Linear Programming Algorithm
for the Multiway Cut Problem

Adam Groce
18.434: Seminar in Theoretical Computer Science
Prof. M. X. Goemans
March 2, 2006

1 The Problem

Recall from last time the multiway cut problem: given a graph with weighted edges and a set of terminals \( S = \{s_1, s_2, \ldots, s_k\} \subseteq V \), find the minimum weight set of edges \( E' \subseteq E \) which, when removed, leaves all terminals separated from all other terminals. Last time, a combinatorial algorithm was given with an approximation factor of \( 2 - \frac{2}{k} \). This lecture will show a randomized linear programming algorithm with an approximation factor of \( \frac{3}{2} \).

2 The Linear Programming Relaxation

Let \( \Delta_k \) denote the \( k - 1 \) dimensional simplex; that is, the surface in \( \mathbb{R}^k \) defined by \( \{x \in \mathbb{R}^k | x \geq 0 \text{ and } \sum_i x^i = 1\} \), where \( x \) is a vector and \( x^i \) is the \( i \)th coordinate of \( x \). The LP relaxation will map each vertex of \( G \) to a point in \( \Delta_k \). Each terminal will be mapped to a different unit vector. Let \( x_v \) represent the point to which vertex \( v \) is mapped. Define the length of an edge \( (u, v) \) to be

\[
d(u, v) = \frac{1}{2} \sum_{i=1}^{k} |x_u^i - x_v^i|
\]
Now consider the relaxation:

\[
\begin{align*}
\text{minimize} & \quad \sum_{(u,v) \in E} c(u,v)d(u,v) \\
\text{subject to} & \quad d(u,v) = \frac{1}{2} \sum_{i=1}^{k} |x^i_u - x^i_v|, \quad (u,v) \in E \\
& \quad x_v \in \Delta_k, \quad v \in V \\
& \quad x_{s_i} = e_i, \quad s_i \in S
\end{align*}
\]

An integer solution to this relaxation maps each vertex of \( G \) to a unit vector. Each vertex represents a component of the graph after \( E' \) is removed. Edges within one component have length 0, and edges between components (i.e., those in \( E' \)) have length 1. The function being minimized is therefore equal to the cost of \( E' \).

It is not clear that the above is a true linear program, due to the absolute values. However, this is not a problem. To create an equivalent true linear program, replace the first constraint with:

\[
\begin{align*}
& \quad x^i_{uv} \geq x^i_u - x^i_v, \quad 1 \leq i \leq k \\
& \quad x^i_{uv} \geq x^i_v - x^i_u, \quad 1 \leq i \leq k \\
& \quad d(u,v) = \frac{1}{2} \sum_{i=1}^{k} x^i_{uv}
\end{align*}
\]

Because of the minimization, any optimal solution must satisfy \( x^i_{uv} = |x^i_u - x^i_v| \).

We may assume, without loss of generality, that for each edge \((u,v) \in E\), \(x_u\) and \(x_v\) differ in at most two coordinates.

**Proof.** Along any edge \((u,v)\) where \(x_u\) and \(x_v\) differ in more than two coordinates, insert a new vertex \(w\) and replace \((u,v)\) with \((u,w)\) and \((w,v)\). Assign both \((u,w)\) and \((w,v)\) the same cost as \((u,v)\). This does not change the cost of the optimal integral solution.

Now consider the optimal fractional solution. Since \(d\) is a valid distance function, \(d(u,w) + d(w,v) \geq d(u,v)\). Therefore the cost of the optimal solution cannot decrease because of the addition of \(w\). Now, let \(i\) be the coordinate in which the difference between \(x^i_u\) and \(x^i_v\) is minimal (disregarding coordinates where \(x^i_u = x^i_v\)). Without loss of generality assume \(x^i_u < x^i_v\) and
let $\alpha = x^i_v - x^i_u$. There must be a coordinate $j$ such that $x^j_u \geq x^j_v + \alpha$. Consider the solution with $x^i_w = x^i_u$ and $x^j_w = x^j_v + \alpha$. All other coordinates of $x^i_w$ are equal to those of $x^i_v$. This gives $x^i_w \in \Delta_k$ and $d(u, v) = d(u, w) + d(w, v)$.

$x^i_v$ and $x^i_w$ differ in only two coordinates, and $x^j_w$ and $x^j_u$ differ in fewer coordinates than $x^i_v$ and $x^i_u$. Repeated application of this process will give a solution with the same cost and with the desired property. 

