Approximation Algorithms **Spring 2019**

Lecture 11: Ball-Growing and Multicut

Scribe: Antares Chen 5/10/2019

In this note, we discuss the multicut problem as well as an approximation algorithm provided by Garg, Vazirani, and Yannakakis [3]. We will demonstrate a rounding procedure that uses ball-growing to achieve a $4\ln(k+1)$-approximation ratio.

11.1 Multicut and its Linear Programming Relaxations

Given a graph $G = (V, E)$, edge costs $c_e \geq 0$, and a set of source-sink vertex pairs $\{(s_i, t_i) : i = 1, \ldots, k\}$, the multicut problem asks for $F \subseteq E$ that, when removed, separates each s_i, t_i with minimal cost according to

$$\text{cost}(F) = \sum_{e \in F} c_e$$

The multicut problem is not only NP-hard in general, but is also NP-hard when G is restricted to be a trees. It is also known to be APX-hard due to a reduction to the Unique Games Conjecture [2].

Theorem 11.1. *Assuming the Unique Games Conjecture, there does not exist an α-approximation for multicut for any $\alpha \geq 1$ unless $\mathsf{P} = \mathsf{NP}$.*

A stronger version of the Unique Games Conjecture demonstrates that there does not exist an $O(\log \log n)$ approximation unless $\mathsf{P} = \mathsf{NP}$. It is currently an open problem to design an approximation algorithm that matches the lower bound. For these notes, we will present a $4\ln(k+1)$-approximation algorithm due to Garg, Vazirani, and Yannakakis. The algorithm first performs a linear programming relaxation, then rounds the solution using a procedure known as *ball-growing* which is applicable as the LP relaxation benefits from certain *metric* properties. Let us first construct a easily interpretable LP relaxation whose metric properties are not immediately evident.

11.1.1 A Path-Based LP Relaxation

We begin with the multicut integer program. First, define decision variables for each edge e.

$$x_e = \begin{cases} 1 & \text{if } e \text{ is removed} \\ 0 & \text{otherwise} \end{cases}$$

Our objective is then to minimize the cost of F. The objective function is

$$\text{cost}(F) = \sum_{e \in E} c_e x_e$$

Finally, we must constrain our integer program to return an assignment that cuts each s_i and t_i pair. We know that s_i, t_i are cut by F if an edge on each path between the two vertices is removed. Let $\mathcal{P}_i$ be the set of all s_i to t_i paths. It suffices for us to ensure that at least one edge admits $x_e = 1$ for every $P \in \mathcal{P}_i$. This is equivalent to adding the constraints

$$\sum_{e \in P} x_e \geq 1 \qquad \forall P \in \mathcal{P}_i \quad \forall i = 1, \ldots, k$$

This gives our integer program. The linear programming relaxation of this replaces the integrality constraint with $x_e \geq 0$. The LP relaxation for multicut is then the following.

$$\begin{aligned} \text{minimize} \quad & \sum_{e \in E} c_e x_e \\ \text{subject to} \quad & \sum_{e \in P} x_e \geq 1 \quad \forall P \in \mathcal{P}_i \quad \forall i = 1, \ldots, k \\ & x_e \geq 0 \qquad \forall e \in E \end{aligned} \tag{11.1}$$

An algorithm that rounds this LP will first need to solve for x_e. But there is an immediate issue: there could be exponentially many path constraint with respect to the size of the graph! There are two ways to get around this. First is to derive an alternative LP relaxation that has polynomially many constraints, while second is to employ the *Ellipsoid Method*, a procedure that can solve exponentially sized linear programs provided that one can demonstrate the existence of a *separation oracle*. We'll instead demonstrate a polynomially sized, relaxation for multicut that more clearly highlights the metric structure that exists in multicut.

11.1.2 A Metric LP Relaxation

Consider a valid multicut F and define the indicator function $d_F : V \times V \to \{0, 1\}$ on pairs of vertices that are separated when F is removed.

$$d_F(u, v) = \begin{cases} 1 & \text{if } u \text{ and } v \text{ are separated by } F \\ 0 & \text{otherwise} \end{cases}$$

This function defines a *metric*[1] over pairs of vertices in G. It holds that $d_F(u, u) = 0$ for any vertex u, is non-negative, symmetric, and satisfies the triangle inequality. To see that d_F satisfies the triangle inequality, consider any three distinct vertices u, v, w. Since distances are $\{0, 1\}$, we need only verify that the triangle with two sides 0 never exists. It cannot exist because any cut which separates two of u, v, w must remove two sides of the triangle.

