CMPUT 675: Topics on Approximation Algorithms and Approximability **Fall 2013**

Lecture 10 and 11 (Oct 8,10): Multiway Cut, Multicut

Lecturer: Mohammad R. Salavatipour *Scribe: Yaochen Hu and older notes*

10.1 Multiway Cut

Definition 10.1 *In a Graph $G(V,E)$ with a cost measure on edges $C : E \to \mathbb{R}^+$, given two vertices $s, t \in V$, a cut is a partition of V $(S, V\backslash S)$, s.t. $s \in S, t \in V\backslash S$; the weight of the cut is the sum of C_e across S and $V\backslash S$.*

Using Max-Flow-Min-Cut, we can solve the Min-cut Problem in polynomial time.

Definition 10.2 (Multiway Cut) *In a Graph $G(V,E)$ with a cost measure on edges $c : E \to \mathbb{R}^+$, given $\{s_1, s_2, ..., s_k\} \subset V$ terminals, the Multiway Cut Problem is to find the collectin of edges whose removal will separate all these terminals with minimum cost (i.e. a set of edges whose removal disconnect each pair s_i, s_j).*

This problem is NP-complete with $K \geq 3$. The first approximation algorithm we present for this problem is a simple greedy method that uses minimum $s - t$-cut as a subroutine.

10.1.1 Algorithm

Multiway Cut Greedy Algorithm

Input: A graph $G(V,E)$ with a cost measure on edges $c : E \to \mathbb{R}^+$, and a set of terminals $\{s_1, s_2, \ldots, s_k\} \subset V$.
Output: A collection of edges C which separate the terminals.
1. **For** $i \leftarrow$ **to** k **do**
2. find a min s_i-cut (separate s_i from all the other terminals), and store it in C_i
3. **end**
4. find the C_β with the max cost among all the C_i
5. $C \leftarrow \cup_{i \neq \beta} C_i$
6. **return** C

Figure 10.1: Greedy Algorithm

10.1.2 Algorithm Analysis

Theorem 10.3 *The greedy algorithm gives a $2 - \frac{2}{k}$-approximation for the Multiway Cut Problem.*

Proof: It is straightforward that this algorithm gives a feasible solution. We need to show the approximation ratio.

Without loss of generality, assuming that the C_k has the maximum cost, we have $C = \cup_{i=1}^{k-1} C_i$. Let A be an optimal solution. In that solution, the graph is cut into k subgraphs and each of them containing exact one terminal S_i. Let G_i be the subgraph containing terminal S_i and let A_i be the edges coming out of G_i (going to $G - G_i$). Now we have

$$A = \cup_{i=1}^{k} A_i. \tag{10.1}$$

Since each edge of A belongs to exactly two A_i, we have

$$\sum_{i=i}^{k} C(A_i) = 2C(A). \tag{10.2}$$

For every terminal S_i, since C_i is the minimum cut separating S_i from the rest, we have $C(C_i) \leq C(A_i)$. Therefore, we have

$$\sum_{i=1}^{k} C(C_i) \leq \sum_{i=1}^{k} C(A_i) \leq 2C(A). \tag{10.3}$$

Since we through away the C_k with the maximum cost, then we get

$$C(C) \leq (2 - \frac{2}{k})C(A) = (2 - \frac{2}{k})opt. \tag{10.4}$$

■

This approximation ratio is tight. Figure (10.2) shows a tight example which is basically k vetices are on a cycle, and every terminal connects to one of the nodes on the cycle. The edge between two nodes on the cycle has the cost of 1, and the edge from the terminal to the node on cycle has the cost $2 - \epsilon$, where ϵ is an arbitrary small positive value. In this case, the cost of the optimal solution is exactly k, while the greedy solution will give a cost of $(2 - \epsilon)(k - 1)$.

