max-flow / min-cut

Given a graph $G(V, E)$ with edge costs $c : E \to \mathbb{Q}^+$, and two vertices $s, t \in V$, the **min-cut problem** is to find a subset $S \subseteq V$ such that $s \in S$, $t \notin S$ and the total cost of edges in the cut $\delta(S)$ of S is minimized.

This problem can be solved in polynomial time using any algorithm to compute the max-flow and by max-flow/min-cut theorm.

15.2 Multiway cut

In the **multiway cut problem**, we are given k vertices, $s_1, s_2, ..., s_k$, called terminals, and asked to find a minimum cost set of edges whose removal would disconnect all terminals from each other.

In the case $k = 2$, this reduces to the min-cut problem. For $k \geq 3$ it is NP-hard. We will start by examining a $2(1 - \frac{1}{k})$-approximation for this problem. First, a definition:

Definition 1 *A set of edges is called s_i-cut if its removal separates s_i from the rest of terminals.*

Note: Considering all terminals in $S - \{s_i\}$ as one single terminal T, we can find minimum s_i-cut in polytime

Mulitway-cut Algorithm: Alg1

Input: Graph $G = (V, E)$, terminals $s_i \in V$, $i = 1...k$ and a cost $c_e \in \mathbb{Q}^+$ for each edge
Output: A minimum cost set of edges whose deletion ensures that no two terminals are connected
1. **for** $i \leftarrow 1$ to k **do**
2. Compute C_i, a minimum cost s_i-cut
3. Reorder the C_i's by cost (so that C_k is the most expensive)
4. **return** $C = \cup_{i=1}^{k-1} C_i$

Figure 15.1: Multi-way cut Algorithm

Theorem 1 *Algorithm 1 is a $(2 - \frac{2}{k}) -$ approximation for Multiway-cut problem.*

Proof. First note that C is a multi-way cut, since each terminal s_i ($i = 1...k - 1$) has been isolated from the rest by the s_i-cut C_i. Therefore no edges remain to connect the last terminal s_k to any others.

Assume A is an optimum multiway cut in G. Then $G - A$ has k-components (each with one s_i). Let the components be $G_1, G_2, ..., G_k$. Then we look at the edges among components, as figure **??** shows:

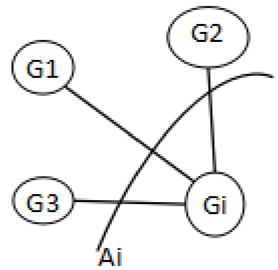

Figure 15.2: separate V into different components

Let $A_i = \delta(G_i, G - G_i)$, we say $A = \bigcup_{i=1}^{k} A_i$. Since each edge of A is incident at two of these components, each edge will be in two of the cuts A_i. Thus,

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

Since C_i is the munimum cut, we have $w(C_i) \leq w(A_i)$. Then

$$\sum_{i=1}^{k} w(C_i) \leq 2w(A)$$

This gives a factor 2 algorithm. Since C is obtained by discarding the heaviest of the cuts C_i,

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

■

The following example demonstrates that the approximation ratio in Theorem **??** is tight, for $0 < \epsilon < 1$:

Notice that Alg1 will, at each iteration, identify a $2 - \epsilon$ edge for the minimum cost s_i-cut, and therefore accrue a cost of $(2 - \epsilon)(k - 1)$. On the other hand, the optimal solution is to simply cut each edge in the cycle, for a total cost of k. The approximation ratio is $(2 - \epsilon)(1 - \frac{1}{k})$, and thus can be made arbitrarily close to the bound in Theorem **??**.

15.3 Min Steiner k-cut and min k-cut

Definition 2 *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 3 **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 $1 \leq i \leq k$.*

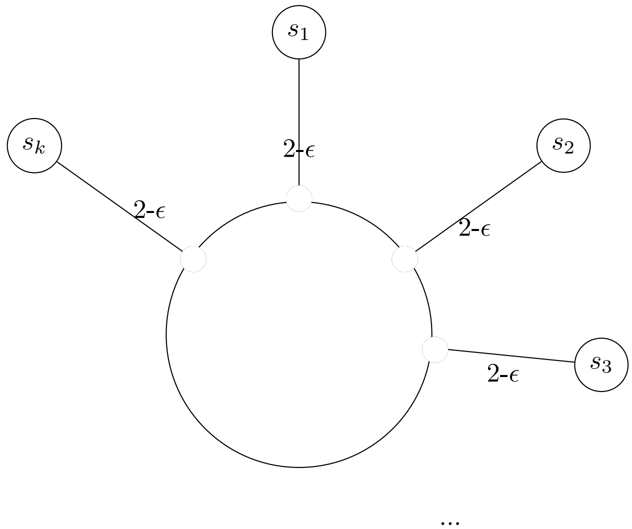

Figure 15.3: Example problem. Each vertex in a cycle of k nodes is connected with a terminal by an edge with weight $2-\epsilon$. Edges in the cycle all have weight 1. Alg1 will always select the more costly edges for removal.

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 1 *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 2 *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 2 *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, $1 \leq i \leq k$. Without loss of generality, assume that $w(A_1) \leq w(A_2) \leq \ldots \leq w(A_k)$. 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 $(1 \leq i \leq k)$ 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 $(1 \leq i \leq k-1)$, 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:

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

Therefore:

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

■

15.4 An LP approach to multi-way cut

We now consider an LP formulation of the multi-way cut problem. To do this, we will construct a set of subsets of vertices, $C_1, C_2, ... C_k \subseteq V$, such that (for $i = 1...k$) each s_i belongs to C_i (and to no others), analogous to the $G_1, G_2, ... G_k$ defined in the proof of Theorem **??**.