The function d_F is often called the *cut metric* induced by the partition. It is a 0-1 metric defined over all pairs of vertices. The multicut problem thus asks for the minimum cost metric among all cut metrics separating each s_i, t_i pair. A valid relaxation of this problem would then be to ask for the minimum cost metric among *all metrics* separating each s_i, t_i pair. To formulate this relaxation as a linear program, we begin by noting that the metric constraints can be encoded via linear expressions. If x_{uv} denotes the distance between two

[1] Actually, d_F is called a semi-metric as $d_F(u, v) = 0$ does not necessarily imply $u = v$ as usually required by metrics.

vertices u, v, we can write the constraints as

$$\begin{array}{lll} x_{uv} + x_{vw} \geq x_{uw} & \forall u \neq v \neq w & \text{triangle inequality} \\ x_{uv} = x_{vu} & \forall u, v \in V & \text{symmetry} \\ x_{uv} \geq 0 & \forall u, v \in V & \text{non-negativity} \end{array}$$

Next, to enforce that the metric separates each s_i, t_i pair, we need only to ensure that the distance between the source and sink is at least 1. For each i, we add the constraint $x_{s_i,t_i} \geq 1$. We can thus write the following *metric* LP relaxation for multicut.

$$\begin{array}{lll} \text{minimize} & \displaystyle\sum_{u,v \in V} c_{uv} x_{uv} & \\ \text{subject to} & x_{uv} + x_{vw} \geq x_{uw} & \forall u \neq v \neq w \\ & x_{s_i,t_i} \geq 1 & \forall i = 1, \ldots, k \\ & x_{uv} \geq 0 & \forall u, v \in V \end{array} \tag{11.2}$$

Note we need not write the symmetry constraint as the edge costs are symmetric. It is worth checking that this is a valid relaxation of multicut.

Claim 11.2. *Let x be an integral solution to LP 11.2. Then removing any e where $x_e = 1$ removes a valid multicut.*

Proof. Suppose not; then there is a pair s_i, t_i with a path P such that each edge $e \in P$ admits $x_e = 0$. Intuitively, this violates the triangle inequality as the direct distance x_{s_i,t_i} should be the shortest distance between s_i, t_i of which the constraint $x_{s_i,t_i} \geq 1$ dictates must be at least 1.

More precisely, imagine a new graph G' constructed from G with its edges weighted by x_e. We claim that x_{s_i,t_i} denotes the shortest path distance between s_i, t_i in G'. Consider any other s_i, t_i path P and write it as

$$s_i \quad u_1 \quad \ldots \quad u_\ell \quad t_i$$

Now, the triangle inequality allows us to write a sequence of inequalities like so

$$x_{s_i,u_1} + x_{u_1,t_i} \geq x_{s_i,t_i} \qquad x_{u_1,u_2} + x_{u_2,t_i} \geq x_{u_1,t_i} \qquad \ldots \qquad x_{u_{\ell-1},u_\ell} + x_{u_\ell,t_i} \geq x_{u_{\ell-1},t_i}$$

Together, these inequalities imply the following relation

$$x_{s_i,u_1} + x_{u_1,u_2} + \ldots + x_{u_{\ell-1},u_\ell} + x_{u_\ell,t_i} \geq x_{s_i,t_i}$$

The LHS is the length of P while the RHS is the weight of the edge (s_i, t_i) in G'. Thus the length of any s_i, t_i path in G' must be at least x_{s_i,t_i}. Now, if there is a path P such that $x_e = 0$ for $e \in P$ then its length in G' is 0 contradicting the fact that the shortest s_i, t_i path must have length at least 1. □

One thing to note about this relaxation is that the metric is defined over all pairs of vertices, but the given graph may not have an edge between each vertex pair. It suffices to let $c_{uv} = 0$ whenever $(u, v) \notin E$ as cutting an edge that does not exist in the graph does not contribute to the cost of the cut.