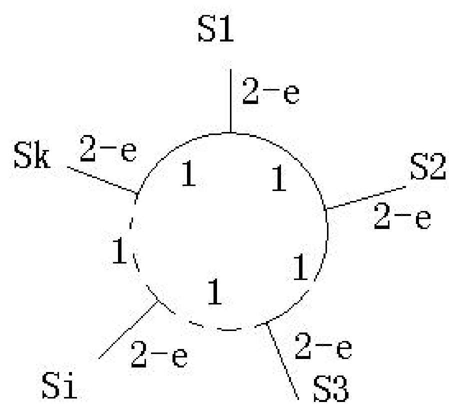

Figure 10.2: A tight example for the greedy algorithm

10.2 An anlgorithm based on Randomized Rounding of an LP

The natural LP relaxation for Multiway cut has a bad integrality gap. So we present a different LP relxation and show how rounding this LP yields a better approximation for this problem. Another way of looking at the multiway cut problem is finding an optimal partition of V, say $V_1, V_2, \ldots, V_k$, such that $s_i \in V_i, i = 1, 2, \ldots, k$ and the cost of $\cup_{i=1}^k \delta(V_i)$ is minimized.

To formulate the problem as an integer program, we need to define some sets of variables. For each vertex $v \in V$, we have k boolean variables x_v^i such that $x_v^i = 1$ if and only if v is assigned to the set V_i. For each edge $e \in E$, we create a boolean variable z_e^i such that $z_e^i = 1$ if and only if $e \in \delta(V_i)$. Since if $e \in \delta(V_i)$, it is also the case that $e \in \delta(V_j)$ for some $j \neq i$, the objective function of the integer program is then

$$\frac{1}{2} \sum_{e \in E} c_e \sum_{i=1}^{k} z_e^i.$$

Now we consider the constraints for the integer program. Obviously, we have $x_{s_i}^i = 1, i = 1, \ldots, k$ since each s_i must be assigned to V_i and we can also have $\sum_{i=1}^k x_u^i = 1$ for any vertex $u \in V$ since u must be contained in some V_i. Because for any edge $e = (u, v)$, $e \in \delta(V_i)$ if and only if exactly one of its endpoints is in V_i, we have $z_e^i \geq |x_u^i - x_v^i|$. Then the overall integer program is as follows:

$$\begin{aligned}
\text{minimize} \quad & \frac{1}{2} \sum_{e \in E} c_e \sum_{i=1}^{k} z_e^i & \\
\text{subject to} \quad & \sum_{i=1}^{k} x_u^i = 1, & \forall u \in V, \\
& z_e^i \geq x_u^i - x_v^i, & \forall e = (u,v) \in E, \\
& z_e^i \geq x_v^i - x_u^i, & \forall e = (u,v) \in E, \\
& x_{s_i}^i = 1, & i = 1, \ldots, k, \\
& x_u^i \in \{0, 1\}, & \forall u \in V, i = 1, \ldots, k.
\end{aligned} \tag{10.5}$$

Since the relaxed linear program of this integer program is closely related with the l_1-metric for measuring distances in Euclidean space, we give the definition of l_1-metric below.

Definition 10.4 *l_1-metric is a metric space where for any $x = (x^1, \ldots, x^n), y = (y^1, \ldots, y^n) \in \mathbb{R}^n$ the distance between them is $||x - y||_1 = \sum_{i=1}^n |x^i - y^i|$.*

Let Δ_k denote the $k-1$ dimensional simplex, that is, the surface in $\mathbb{R}^k$ defined by $\{x \in \mathbb{R}^k | x \geq 0 \;\&\; \sum_{i=1}^k x^i = 1\}$, where x is a vector and x^i is the ith coordinate of x. The LP relaxation will map each vertex of G to a point in Δ_k, and especially map each terminal to a unit vector. Let x_v represent the point to which vertex v is mapped. Thus, the relaxed linear program is as follows:

$$\begin{aligned}
\text{minimize} \quad & \frac{1}{2} \sum_{e=(u,v) \in E} c_e ||x_u - x_v||_1 & \\
\text{subject to} \quad & x_v \in \Delta_k, & \forall v \in V, \\
& x_{s_i} = e_i, & i = 1, \ldots, k,
\end{aligned} \tag{10.6}$$

For any $r \in [0, 1]$ and $1 \leq i \leq k$, let $B(s_i, r)$ be the set of vertices corresponding to the points x_v in a ball of radius r around s_i under the measure of l_1-metric, that is, $B(s_i, r) = \{v \in V | \frac{1}{2}||s_i - x_v||_1 \leq r\}$.

