Saarland Informatics Campus

Script of Lecture 8, Approximation Algorithms
Summer term 2017

Tobias Mömke, Hang Zhou
`http://www-cc.cs.uni-saarland.de/course/61/`

Written by Hang Zhou

1 The Multicut Problem

Input: an undirected graph $G = (V, E)$ with nonnegative edge costs $c_e \geq 0$ for all $e \in E$, and k pairs of vertices $(s_1, t_1), \ldots, (s_k, t_k)$.

Output: a minimum-cost set of edges F whose removal disconnects all pairs, i.e., for each $1 \leq i \leq k$, there is no path connecting s_i and t_i in the graph $(V, E - F)$.

2 Programs

For each edge $e \in E$, create a variable $x_e \in \{0, 1\}$, such that $x_e = 1$ if e is in the solution F, and $x_e = 0$ otherwise.

Let $\mathcal{P}_i$ be the set of paths P connecting s_i and t_i.

Integer programming formulation:

$$\begin{aligned} \text{minimize} \sum_{e \in E} c_e x_e & \\ \text{subject to} \sum_{e \in P} x_e \geq 1, & \qquad \forall P \in \mathcal{P}_i, 1 \leq i \leq k, \\ x_e \in \{0, 1\}, & \qquad \forall e \in E. \end{aligned}$$

Linear programming relaxation: replace the constraints $x_e \in \{0, 1\}$ by $x_e \geq 0$.

This LP can be solved in polynomial time, because we have a polynomial-time separation oracle.

Separation oracle: Let x be any given solution. Consider the graph G where each edge e has length x_e. For each $1 \leq i \leq k$, compute a shortest path between s_i and t_i. If for some i, the shortest path has length less than 1, then we return this path as a violated constraint; if for all i, the shortest path has length at least 1, then the solution x is feasible.

3 Rounding

Idea: grow a region centered at some vertex s_i, cut all edges at the boundary of this region, remove all vertices inside this region from the graph and repeat.

Let x be an optimal solution to the LP. We define a distance metric d, where $d(u, v)$ is the length of the shortest path from vertex u to v using $\{x_e\}_e$ as edge lengths. Then d satisfies the triangle inequality. Also, $d(s_i, t_i) \geq 1$ for all i (LP constraints). Let $B(s_i, r)$ be the ball of radius r around vertex s_i, i.e., $B(s_i, r) = \{v \in V : d(s_i, v) \leq r\}$.

Let $V^* = \sum_{e \in E} c_e x_e$ be the optimal value of the LP. V^* can be viewed as the total *volume*.

Define:

$$V(s_i, r) = \frac{V^*}{k} + \sum_{e=(u,v):u,v \in B(s_i,r)} c_e x_e + \sum_{e=(u,v):u \in B(s_i,r), v \notin B(s_i,r)} c_e(r - d(s_i, u)).$$

$V(s_i, r)$ can be viewed as the volume of the ball $B(s_i, r)$.

For any set of vertices S, let $\delta(S)$ be the set of all edges that have exactly one endpoint in the set S. For a given radius r, let $c(\delta(B(s_i, r)))$ be the cost of the edges in $\delta(B(s_i, r))$. We can show that there is a ball of radius $r < 1/2$ such that the cost of the cut induced by the ball is not much larger than the volume of this ball. Finding such a ball is called *region growing*.

Lemma 1. *For any s_i one can find in polynomial time a radius $r < 1/2$ such that*

$$c(\delta(B(s_i, r))) \leq (2 \ln(k + 1))V(s_i, r).$$

Algorithm 1 Rounding algorithm for Multicut

$x \leftarrow$ an optimal solution to the LP
$F \leftarrow \emptyset$
for $i \leftarrow 1$ to k **do**
  if s_i and t_i are connected in $(V, E - F)$ **then**
    Choose radius $r < 1/2$ around s_i as in Lemma 1
    $F \leftarrow F \cup \delta(B(s_i, r))$
    Remove $B(s_i, r)$ and incident edges from graph
return F

Note that the balls $B(s_i, r)$ and the volumes $V(s_i, r)$ are with respect to the edges and the vertices remaining in the current graph.

Correctness: We only need to show that for every pair (s_j, t_j), there does not exist a ball $B(s_i, r)$ containing both s_j and t_j. If both s_j and t_j belong to some ball $B(s_i, r)$, since $r < 1/2$, we have $d(s_i, s_j) < 1/2$ and $d(s_i, t_j) < 1/2$. By the triangle inequality, we have $d(s_j, t_j) < 1$, which contradicts the LP constraint.