11.1.3 Metric Completions

Given a solution to LP 11.2, we can construct an alterate LP solution by performing a *metric completion*. For every pair of vertices $u, v \in V$, the metric completion of x assigns $\hat{x}_{uv}$ to be the length of the shortest path between u and v in the graph G whose edges e are weighted as x_e. That is if we denote $\mathcal{P}_{uv}$ as the set of all paths from u to v, $\hat{x}_{uv}$ is given by

$$\hat{x}_{uv} = \min_{P \in \mathcal{P}_{uv}} \sum_{e \in P} x_e$$

Notice that $\hat{x}_{uv}$ is both a feasible and optimal solution for LP 11.2.

Claim 11.3. *Given a solution x to LP 11.2, its metric completion $\hat{x}$ is feasible and optimal.*

Proof. Notice that $\hat{x}_{uv}$ automatically forms a metric between all pairs of vertices as shortest path distances satisfy the triangle inequality. Thus to demonstrate feasibility, we need only verify that $\hat{x}_{s_i,t_i} \geq 1$ for each i. However, the argument for claim 11.2 demonstrates that $\hat{x}_{s_i,t_i} = x_{s_i,t_i} \geq 1$ since x_{s_i,t_i} is the shortest path length between s_i, t_i in a graph weighted by x_e's.

To demonstrate optimality, notice that the objective value of $\hat{x}$ is equivalent to that of x. For any $e \in E$, we have that $\hat{x}_e \leq x_e$. This is because $\hat{x}_e$ is the length of the shortest path between the endpoints of e. Taking just the edge e is one such path, thus the length of the shortest path can only be shorter than x_e. Consequently, the objective value of $\hat{x}$ is at most

$$\sum_{e \in E} c_e \hat{x}_e \leq \sum_{e \in E} c_e x_e$$

Because x is an optimal solution to the LP, $\hat{x}$ must be an optimal solution as well. □

Optimality in claim 11.3 actually follows for a simpler reason – $\hat{x}_e = x_e$ for any $e \in E$ thus the two objective values are in fact equal. The reason why the above argument is provided is because it also applies to demonstrating that the metric completion of a solution to the path-based relaxation 11.1 is optimal and feasible for LP 11.1.

In fact, performing a metric completion tells us how to relate the above two LP relaxations together since the completion of a solution to either LP makes it feasible for the other. The metric completion of a solution to the metric relaxation 11.2 is feasible for the path-based LP 11.1 because it will satisfy the following constraints:

$$\sum_{e \in P} \hat{x}_e \geq 1 \qquad \forall P \in \mathcal{P}_i \; \forall i = 1, \ldots, k$$

The metric relaxation requires $x_{s_i,t_i} \geq 1$. The value of x_{s_i,t_i} is also the shortest path length between s_i, t_i. Thus any other path must have length at least 1. In the other direction, we can show that the metric completion of x the solution to the path-based LP relaxation 11.1 is also a feasible and optimal solution for that LP. It is then feasible for LP 11.2 as shortest path distances automatically satisfy the triangle inequality, and $\hat{x}_{s_i,t_i} \geq 1$ for all i because $\sum_{e \in P} x_e \geq 1$ for any path P between s_i, t_i.

11.2 Rounding the Linear Program

We will now develop an algorithm for rounding a solution x to the metric LP relaxation 11.2. Let's start by addressing how to construct a *valid* multicut, then determine how to make it *low-cost*. For the remainder of these notes, we'll denote $d_x(u, v)$ as the metric completion of x on edge (u, v).

11.2.1 Balls and Pipe Systems

We want to leverage the metric structure from our LP relaxation to construct our multicut. One way to do this is to partition the graph into clusters separating each s_i, t_i pair, then remove the multicut consisting of edges that cross the boundary of each cluster. We can use the fact that d_x is a metric to construct our clusters by choosing balls of a certain radius as measured by d_x. Let's define the ball of radius r centered at vertex $u' \in V$ as $\mathcal{B}_x(u', r) \subseteq V$, the set of all vertices within distance r of u' as measured by the distance d_x.