Here, $\delta(C_i)$ is the cutting set separating C_i from the rest.

Multiway Cut LP Rounding Algorithm

Input: A graph $G(V, E)$ with a cost measure on edges $c : E \to \mathbb{R}^+$, and a set of terminals $\{s_1, s_2, \ldots, s_k\} \subset V$.
Output: A collection of edges F which separate the terminals.
1. Find the fractional solution for the LP in (**??**)
2. **For** $i = 1$ **to** k**, do**
3. $\quad C_i \leftarrow \emptyset$
4. **end**
5. Uniformly randomly pick $r \in (0, 1)$
6. Pick a random permutation π of $\{1, 2, ..., k\}$
7. **For** $i = 1$ **to** $k - 1$**, do**
8. $\quad C_{\pi_i} \leftarrow B(s_{\pi_i}, r) - \cup_{j<i} C_{\pi_j}$
9. **end**
10. $C_{\pi_k} \leftarrow V - \cup_{i=1}^{k-1} C_{\pi_i}$
11. **return** $F = \cup_{i=1}^{k} \delta(C_i)$

Figure 10.3: LP Rounding Algorithm

Theorem 10.5 *The randomized-LP-rounding algorithm is a $\frac{3}{2}$-approximation algorithm.*

To prove this theorem, we need to introduce some useful lemmas first.

Lemma 10.6 *$\forall u, v \in V$ and any index l, $|x_u^l - x_v^l| \leq \frac{1}{2}||x_u - x_v||_1$.*

Proof: Without loss of generality, assume that $x_u^l \geq x_v^l$. Then

$$
\begin{aligned}
|x_u^l - x_v^l| &= x_u^l - x_v^l = (1 - \sum_{j \neq l} x_u^j) - (1 - \sum_{j \neq l} x_v^j) \\
&= \sum_{j \neq l} (x_u^j - x_v^j) \\
&\leq \sum_{j \neq l} |x_u^j - x_v^j|
\end{aligned}
$$

Thus we have

$$
2|x_u^l - x_v^l| \leq |x_u^l - x_v^l| + \sum_{j \neq l} |x_u^j - x_v^j| = \sum_{j=1}^{k} |x_u^j - x_v^j| = ||x_u - x_v||_1,
$$

which implies $|x_u^l - x_v^l| \leq \frac{1}{2}||x_u - x_v||_1$. ■

Lemma 10.7 *$u \in B(s_i, r)$ if and only if $1 - x_u^i \leq r$.*

Proof:

$$\begin{aligned} u \in B(s_i, r) &\Leftrightarrow \frac{1}{2}||s_i - x_u||_1 \le r \Leftrightarrow \frac{1}{2}\sum_{j=1}^{k} |x_u^j - x_v^j| \le r \\ &\Leftrightarrow \frac{1}{2}\sum_{j \ne i} x_u^j + \frac{1}{2}(1 - x_u^i) \le r \\ &\Leftrightarrow \frac{1}{2}(1 - x_u^i) + \frac{1}{2}(1 - x_u^i) \le r \\ &\Leftrightarrow 1 - x_u^i \le r. \end{aligned}$$

■

Lemma 10.8 *For each edge $e = (u, v)$, $\mathbf{Pr}$[e is in cut] $\le \frac{3}{4}||x_u - x_v||_1$.*

Proof: We say that an index i *settles* edge (u, v) if i is the first index in the random permutation such that at least one of $u, v \in B(s_i, r)$. We say that an index i *cuts* edge (u, v) if exactly one of $u, v \in B(s_i, r)$. Let S_i be the event that i settles (u, v) and X_i be the event that i cuts (u, v). Thus, $\mathbf{Pr}[\text{ } e \text{ is in cut}] = \sum_{i=1}^{k} \mathbf{Pr}[S_i \wedge X_i]$. Note that S_i depends on the random permutation, while X_i is independent of the randomized permutation.