For $i = 1...k$ we define two indicator variables: let x_u^i indicate membership of vertex $u \in V$ in set C_i $(i = 1...k)$ and z_e^i membership of edge $e \in E$ in the cut of C_i :

$$x_u^i = \begin{cases} 1 & \text{if } u \in C_i \\ 0 & \text{otherwise} \end{cases} \qquad z_e^i = \begin{cases} 1 & \text{if } e \in \delta(C_i) \\ 0 & \text{otherwise} \end{cases}$$

The LP formulation is to minimize the total weight of selected edges (the objective):

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\sum_{e\in E} c_e \sum_{i=1}^{k} z_e^i & \\ \text{subject to} \quad & \sum_{i=1}^{k} x_u^i = 1, & \forall u \in V, \\ & z_e^i \geq x_u^i - x_v^i, & \forall e=(u,v)\in E, \\ & z_e^i \geq x_v^i - x_u^i, & \forall e=(u,v)\in E, \\ & x_{s_i}^i = 1, & i = 1,\dots,k, \\ & x_u^i \in \{0,1\}, & \forall u \in V, i=1,\dots,k. \end{aligned} \tag{15.1}$$

The first constraint ensures that each vertex belongs to one part. The second and third constraints are to ensure that for every edge, if two vertices are separated the edge is counted in the cut. And finally, each terminal is in a different part. If we recall the l_1 metric for $\mathbf{x}, \mathbf{y} \in \mathbb{R}^k$:

Definition 4 *Given $x, y \in \mathbb{R}^n$, the l_1 metric is a metric such that the distance between x, y is $||x-y||_1 = \sum_{i=1}^{n} |x^i - y^i|$.*

then this formulation can be simplified by thinking of each vertex in V as a point in $\mathbb{R}^k$ space: $\mathbf{x}_u = (x_u^1, x_u^2, ...x_u^k)$. The last constraint in **??** becomes:

$$x_{s_i}^i = 1 \implies \mathbf{x}_{s_i} = \underbrace{\mathbf{e}_i}_{\text{unit vector in ith dimension}}$$

The first constraint can be replaced with:

$$\sum_{i=1}^{k} x_u^i = 1 \implies \mathbf{x}_u \in \underbrace{\Delta_k}_{\text{kth simplex}} := \left\{ x \in \mathbb{R}^k | \sum_{i=1}^{k} x_u^i = 1 \right\}$$

Notice also that we have :

$$\sum_{i=1}^{k} z_e^i = \sum_{i=1}^{k} |x_u^i - x_v^i| = \|\mathbf{x}_u - \mathbf{x}_v\|_1$$

So we rephrase **??** as a simpler problem, in terms of these k-vectors:

$$\min \left\{ \frac{1}{2} \sum_{e=(u,v)\in E} c_e \|\mathbf{x}_u - \mathbf{x}_v\|_1 \right\} \tag{15.2}$$

$$\mathbf{x}_{s_i} = \mathbf{e}_i, \quad \forall i \in \{1, 2...k\} \tag{15.3}$$

$$\mathbf{x}_u \in \Delta_k, \quad \forall u \in V \tag{15.4}$$

15.4.1 An example

Now consider an example:

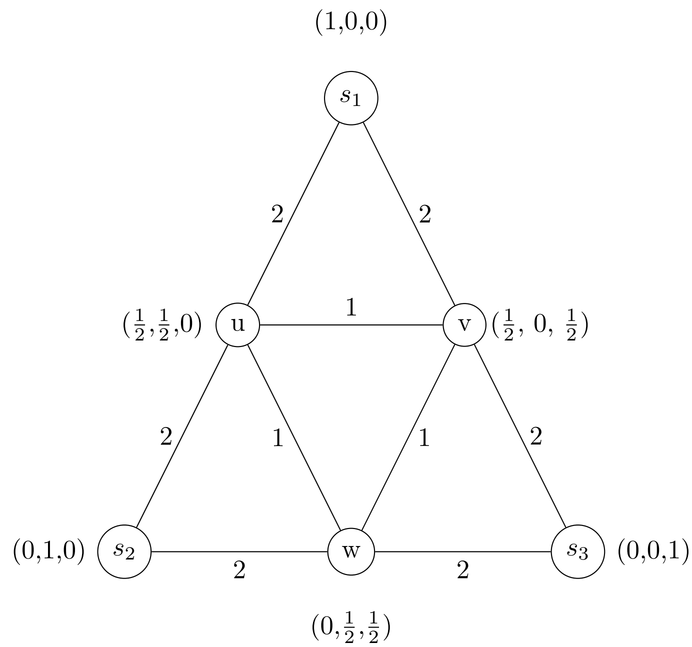

Figure 15.4: Example problem. We have three terminals, so the LP relaxation for the problem assigns a 3-vector to each node in the graph (in brackets)

.

An optimal solution would be to cut the edges (s_1, u), (s_1, v), as well as (s_2, u), (s_2, w), for a total cost of 8. On the other hand, the total cost weight for the solution to the relaxed LP problem is 7.5.