$$\mathcal{B}_x(u', r) = \{v \in V : d_x(u', v) \leq r\}$$

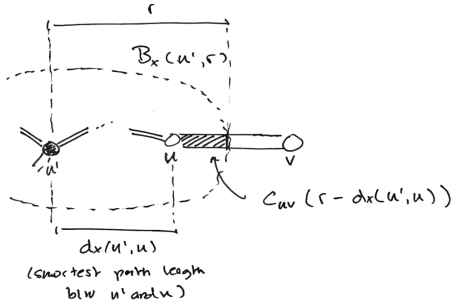
Later on, it will be useful for us to consider the *volume* of each ball. Given $S \subseteq V$, let $E(S)$ denote the set of edges whose endpoints are contained in S. Additionally, let $\partial(S)$ denote the set of edges on the *boundary* of S, i.e. (u, v) with $u \in S$ and $v \notin S$. The volume of a ball $\mathcal{B}_x(u', r)$ is defined by the following:

$$\text{Vol } \mathcal{B}_x(u', r) = \sum_{e \in E(\mathcal{B}_x(u', r))} c_e x_e + \sum_{(u,v) \in \partial(\mathcal{B}_x(u', r)) : u \in \mathcal{B}_x(u', r)} c_{uv}\big(r - d_x(u', u)\big)$$

Williamson and Shmoy's *The Design of Approximation Algorithms* uses the analogy of a *pipe system* to interpret these definitions. We think of the graph as a network of pipes: each edge e is replaced with a pipe of *length* x_e and *cross sectional area* c_e. The term $c_e x_e$ then denotes the volume of the pipe replacing e. The quantity Vol $\mathcal{B}_x(u, r)$ could then be interpreted as the total volume of all pipes within a radius r of u.

$$\text{Vol } \mathcal{B}_x(u', r) = \underbrace{\sum_{e \in E(\mathcal{B}_x(u', r))} c_e x_e}_{\text{Total volume of pipe contained in the ball}} + \underbrace{\sum_{(u,v) \in \partial(\mathcal{B}_x(u', r)) : u \in \mathcal{B}_x(u', r)} c_{uv}\big(r - d_x(u', u)\big)}_{\text{Volume of pipe contained within the boundary}}$$

To visualize the second sum, consider the following diagram

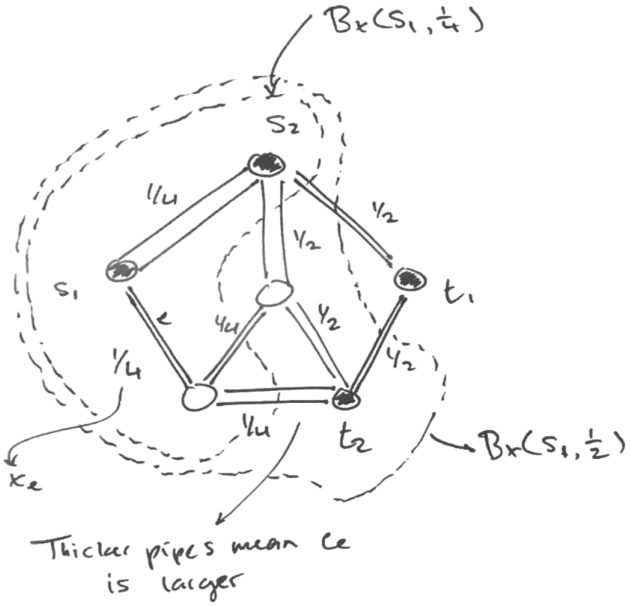


11.2.2 Rounding via Balls

Our rounding algorithm will cluster the graph according to appropriately chosen balls. All that remains is to choose the center and radius for each ball. These values will need to be chosen such that each s_i, t_i pair is separated. A certain method of ensuring separation is to center a ball at each s_i, then set each radius to $r = 0$. $\mathcal{B}_x(s_i, 0)$ is always the singleton containing s_i thus s_i and t_i are always separated. On the other hand, the ball centered at s_i and with too large a radius may contain both s_i and t_i as we may recall that

$$d_x(s_i, t_i) = x_{s_i,t_i} \geq 1$$

Actually, we can always choose $r < 1$ and ensure that s_i and t_i are not in the same cluster because of the above. However, this does not preclude a ball centered at s_i containing some pair s_j and t_j for $i \neq j$. Consider the following graph whose edges are labeled with the solution to its metric relaxation LP.



Notice that $\mathcal{B}_x(s_1, \frac{1}{2})$ contains both s_2 and t_2. What is the largest value of r such that a cluster centered at s_i will not contain any s_j, t_j pair? Because $d_x(s_i, t_i) \geq 1$ for any pair s_i, t_i, choosing $r < \frac{1}{2}$ suffices because of the triangle inequality. If there is a ball $\mathcal{B}_x(s_i, r)$ where $r < \frac{1}{2}$ containing any s_j, t_j, then

$$x_{s_j,t_j} = d_x(s_j, t_j) \leq d_x(s_j, s_i) + d_x(s_i, t_j) < \frac{1}{2} + \frac{1}{2} = 1$$

contradicting the fact that $x_{s_j,t_j} \geq 1$. Consequently, any $r < \frac{1}{2}$ will lead the following rounding procedure to return a valid multicut. Notice that in the following procedure, we remove the ball of radius r at each iteration to ensure that no edge is within two different balls.