By lemma 10.7, we have

$$\mathbf{Pr}[X_i] = \mathbf{Pr}[min(1 - x_u^i, 1 - x_v^i) \le r < max(1 - x_u^i, 1 - x_v^i)] = |x_u^i - x_v^i|.$$

Let $l = argmin_i(min(1 - x_u^i, 1 - x_v^i))$, that is , s_l is the closest terminal to one of u, v. We can claim that any index $i \ne l$ cannot settle the edge $e = (u, v)$ if l comes before i in permutation π, since if at least one of $u, v \in B(s_i, r)$, then at least one of $u, v \in B(s_l, r)$. Note that the probability that l comes before i in the randomized permutation π is $\frac{1}{2}$. Hence for $i \ne l$, we have

$$\begin{aligned} \mathbf{Pr}[S_i \wedge X_i] &= \mathbf{Pr}[S_i \wedge X_i | l >_\pi i]\mathbf{Pr}[l >_\pi i] + \mathbf{Pr}[S_i \wedge X_i | l <_\pi i]\mathbf{Pr}[l <_\pi i] \\ &= \frac{1}{2}\mathbf{Pr}[S_i \wedge X_i | l >_\pi i] + 0 \\ &\le \frac{1}{2}\mathbf{Pr}[X_i | l >_\pi i] \end{aligned}$$

Since the event X_i is independent of the randomized permutation, $\mathbf{Pr}[X_i | l >_\pi i] = \mathbf{Pr}[X_i]$ and therefore for $i \ne l$,

$$\mathbf{Pr}[S_i \wedge X_i] \le \frac{1}{2}\mathbf{Pr}[X_i] = \frac{1}{2}|x_u^i - x_v^i|.$$

We also have that $\mathbf{Pr}[S_l \wedge X_l] \le \mathbf{Pr}[X_l] \le |x_u^l - x_v^l|$. Therefore, we have

$$\begin{aligned} \mathbf{Pr}[e \text{ is in cut}] &= \sum_{i=1}^{k} \mathbf{Pr}[S_i \wedge X_i] \\ &\le |x_u^l - x_v^l| + \frac{1}{2}\sum_{i \ne l} |x_u^i - x_v^i| \\ &= \frac{1}{2}|x_u^l - x_v^l| + \frac{1}{2}||x_u - x_v||_1 \\ &\le \frac{1}{4}||x_u - x_v||_1 + \frac{1}{2}||x_u - x_v||_1 \qquad \text{By lemma 10.6} \\ &= \frac{3}{4}||x_u - x_v||_1 \end{aligned}$$

■

Now using the above three lemma, we can prove the theorem 10.5.

Proof: Let Z_{uv} be a boolean variable which is 1 if u and v are in different parts of the partition. Then the total cost of the cut returned by this algorithm is $W = \sum_{e=(u,v)\in E} c_e Z_{uv}$, which have the expectation

$$
\begin{aligned}
E[W] &= E\left[\sum_{e=(u,v)\in E} c_e Z_{uv}\right] \\
&= \sum_{e=(u,v)\in E} c_e E[Z_{uv}] \\
&= \sum_{e=(u,v)\in E} c_e \mathbf{Pr}[e \text{ is in cut }] \\
&\leq \sum_{e=(u,v)\in E} c_e \frac{3}{4} ||x_u - x_v||_1 \\
&= \frac{3}{2} * \frac{1}{2} \sum_{e=(u,v)\in E} c_e ||x_u - x_v||_1 \\
&\leq \frac{3}{2} OPT
\end{aligned}
$$

■

10.3 Multicut in General graphs

Definition 10.9 Multi-cut Problem:
Input: *a weighted graph $G(V,E)$, with a cost measure on edges $c_e : E \to \mathbb{R}^+$; a set of terminals pairs $(s_i, t_i), 1 \leq i \leq k$.*
Goal: *find the minimum cost set of edges whose removal separate each $(S_i, T_i), 1 \leq i \leq k$.*

The problem is NP-complete even when G is a star. Vertex conver in general graph can be reduced to multi-cut on stars. We can give a 2-approximation algorithm by primal and dual LP when G is a tree. Today, we present an $O(\log k)$ approximation for multicut in general graphs.