Approximation Ratio:

Theorem 1. *Algorithm 1 is a $(4 \ln(k + 1))$-approximation algorithm for the multicut problem.*

Proof of Theorem 1 using Lemma 1. Let B_i be the set of vertices in the ball $B(s_i, r)$ chosen by the algorithm. Let F_i be the edges in $\delta(B_i)$. Then $F = \bigcup_{i=1}^{k} F_i$. Let V_i be the volume of the edges removed when the vertices in B_i and their incident edges are removed from the graph. Then $V_i \geq V(s_i, r) - \frac{V^*}{k}$. So we have:

$$\begin{aligned} c(F_i) &\leq (2 \ln(k + 1))V(s_i, r) && \text{(Lemma 1)} \\ &\leq (2 \ln(k + 1))(V_i + \frac{V^*}{k}). \end{aligned}$$

We also observe that each edge belongs to at most one V_i, so $\sum_{i=1}^{k} V_i \leq V^*$.

Therefore, we have:

$$\begin{aligned} \sum_{e \in F} c_e &= \sum_{i=1}^{k} \sum_{e \in F_i} c_e \\ &\leq (2 \ln(k + 1)) \sum_{i=1}^{k} \left(V_i + \frac{V^*}{k} \right) \\ &\leq (4 \ln(k + 1))V^* \\ &\leq (4 \ln(k + 1))\text{OPT}. \end{aligned}$$

□

Now it only remains to prove Lemma 1. We need the following fact from calculus.

Fact 1. *(Mean Value Theorem) For a function f that is differentiable on the interval (a, b), there exists some $c \in (a, b)$ such that*

$$f'(c) = \frac{f(b) - f(a)}{b - a}.$$

Proof of Lemma 1. For simplicity, we let $V(r) = V(s_i, r)$ and $c(r) = c(\delta(B(s_i, r)))$. First, we show that when r is chosen uniformly at random from $[0, 1/2)$, the expectation of $c(r)/V(r)$ is at most $2\ln(k+1)$. Afterwards, we use a deterministic algorithm to find a radius r such that $c(r) \leq (2\ln(k+1))V(r)$.

A key observation is that the derivative of $V(r)$ is the cost of the edges in the cut of the ball of radius r around s_i, i.e., $\frac{dV}{dr} = c(r)$, as soon as $V(r)$ is differentiable.

To convey the intuition, first, suppose $V(r)$ is differentiable on every point in $[0, 1/2)$. Define $f(r) := \ln V(r)$. Then we have

$$f'(r) = \frac{\frac{d}{dr}V(r)}{V(r)} = \frac{c(r)}{V(r)}.$$

Since $V(0) = \frac{V^*}{k}$ and $V(1/2) \leq V^* + \frac{V^*}{k}$, by the Mean Value Theorem (Fact 1), there is some $r \in (0, 1/2)$, such that

$$f'(r) = \frac{\ln V(1/2) - \ln V(0)}{1/2 - 0} = 2\ln\left(\frac{V(1/2)}{V(0)}\right) \leq 2\ln\left(\frac{V^* + \frac{V^*}{k}}{V^*/k}\right) = 2\ln(k+1). \tag{1}$$

However, $V(r)$ is not always differentiable (at points $r = d(s_i, v)$ for some vertex v). Therefore, we need to pay more attention in the proof.

We sort the vertices in $B(s_i, 1/2)$ in increasing order of their distance from s_i, and re-index the vertices, such that

$$0 = r_0 \leq r_1 \leq \cdots \leq r_{l-1} \leq 1/2,$$

where $r_j = d(s_i, v_j)$. Let r_j^- be a value infinitesimally smaller than r_j.

The expected value of $c(r)/V(r)$ for $r \in [0, 1/2)$ is

$$\begin{aligned}
2\sum_{j=0}^{l-1}\int_{r_j}^{r_{j+1}^-}\frac{c(r)}{V(r)}dr &= 2\sum_{j=0}^{l-1}\int_{r_j}^{r_{j+1}^-}\frac{1}{V(r)}\frac{dV}{dr}dr \\
&= 2\sum_{j=0}^{l-1}\Big[\ln V(r)\Big]_{r_j}^{r_{j+1}^-} \\
&= 2\sum_{j=0}^{l-1}\Big[\ln V(r_{j+1}^-) - \ln V(r_j)\Big] \\
&\leq 2\sum_{j=0}^{l-1}\Big[\ln V(r_{j+1}) - \ln V(r_j)\Big] && (V(r) \text{ is non-decreasing}) \\
&= 2(\ln V(1/2) - \ln V(0)) \\
&= 2\ln(k+1). && (\textit{Eq.}\ (1))
\end{aligned}$$

So there must be some $r \in [0, 1/2)$ such that $c(r) \leq (2\ln(k+1))V(r)$. We can show that such an r can be found deterministically in polynomial time (exercise). □

Theorem 2 (hardness)**.** *Assuming the unique games conjecture, for any constant $\alpha \geq 1$, there is no α-approximation algorithm for the multicut problem unless $P = NP$.*