LP Rounding Algorithm 1
Given $G = (V, E)$, source-sink pairs $\{(s_i, t_i) : i = 1, \ldots, k\}$ and radius $r \in [0, \frac{1}{2})$, do the following:

1. Solve the linear program relaxation 11.2 for x and set $F = \varnothing$.
2. For $i = 1, \ldots, k$ do:
   - If s_i, t_i are not separated in $(V, E - F)$, then choose $\mathcal{B}_x(s_i, r)$
   - Update $F = F \cup \partial(\mathcal{B}_x(s_i, r))$
   - Remove vertices in $\mathcal{B}_x(s_i, r)$ and edges in $E(\mathcal{B}_x(s_i, r)) \cup \partial(\mathcal{B}_x(s_i, r))$.
3. Return F

Now that we have a way of procuring a valid multicut, we can turn our attention towards making it low-cost. Our goal will now be to show that an auspicious choice of r will allow us to bound the cost of the rounded multicut.

11.3 Low-cost Cuts via Ball-Growing

It will be useful for us to develop some more machinery to discuss the volume of a ball as well as the cost of the partitions added to F. Let us define the following.

(1) Define the total volume of the given graph to be V^* given by

$$V^* = \sum_{e \in E} c_e x_e$$

Since x is the LP optimal solution, V^* lower bounds the cost of the optimal multicut OPT.

(2) Previously, we defined the volume of a ball Vol $\mathcal{B}_x(u', r)$. Let us now denote another quantity $V_x(s_i, r)$ as the volume of $\mathcal{B}_x(s_i, r)$ with an added $\frac{V^*}{k}$ term.

$$V_x(s_i, r) = \frac{V^*}{k} + \text{Vol } \mathcal{B}_x(s_i, r)$$

The $\frac{V^*}{k}$ term seems a bit mysterious, but adding this term ensures two properties. First, $V_x(s_i, 0) > 0$ for all $i = 1, \ldots, k$ since $\frac{V^*}{k} > 0$. Second, we have

$$\sum_{i=1}^{k} V_x(s_i, 0) = \sum_{i=1} \frac{V^*}{k} = V^*$$

These two properties will be quite handy later.

(3) Define $c_x(s_i, r)$ to be the cost of the cut induced by removing $\mathcal{B}_x(s_i, r)$ from the graph. That is

$$c_x(s_i, r) = \sum_{e \in \partial(\mathcal{B}_x(s_i, r))} c_e$$

11.3.1 Ball-Growing

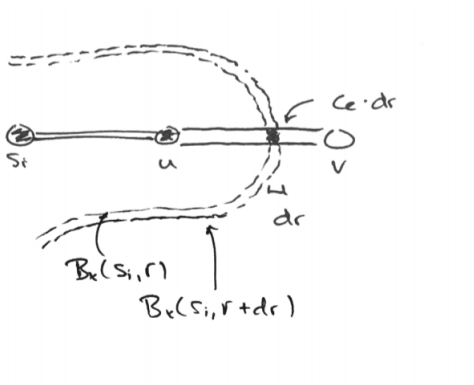
Consider a the ball $\mathcal{B}(s_i, r)$ removed during an iteration of algorithm 1. It is uncertain how we can handle the cost of cut $\partial(\mathcal{B}(s_i, r))$ directly, but suppose we could *charge* the cost of the cut to the volume of the ball. That is to say we could discover α such that

$$c_x(s_i, r) \leq \alpha \cdot V_x(s_i, r) \tag{11.3}$$

This would be very useful as we would then have a potential way to relate the cost of the cut to V^* a lowerbound on OPT. The reason why we can expect such an α to exist, and also why pipe networks provide such a nice analogy for this problem, is because the change in volume of a pipe captured by an infinitesimal change in the radius of the ball is proportional to the pipe's cross-sectional area. That is to say:

$$V_x'(s_i, r) = \frac{d}{dr} V_x(s_i, r) = c_x(s_i, r)$$

Imagine a ball around s_i like so. If we grow r by an infinitesimally small amount then we add the sum of cross-sectional area of all pipes crossing the boundary $\mathcal{B}(s_i, r)$ to the volume of $\mathcal{B}(s_i, r)$. The sum of cross-sectional areas of pipes crossing the boundary of s_i's ball is exactly the quantity measured by $c_x(s_i, r)$!