For each vertex pair (s_i, t_i), let P_i denote the set of all paths from s_i to t_i. Let $P = \bigcup_{i=1}^{k} P_i$. Consider the LP-formulation of the minimum multicut problem:

$$
\begin{aligned}
\textit{minimize} \quad & \sum_{e\in E} c_e \cdot x_e \\
\textit{subject to} \quad & \sum_{e\in p_i} x_e \geq 1, \qquad p_i \in P_i, 1 \leq i \leq k \\
& x_e \geq 0, \qquad e \in E
\end{aligned}
$$

The above LP-formulation has an exponential number of constraints. However, we can still solve this LP using the Ellipsoid method. Given a (possible) solution vector $\vec{x}$ (assignments to $x_e, e \in E$), we can check if it is feasible in polynomial time (this implies that the separation oracle for this LP is in P). To do so we interpret variable x_e as a distance label on edge e, for each $e \in E$; then we compute the lengths of shortest paths between each source-sink pair (s_i, t_i) w.r.t the current distance labels. If all the lengths are ≥ 1, then all the paths between each pair (s_i, t_i) must have lengths ≥ 1, and therefore, we can conclude that all constraints are satisfied and the solution is feasible. If the shortest path is < 1 then we obtain a violated constraint.

The following example shows that this LP has a integrality gap of at least 4/3.

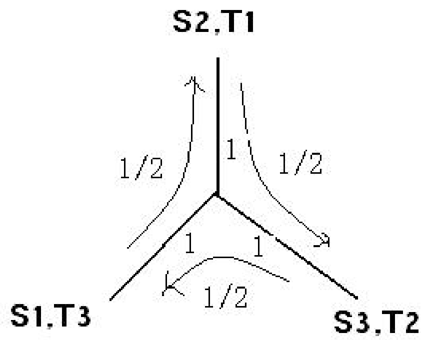

Figure 10.4: An example indicating the gap between the primal and dual

The cost of optimum integer solution is 2 as we have to remove two edges whereas the optimum LP could pick each edge to the extend of 1/2 for a total cost of $\frac{3}{2}$.

Now, we introduce the beautiful $O(\log k)$-factor approximation algorithm due to Garg, Vazirani, and Yannakakis [GVY]. Before giving the algorithm, we restate the problem as a pipe system. This will help to some intuition behind the algorithm.

Consider a feasible solution $\vec{x}$ to the LP. Suppose that we have a pipe running between i, j if there is an edge $e = (i, j)$ in E. Let the length of this pipe be x_e and the cross-sectional area of this pipe be c_e. Therefore, $c_e \cdot x_e$ will be the volume of this pipe and $\sum_{e \in E} c_e \cdot x_e$ will be the total volume of the pipes in our system. With this definition, the multicut problem is in fact the question of designing a pipe system such that the distance between every source-sink pair is at least 1 and the total volume of pipes in the system is minimized. Therefore, the fractional optimal solution, i.e. the solution to the LP, is the volume of the pipe system and we denote it by V^*.

Definition 10.10 *For a feasible solution $\vec{x}$, denote $d_x(u, v)$ to be the length of the shortest path between u and v in G w.r.t the distance labels of $\vec{x}$.*

Definition 10.11 *For a set of vertices $S \subseteq V$, denote the set of edges in the cut $(S, V - S)$ as $\delta(S)$.*

Definition 10.12 *For a vertex $v \in V$ and a (real) radius r, define the set of vertices in G with distance $\leq r$ (with respect to distance label given by $\vec{x}$) to v as $B_x(v, r)$, i.e. $B_x(v, r) = \{u | d_x(u, v) \leq r\}$.*

The algorithm will find disjoint sets of vertices $S_1, \ldots, S_{\ell \leq k}$, called regions by growing balls around terminals such that:

- no region contains any source-sink pair, and for each $1 \leq i \leq k$, either s_i or t_i is in one of the S_j's.
- For each region, the weight of $\delta(S_j)$ is "small".

