CMPUT 675: Approximation Algorithms **Fall 2015**

Lecture 10 (Oct 1st, 2015): Cut Problems

Lecturer: Mohammad R. Salavatipour *Scribe: Dean Koch*

10.1 Cut problems

10.1.1 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

10.1.2 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:

A set of edges is called an **s_i-cut** if they separate s_i from the other terminals in the current graph

10.1.3 An approximation for multi-way cut

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 10.1: Multi-way cut Algorithm

Theorem 1 *this is a $2(1 - \frac{1}{k})$-approximation*

Proof. First I 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.

Suppose that A is an optimal solution. Note that $G - A$ has k disconnected components, $G_1, G_2, ...G_k$ and the terminals $s_1, s_2, ...s_k$ each belong to exactly one of these (see Figure 10.2).

Denote by A_i the set of edges in A which form the cut of G_i $(i = 1...k)$:

$$A_i := \delta(G_i)$$

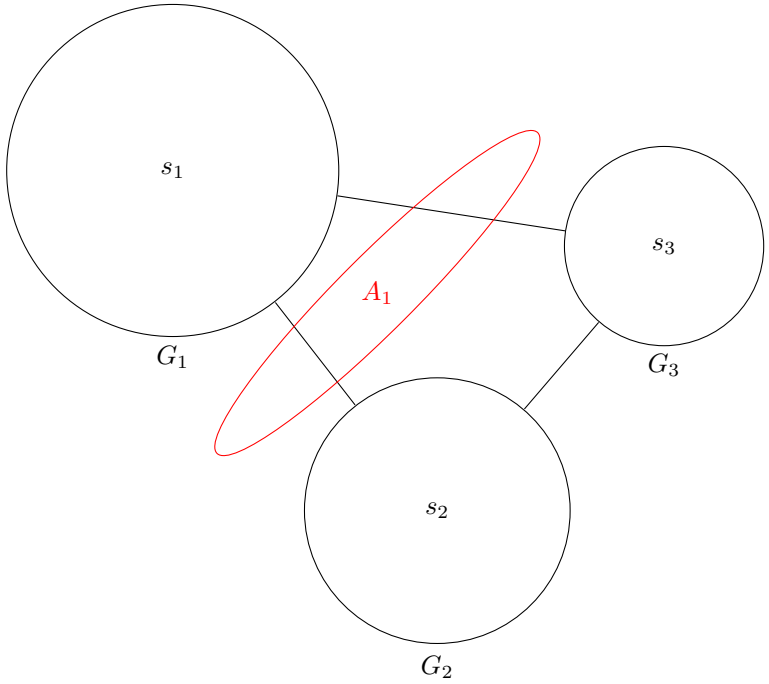

Figure 10.2: Illustration of the vertex sets G_i and an edge set A_1 in an optimal solution

Thus we have $A = \cup_{i=1}^{k} A_i$. Observing that every edge in A belongs to exactly two of these sets A_i, we have:

$$\sum_{i=1}^{k} w(A_i) := 2\text{OPT}$$

Since C_i is a minimum cost s_i-cut, we have the property that $w(C_i) \leq w(A_i)$ for every $i \in \{1, 2, ...k\}$. Furthermore, the costliest cut C_k must have cost no less than the average among all the k. Therefore, the combined cost of C is no more than $1 - \frac{1}{k}$ times the total cost of all sets identified in the for-loop. This yields:

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

■

10.1.4 An example for Alg1

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

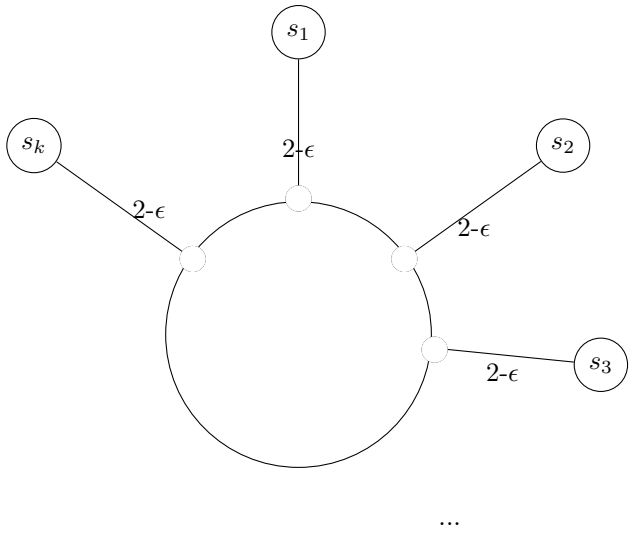

Figure 10.3: Example problem. Each vertex in a cycle of k nodes is connected with a terminal by an edge with weight $2 - \epsilon$. Edges in the cycle all have weight 1. Alg1 will always select the more costly edges for removal.

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 1.

10.1.5 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 1.