Inequality 11.3 would then reduce to the following

$$c_x(s_i, r) \leq \alpha \cdot V_x(s_i, r) \qquad \Longrightarrow \qquad \frac{c_x(s_i, r)}{V_x(s_i, r)} \leq \alpha \qquad \Longrightarrow \qquad \frac{V_x'(s_i, r)}{V_x(s_i, r)} \leq \alpha$$

But $\frac{V_x'(s_i,r)}{V_x(s_i,r)}$ is the derivative of $\ln(V_x'(s_i, r))$. If we let $F'(r) = \frac{V_x'(s_i,r)}{V_x(s_i,r)}$, we can reduced our task of determining α to bounding the value of derivative of $F(r)$. For that we use the Mean Value Theorem. If $F(r)$ is continuous on $[a, b]$ and differentiable on (a, b), then there exists $c \in (a, b)$ such that

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

However, F is a monotonic non-increasing function. Thus we have for any $r \in [0, \frac{1}{2})$

$$F'(r) \leq \frac{F(\frac{1}{2}) - F(0)}{\frac{1}{2} - 0}$$

Let's first bound $F(\frac{1}{2})$. The volume of any ball of radius r will be at most the total volume of the graph, hence we have the following.

$$F\left(\frac{1}{2}\right) = \ln\left(V_x\left(s_i, \frac{1}{2}\right)\right) \leq \ln\left(V^* + \frac{V^*}{k}\right)$$

Then we bound $F(0)$. This is where it's critical that $V_x(s_i, 0) > 0$, otherwise the logarithm is undefined!

$$F(0) = \ln(V_x(s_i, 0)) = \ln\left(\frac{V^*}{k}\right)$$

We can now bound $F'(r)$. We have

$$\frac{c_x(s_i, r)}{V_x(s_i, r)} = F'(r) \leq \frac{F(\frac{1}{2}) - F(0)}{\frac{1}{2} - 0} \leq 2\left(\ln\left(V^* + \frac{V^*}{k}\right) - \ln\left(\frac{V^*}{k}\right)\right) = 2\ln(k+1)$$

What we have shown is that, for an auspicious choice of r, we can bound the cost our cut with the volume of the cut. The technique of finding such a cut where we can charge the cost of its boundary to the volume is known as *ball-growing*. To summarize, we have demonstrated

Theorem 11.4. *Given a feasible solution x to LP 11.1, for any s_i there exists an $r \in [0, \frac{1}{2})$ that can be found in polynomial time such that*

$$\text{cost}(\partial(\mathcal{B}_x(s_i, r))) \leq 2\ln(k+1) \cdot V_x(s_i, r)$$

Except we haven't really demonstrated this. The application of the Mean Value Theorem requires F to be differentiable. However, $F(r)$ may not even be continuous at certain points! We will fix this issue with a more careful application of the Mean Value Theorem and also demonstrate how r can be found in polynomial time later on in the notes. For now, let's suppose we have theorem 11.4 and complete our analysis of the approximation ratio.

11.3.2 The Approximation Ratio

Our heuristic argument in the preceding section demonstrates the existence of an r such that the cost of a ball is at most $2\ln(k+1)$ times its volume. Using this, we'll show that algorithm 1 returns a $4\ln(k+1)$-approximation.

Theorem 11.5. *Algorithm 1 returns a $4\ln(k+1)$-approximation for multicut*

Proof. At each iteration of algorithm 1, we add $F_i = \partial(\mathcal{B}_x(s_i, r))$ to F. In iterations i where no ball is removed, we'll let $F_i = \varnothing$ and Vol $\mathcal{B}_x(s_i, r) = 0$. Choosing r according to theorem 11.4 gives the cost of F_i.

$$\text{cost}(F_i) \leq 2\ln(k+1) \cdot V_x(s_i, r) = 2\ln(k+1) \cdot \left(\text{Vol } \mathcal{B}_x(s_i, r) + \frac{V^*}{k}\right)$$