Lemma 10.13 *The algorithm terminates.*

Region Growing Algorithm for Multi-cut Problem (GVY)

Input: A graph $G(V, E)$ with a cost measure on edges $C_e : E \to \mathbb{R}^+$, and a set of terminal pairs $\{(S_1, T_1), (S_2, T_2), ..., (S_k, T_k)\}$.
Output: A collection of edges C which separate all the pairs.
1. $C \leftarrow \emptyset$; $V \leftarrow V$
2. Find the optimal fractional solution for (**??**)
3. **While there is an** (S_i, T_i) **connected in** V**, do**
4. $S \leftarrow B(S_i, r) \cap V$ for some $r < \frac{1}{2}$
5. $C \leftarrow C \cup \delta(S)$ ($\delta(S)$ is the set of edges cutting S from the rest)
6. $V \leftarrow V - S$
7. **end**
8. **return** C

Figure 10.5: GVY Algorithm

Proof: Since the ball grown around a terminal s_i at each iteration has radius at most $\frac{1}{2}$ it cannot contain t_i. Thus, $\delta(S)$ will separate (at least) one source-sink pair. The algorithm has at most k iterations. ∎

Lemma 10.14 *The algorithm returns a multicut.*

Proof: The only problem is when the algorithm separates some pair (s_i, t_i) and there is a pair (s_j, t_j), such that both $s_j \in B_X(s_i, r)$ and $t_j \in B_X(s_i, r)$. In this case, since we are removing all the vertices of the ball around s_i, then (s_j, t_j) will not be separated by the algorithm. But this scenario is impossible to happen. Otherwise, from $d_x(s_j, t_j) \le d_x(s_i, s_j) + d_x(s_i, t_j) \le 2 \cdot r < 1$, one of the LP constraints for s_j, t_j is violated. ∎

Definition 10.15 *Let V^* be the optimal fractional solution to the LP. Given a vertex $v \in V$ and a radius r, a ball with radius r is defined. Define the volume of this ball (region) as*

$$V_x(v, r) = \sum_{e=(u,v)\in B_x(v,r)} c_e \cdot x_e + \sum_{\substack{e=(u,v)\in\delta(B_x(v,r)) \\ v\in B_x(v,r)}} c_e(r - d_x(u, v)) + \frac{V^*}{k}$$

and the cut volume of this region as

$$C_x(v, r) = \sum_{e\in\delta(B_x(v,r))} c_e$$

Note that $V_x(v, r)$ is an increasing function of r. It is a piece-wise linear function with possible discontinuities at values of r where new vertices are added to the region. Therefore, $V_X(v, r)$ is differentiable everywhere except those possible discontinuous points and

$$\frac{\mathbf{d}\, V_x(v, r)}{\mathbf{d}\, r} = C_x(v, r) \tag{10.7}$$

Lemma 10.16 *There is some $r < \frac{1}{2}$, such that $\frac{C_x(s_i,r)}{V_X(s_i,r)} \le 2 \cdot \ln(k + 1)$ and we can find such an r in polynomial time.*

Proof: By contradiction, assume throughout the region growing process, starting with $r = 0$ ending at $r = \frac{1}{2}$:

$$\frac{C_x(s_i, r)}{V_x(s_i, r)} > 2 \cdot \ln(k + 1).$$

This implies that

$$\frac{\mathbf{d}\, V_x(v,r)}{\mathbf{d}\, r} \cdot \frac{1}{V_x(s_i,r)} > 2 \cdot \ln(k+1).$$