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):

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2} \sum_{e \in E} c_e \sum_{i=1}^{k} z_e^i \\ \text{subject to} \quad & \sum_{i=1}^{k} x_u^i = 1, && \forall u \in V, \\ & z_e^i \geq x_u^i - x_v^i, && \forall e = (u,v) \in E, \\ & z_e^i \geq x_v^i - x_u^i, && \forall e = (u,v) \in E, \\ & x_{s_i}^i = 1, && i = 1, \ldots, k, \\ & x_u^i \in \{0,1\}, && \forall u \in V, i = 1, \ldots, k. \end{aligned} \tag{10.1}$$

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 $\mathbf{x}, \mathbf{y} \in \mathbb{R}^k$, defined by:

$$\|\mathbf{x} - \mathbf{y}\|_1 = \sum_{i=1}^{k} |x_i - y_i|$$

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 10.1 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 10.1 as a simpler problem, in terms of these k-vectors:

$$\min \left\{ \frac{1}{2} \sum_{e=(u,v)\in E} c_e \|\mathbf{x}_u - \mathbf{x}_v\|_1 \right\} \tag{10.2}$$

$$\mathbf{x}_{s_i} = \mathbf{e}_i, \quad \forall i \in \{1, 2...k\} \tag{10.3}$$

$$\mathbf{x}_u \in \Delta_k, \quad \forall u \in V \tag{10.4}$$

10.1.6 An example

Now consider an example:

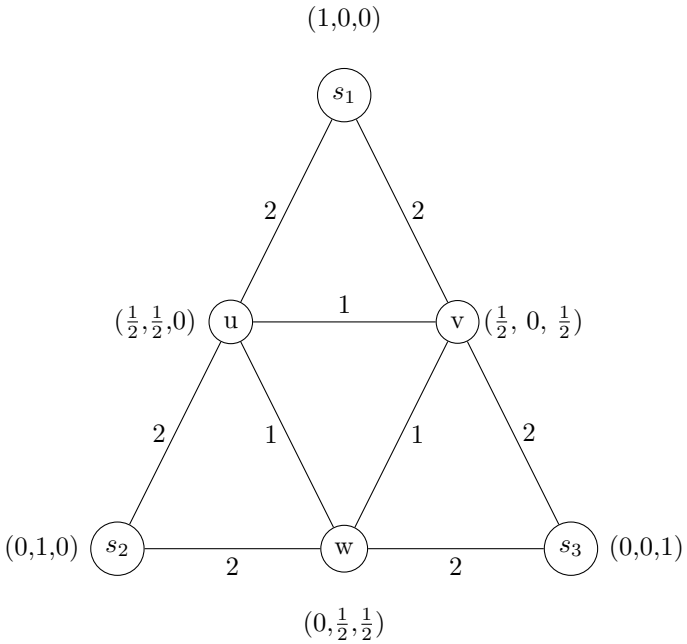

Figure 10.4: Example problem. We have three terminals, so the LP relaxation for the problem assigns a 3-vector to each node in the graph (in brackets)

.

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.

10.1.7 An randomization algorithm for multi-way cut with the LP relaxation

We now consider a method of converting the fractional solution to the LP relaxation into an integer solution, using random assignments. First, we define a neighbourhood on our graphs using the l_1 metric:

Define the ball $B(s_i, r)$ to be the set of vertices within radius $\frac{1}{2}r$ about vertex s_i:

$$B(s_i, r) = \left\{ u \in V \text{ such that } \frac{1}{2}\|\mathbf{x}_{s_i} - \mathbf{x}_u\|_1 \leq r \right\}$$

Observe that for any solution to (10.2-10.4) we have $B(s_i, 1) = V$ (for all $i = 1...k$). This is because $\mathbf{x}_u$ lies in Δ_k, and so the sum of the differences of its components with the unit vector $\mathbf{e}_j$ can be written $\|\mathbf{x}_{s_i} - \mathbf{x}_u\|_1 = \sum_{i \neq j} + |1 - x_u^j| \leq 1 + 1 = 2$.

Now we are ready to state the algorithm:

Algorithm2: Randomized Rounding for Multiway Cut

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. Let $\mathbf{x}$ be an optimal (fractional) solution to the LP
2. $C_i \leftarrow \varnothing$ for $i = 1...k$
3. pick $r \in (0, 1)$ uniformly at random
4. pick a random permutation of terminals, π
5. **for** $i = 1$ to $k - 1$, **do:**
6. $\quad C_{\pi(i)} \leftarrow B(s_{\pi(i)}, r) - \cup_{j<i} C_{\pi(j)}$
7. $C_{\pi(k)} \leftarrow V - \cup_{j<k} C_{\pi(j)}$
8. **return** $F = \cup_{i=1}^{k} \delta(C_i)$

Figure 10.5: Multi-way cut Algorithm # 2

Before starting the analysis, we consider a special case to get a feel for the algorithm. Consider a part of the graph with the following structure, and LP assignment:

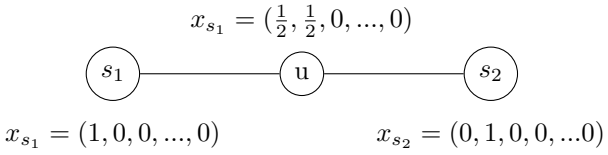

Figure 10.6: Special case for Alg2. LP (fractional) solutions are indicated in brackets

.

Note that we have three possible outcomes here: either $u \in B(s_1, r)$ (and u is assigned to C_1), or $u \in B(s_2, r)$ (and u is assigned to C_2, as long as $u \notin B(s_1, r)$), or neither (in which case u is assigned to $C_{\pi(k)}$).

If s_1 comes before s_2 (in the permutation), and s_2 is not the last terminal ($\pi(k) \neq 2$) then (u, s_2) enters the cut with probability 1. This is because either $r \in (0, \frac{1}{2})$, in which case:

$$u \in B(s_1, r) \implies u \in C_1 \text{ and } s_2 \notin B(s_1, r) \implies s_2 \in C_2$$

Or, $r \in [\frac{1}{2}, 1)$, in which case (u, v) also gets cut, since

$$s_2 \in B(s_2, r) \implies s_2 \in C_2 \text{ and } u \notin B(s_2, r) \implies u \in C_1$$

In the next lecture, we will see how to extend this line of reasoning to the general case, and attain a bound of $\frac{3}{4}\|\mathbf{x}_u - \mathbf{x}_v\|_1$ on the probability of any edge (u, v) belonging to a cut.