Since $\mathcal{B}_x(s_i, r)$ is removed from G at every iteration along with all adjacent edges, $F_i \cap F_j = \varnothing$ for $i \neq j$. Additionally, the volume measured by Vol $\mathcal{B}_x(s_i, r)$ will not overlap with that measured by Vol $\mathcal{B}_x(s_j, r)$ for

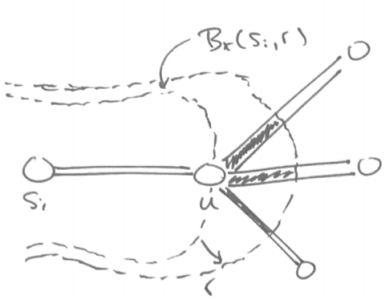
$i \neq j$. The total cost of F returned by the algorithm is the following.

$$\begin{aligned}\text{cost}(F) &= \sum_{i=1}^{k} \text{cost}(F_i) \\ &\leq 2\ln(k+1) \cdot \sum_{i=1}^{k} \left(\text{Vol}\, \mathcal{B}_x(s_i, r) + \frac{V^*}{k} \right) \\ &= 2\ln(k+1) \cdot \left(V^* + V^* \right) \\ &= 4\ln(k+1) \cdot V^*\end{aligned}$$

and because V^* is the cost of the optimal LP solution x, which lower bounds the optimal cost OPT of any multicut, we have $\text{cost}(F) \leq 4\ln(k+1) \cdot \text{OPT}$ as required. □

11.3.3 Fixing Continuity Issues

Let's revisit the continuity of $F(r) = \ln(V_x(s_i, r))$ and ask where $F(r)$ could be discontinuous. Around where $r = d_x(s_i, u)$ for some vertex $u \neq s_i$, there could be a discontinuous jump in $V_x(s_i, r)$ because u could be connected to more than one pipe not currently in $\mathcal{B}_x(s_i, r)$.



However, on intervals of $r \in [0, \frac{1}{2})$ where increasing r does not introduce a new vertex into $\mathcal{B}_x(s_i, r)$, the value of $V_x(s_i, r)$ grows smoothly with respect to r. This suggests that we should partition the interval $[0, \frac{1}{2})$ into intervals where *no new vertex* is introduced into $\mathcal{B}_x(s_i, r)$. We now prove theorem 11.4.

Proof of theorem 11.4. We will show that there exists $r \in [0, \frac{1}{2})$ such that

$$\frac{c_x(s_i, r)}{V_x(s_i, r)} \leq 2\ln(k+1)$$

Let us order the vertices $v \neq s_i$ as $v_1, \ldots, v_\ell$ where

$$d_x(s_i, v_{j_1}) \leq d_x(s_i, v_{j_2})$$

when $j_1 \leq j_2$, define $r_j = d_x(s_i, v_j)$, and denote r_j^- as value that's infinitesimally smaller than r_j. We will demonstrate the existence of r by choosing it uniformly at random on the interval $[0, \frac{1}{2})$. If we can bound

the *expectation* of $\frac{c_x(s_i,r)}{V_x(s_i,r)}$ by what we want, then there must exist an actual choice of r where the bound holds deterministically. Our choice of partitioning $[0, \frac{1}{2})$ using r_j's is critical as $F(r)$ is continuous over interval $[r_j, r_{j+1}^-]$ and differentiable over (r_j, r_{j+1}^-). For notational simplicity, let $r_{\ell+1} = \frac{1}{2}$. Let us compute the expectation of $\frac{c_x(s_i,r)}{V_x(s_i,r)}$ when r is distributed uniformly at random on $[0, \frac{1}{2})$.

$$\begin{aligned}
\mathbb{E}\left[\frac{c_x(s_i,r)}{V_x(s_i,r)}\right] &= \frac{1}{\frac{1}{2}-0} \cdot \int_0^{\frac{1}{2}} \frac{c_x(s_i,r)}{V_x(s_i,r)}\, dr \\
&= \frac{1}{\frac{1}{2}-0} \cdot \sum_{j=1}^{\ell} \int_{r_j}^{r_{j+1}^-} \frac{c_x(s_i,r)}{V_x(s_i,r)}\, dr \\
&= 2 \cdot \sum_{j=1}^{\ell} \Big(\ln(V_x(s_i, r_{j+1}^-)) - \ln(V_x(s_i, r_j)) \Big) \\
&\leq 2 \cdot \sum_{j=1}^{\ell} \Big(\ln(V_x(s_i, r_{j+1})) - \ln(V_x(s_i, r_j)) \Big)
\end{aligned}$$