Let $r_1 = 0 \leq r_2 \leq \cdots \leq r_q = \frac{1}{2}$ be the radii at which new vertices are added to the region (s_i, r_q). For all r in (r_j, r_{j+1}):

$$\begin{aligned}
\int_{rj}^{r_{j+1}} \frac{\mathbf{d}\, V_x(v,r)}{\mathbf{d}\, r} \cdot \frac{1}{V_x(s_i,r)} \quad &> \quad \int_{rj}^{r_{j+1}} 2 \cdot \ln(k+1) \mathbf{d}\, r \\
&\Downarrow \\
\ln(V_x(s_i, r_{j+1})) - \ln(V_x(s_i, r_j)) \quad &> \quad (r_{j+1} - r_j) \cdot 2 \cdot \ln(k+1).
\end{aligned}$$

We are going to sum up over all intervals (r_j, r_{j+1}) for $1 \leq j < q$. This will give us a telescopic sum and the terms will be canceled out except the first and the last term. Doing this, even though the function is discontinuous at the end-points of the intervals, is valid because the function $V_x(s_i, r)$ is an increasing function. Thus:

$$\begin{aligned}
\ln(V_X(s_i, r_q)) - \ln(V_x(s_i, r_1)) \quad &> \quad 2 \cdot \ln(k+1) \cdot (r_q - r_1) \\
&\Downarrow \\
\ln(V_x(s_i, \frac{1}{2})) \quad &> \quad 2 \cdot \ln(k+1) \cdot r_q + \ln(\frac{V^*}{k}) \\
&\Downarrow \\
\ln(V_X(s_i, \frac{1}{2})) \quad &= \quad \ln(k+1) + \ln(\frac{V^*}{k}) \\
&\Downarrow \\
\ln(V_x(s_i, \frac{1}{2})) \quad &> \quad \ln(V^* + \frac{V^*}{k}) \\
&\Downarrow \\
V_x(s_i, \frac{1}{2}) \quad &> \quad V^* + \frac{V^*}{k}.
\end{aligned}$$

But this cannot happen, since $V_x(s_i, \frac{1}{2})$ is part of the total volume and cannot be larger than it. This implies that there exists such an $r < \frac{1}{2}$. To find r, consider the vertices of G according to non-decreasing order of distance from s_i: $s_i = v_1, v_2, \ldots, v_p$ with distances $r_1 = 0 \leq r_2 \leq \ldots, r_p \leq r_{p+1}$ where $r_{p+1} \geq \frac{1}{2}$ and $r_p < \frac{1}{2}$. At any interval (r_j, r_{j+1}), the volume $V_x(s_i, r)$ increases while the value of cut $C_x(s_i, r)$ is fixed. Therefore, the volume is maximized (i.e. $\frac{C_x(s_i,r)}{V_x(s_i,r)}$ is minimized) at the end of the interval. So it is enough to check the ratio $\frac{C_x(s_i,r)}{V_x(s_i,r)}$ at the end of the intervals. ∎

Theorem 10.17 *The GVY algorithm is a $(4\ln(k+1))$-factor approximation algorithm for IMC.*

Proof: We charge the cost of the edges removed from the graph at each iteration against the volume of the region removed. By lemma 10.16, at each iteration:

$$\begin{aligned}
C_x(s_i, r) \quad &\leq \quad 2\ln(k+1) \cdot V_x(s_i, r). \\
&\Downarrow \quad (\text{ Summing up for } 1 \leq i \leq k) \\
\sum_{e \in C} c_e \quad &\leq \quad 2\ln(k+1) \sum_{i=1}^{k} V_x(s_i, r) \\
&\leq \quad 2\ln(k+1) \cdot (V^* + \frac{V^*}{k} \cdot k) \\
&= \quad 4\ln(k+1) \cdot V^* \\
&\leq \quad 4\ln(k+1) \cdot OPT.
\end{aligned}$$

■

References

GVY N. Garg, V.V. Vazirani, and M. Yannakakis, Approximate max-flow min-(multi)cut theorems and their applications, *SIAM Journal on Computing*, 1996, 25:235–251.