3 The Algorithm

Take an optimal solution to the relaxation with edges whose endpoints differ in at most two coordinates, and let $OPT$ denote its cost. Define $E_i = (u, v) \in E | x^i_u \neq x^i_v$. (Note that each edge will lie in two of these sets.) Define $W_i = \sum_{e \in E_i} c(e)d(e)$. Without loss of generality, assume that $W_k$ is the greatest of $W_1, \ldots, W_k$. Also define $B(s, \rho) = v \in V | x^i_v \geq \rho$.

The algorithm operates as follows. First, pick $\rho$ at random in $(0, 1)$ and an ordering $\sigma$ from $(1, 2, \ldots, k - 1, k)$ and $(k - 1, k - 2, \ldots, 1, k)$. Then partition $V$ into $V_1, \ldots, V_k$ as follows. Proceed in the order given by $\sigma$. Each $V_i$ should contain all vertices in $B(s_i, \rho)$ that have not already been assigned to a previous $V_i$. At the end, assign all unused vertices to $V_k$. The sets $V_1, \ldots, V_k$ are the components after removing the cut, and edges between vertices in two different sets are in the cut.

More formally, the algorithm is:

1. Compute an optimal solution to relaxation.
2. Renumber the terminals so that $W_k$ is largest among $W_1, \ldots, W_k$.
3. Pick uniformly at random $\rho \in (0, 1)$ and $\sigma \in (1, 2, \ldots, k - 1, k), (k - 1, k - 2, \ldots, 1, k)$.
4. For $i = 1$ to $k - 1$: $V_{\sigma(i)} \leftarrow B(s_i, \rho) - \bigcup_{j<i} V_{\sigma(j)}$.
5. $V_k \leftarrow V - \bigcup_{i<k} V_i$.
6. Let $C$ be the set of edges that run between sets in the partition $V_1, \ldots, V_k$. Output $C$. 

3
4 Proof of the Approximation Factor

Let $C$ be the cut produced by the algorithm, $c(C)$ be the cost of $C$, and $OPT$ be the cost of the optimal solution to the linear program. We will show that $E[c(C)]$, the expected value of $c(C)$, is at most $(1.5 - \frac{1}{k}) \times OPT$.

Lemma 1. If $e \in E_k$, then $Pr[e \in C] \leq d(e)$.

Proof. The endpoints of $E$ differ in coordinates $i$ and $k$. Since $V_k$ is determined without considering the coordinates of the points left over, and all coordinates except $i$ and $k$ are equal, the only way that endpoints $u$ and $v$ will end up in different sets is if one (but not the other) is in $V_i$. This occurs if and only if $\rho$ is between $x^i_u$ and $x^i_v$. This has probability $d(e)$.

Lemma 2. If $e \in E - E_k$, then $Pr[e \in C] \leq 1.5d(e)$.

Proof. The endpoints of $E$, $u$ and $v$, differ in coordinates $i$ and $j$. Let $\beta$ be the interval $[x^i_u, x^i_v]$ and let $\alpha$ be the part of $[x^i_u, x^i_v]$ that does not overlap with $\beta$. $u$ and $v$ can each end up in either $V_i$, $V_j$, or $V_k$. Assume without loss of generality that $\alpha$ is closer to 0 than $\beta$. (If not, switching the values of $i$ and $j$ makes it so.) $u$ and $v$ end up in different sets if and only if $\rho \in \beta$, or $\rho \in \alpha$ and $\sigma(i) < \sigma(j)$. Therefore $Pr[e \in C] = |\beta| + \frac{|\alpha|}{2} \leq 1.5d(e)$ (because $|\alpha| \leq |\beta| = d(e)$). □

Theorem 1. $E[c(C)] \leq (1.5 - \frac{1}{k}) \times OPT$.

Proof. First, note that $\sum_{i=1}^{k} W_i = 2 \cdot OPT$, and $W_k$ was chosen to be the greatest, so $W_k \geq \frac{2}{k} \cdot OPT$.

$$E[c(C)] = \sum_{e \in E} c(e)Pr[e \in C] = \sum_{e \in E - E_k} c(e)Pr[e \in C] + \sum_{e \in E_k} c(e)Pr[e \in C]$$

$$\leq 1.5 \sum_{e \in E - E_k} c(e)d(e) + \sum_{e \in E_k} c(e)d(e) = 1.5 \sum_{e \in E} c(e)d(e) - 0.5 \sum_{e \in E_k} c(e)d(e)$$

$$\leq 1.5 \cdot OPT - 0.5 \left( \frac{2}{k} \cdot OPT \right)$$

$$\leq (1.5 - \frac{1}{k}) \cdot OPT$$

□