The last line follows as $F(r)$ is monotonically non-decreasing. This forms a telescoping sum which reduces to

$$2 \cdot \sum_{j=1}^{\ell} \Big(\ln(V_x(s_i, r_{j+1})) - \ln(V_x(s_i, r_j)) \Big) = 2 \cdot \Big(\ln(V_x(s_i, \tfrac{1}{2})) - \ln(V_x(s_i, 0)) \Big) \leq 2\ln(k+1)$$

Consequently, the expectation is bounded by $2\ln(k+1)$ hence there must be an r achieving $\frac{c_x(s_i,r)}{V_x(s_i,r)} \leq 2\ln(k+1)$. To find r in polynomial time, observe that on the interval $[r_j, r_{j+1}]$ the cost of the boundary $c_x(s_i, r)$ remains constant while $V_x(s_i, r_{j+1})$ grows monotonically. This means the ratio $\frac{c_x(s_i,r)}{V_x(s_i,r)}$ is minimized at r_{j+1}. Finding the minimizer requires only checking each $r_1, \ldots, r_\ell$, and as $\ell \leq n$, a linear number of computations suffice. □

11.4 Final Remarks

We highlighted two key ideas in this note. First, we construct a metric LP relaxation to capture metric structure that is ultimately useful for rounding the solution. This strategy can be traced to a paper by Leighton and Rao [5][6] where they use this to construct low-cost graph partitions for various cut problems. Another result by Arora, Rao, Vazirani [1] uses this idea (and many more sophisticated ideas) to round a semidefinite programming relaxation for uniform sparsest cut and produce an $O(\sqrt{\log n})$-approximation.

We also described ball-growing, a method of constructing cuts that charge the cost of the boundary to its volume. The probabilistic proof of theorem 11.4 is more aligned with how ball-growing is described in literature. One often sees a procedure similar to the following describing the process.

1. Pick an arbitrary vertex of the graph u' and start growing a ball $\mathcal{B}(u', r)$ around u'.
2. Increment $r = r + \Delta r$, until a certain condition (such as the size of the boundary being bounded by the volume with some factor α) is met.
3. Remove $\mathcal{B}(u', r)$ and all edges adjacent to it. Repeat until no vertices remain.

The probabilistic argument then demonstrates the existence of an r that meets the condition and, with an appropriate choice of Δr, the algorithm will terminate in time polynomial with respect to the size of the graph. The condition upon which a ball is removed can be fluid. For example, the argument can be tweaked to construct *low-diameter* decompositions of a graph – one where each cluster has boundary size bounded by volume and low shortest path length diameter. This has many uses. Kelner and Mądry [4] use a low-diameter decomposition to construct a fast algorithm for sampling random spanning trees (a primitive useful for problems like maxflow and solving certain linear systems), while Trevisan [7] uses this to construct better approximation algorithm for solving Unique Games.

References

[1] Arora, S., Rao, S., & Vazirani, U. (2009). "Expander flows, geometric embeddings and graph partitioning." In *Journal of the ACM (JACM)*, 56(2), 5.

[2] Chawla, S., Krauthgamer, R., Kumar, R., Rabani, Y., & Sivakumar, D. (2006). "On the hardness of approximating multicut and sparsest-cut." In *computational complexity*, 15(2), 94-114.

[3] Garg, N., Vazirani, V. V., & Yannakakis, M. (1996). "Approximate max-flow min-(multi) cut theorems and their applications." In *SIAM Journal on Computing*, 25(2), 235-251.

[4] Kelner, J. A., & Madry, A. (2009, October). "Faster generation of random spanning trees". In *2009 50th Annual IEEE Symposium on Foundations of Computer Science* (pp. 13-21). IEEE.

[5] Leighton, T., & Rao, S. (1988). "An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms." In *[Proceedings 1988] 29th Annual Symposium on Foundations of Computer Science*. (pp. 422-431). IEEE.

[6] Leighton, T., & Rao, S. (1999). "Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms." In *Journal of the ACM (JACM)*, 46(6), 787-832.

[7] Trevisan, L. (2005). "Approximation algorithms for unique games." In *46th Annual IEEE Symposium on Foundations of Computer Science* (pp. 197-205). IEEE.