Approximation Algorithms **Spring 2019**

Lecture 11: Region Growing and Multicut

Scribe: Antares Chen 5/10/2019

In these notes, we discuss an approximation algorithm for the multicut problem due to Garg, Vazirani, and Yannakakis [2]. Our analysis will highlight the use of *region growing*, a technique that charges the cost of partitioning a graph to the volume of its partitions. This technique has been applied to a wide range of problems in graph theory and approximation algorithms including constructing high girth graphs, sampling random spanning trees, and approximating uniform sparsest cut.

11.1 Multicut

In the MULTICUT problem, we are given an undirected graph $G = (V, E)$, edge costs $c_e \geq 0$, and source-sink vertex pairs $\{(s_i, t_i) : i = 1, \ldots, k\}$. Our goal is to compute a cut $F \subseteq E$ with minimum cost such that removing F from G will disconnect s_i, t_i for all $i = 1, \ldots, k$. More precisely, there must not exist any path connecting s_i to t_i in the graph $(V, E - F)$. We measure the cost of the cut via

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

The MULTICUT problem is fairly difficult; it is NP-hard to solve even in the case G is a trees. We will discuss the history of this problem near the end of these notes. For now, we focus our attention towards designing an approximation algorithm via LP rounding.

11.1.1 An LP Relaxation

Let's begin by constructing an integer linear program for MULTICUT. Since we wish to compute $F \subseteq E$, we place decision variables on our edges. For each $e \in E$, let $x_e = 1$ if e is to be added to F otherwise $x_e = 0$. With these x_e's, the cost of our cut F is given by $\sum_{e \in E} c_e x_e$. What about our constraints? Beyond non-negativity, we need only enforce that s_i and t_i are disconnected once F is removed.

Denote $\mathcal{P}_i$ to be the set of all s_i-t_i paths. Since we are removing e if $x_e = 1$, it suffices for us to ensure that at least one edge admits $x_e = 1$ for every $P \in \mathcal{P}_i$ for all $i = 1, \ldots, k$. Our constraint will thus be the following:

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

Finally, we relax our integer linear program by replacing the integrality constraint $x_e \in \{0, 1\}$ with $x_e \geq 0$.

Our LP relaxation for MULTICUT is the following.

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

11.1.2 The Ellipsoid Method and A Separation Oracle for Multicut

Our LP rounding algorithm will first solve the linear program relaxation for a optimal continuous solution x. However, one look at relaxation 11.1 raises an immediate issue. How can we efficiently solve the LP when there may be exponentially many paths between s_i and t_i in the given graph! There does exist a polynomial sized (with respect to the number of vertices and edges in the graph) LP relaxation for MULTICUT. For these notes, we will approach this problem using the *Ellipsoid Method*.

The Ellipsoid Method is a optimization method that will solve a linear program provided one can construct a polynomial-time *separation oracle*. A separation oracle is a *problem-specific* algorithm that operates as follows. Given a potential solution to the LP x, it returns YES if x is feasible. If x is not feasible, then it returns NO and provides a constraint that x violates.

Theorem 11.1. *Given a linear program and a valid separation oracle, there exists an algorithm that does the following:*

i. Solves for the optimal solution of the linear program.

ii. Runs in polynomial time with respect to the number of variables in the LP as well as the bit-wise complexity of encoding any one constraint.

Using the Ellipsoid Method to solve LP 11.1 requires us to exhibit a polynomial-time separation oracle that distinguishes the feasibility of x.

Separation Oracle
Given a potential solution x to linear program 11.1, do the following:

1. Check if all $x_e \geq 0$. Return NO and x_e if any $x_e < 0$.
2. Construct the graph $G' = (V, E)$ with edges weighted by x_e.
3. For all $i = 1, \ldots, k$, compute the shortest path p_i between s_i and t_i on G'.
4. If the length of any p_i is less than 1, then return NO and the violated constraint:

$$\sum_{e \in p_i} x_e \geq 1$$

5. Return YES otherwise.

Notice that this certainly runs in polynomial time with respect to the size of the graph. We need only perform

k shortest path computations and a linear pass through all x_e's. All that remains is to show that this is a valid separation oracle.

Claim 11.2. *The separation oracle returns* YES *if and only if the given x is feasible.*

Proof. Suppose x is feasible, then $x_e \geq 0$ for all $e \in E$ implying step (1) will not fail. Now consider any s_i, t_i pair. Since x is feasible, the following constraint holds for all $P \in \mathcal{P}_i$.

$$\sum_{e \in P} x_e \geq 1$$

However, this sum is exactly the length of P in graph G'. Since the length of any path $P \in \mathcal{P}_i$ is at least 1, the shortest path p_i will also have distance at least 1 implying step (4) will not fail. The oracle will correctly return YES.

Now suppose x is not feasible. Then either $x_e < 0$, for which step (1) will fail, or $\sum_{e \in P} x_e < 1$ for some $P \in \mathcal{P}_i$. If the length of P is less than 1, then the length of the shortest path between s_i and t_i will less than 1 as its length is at most that of P. It follows that step (4) will fail and the oracle will correctly return NO. □

11.1.3 Pipes and Balls

Now that we can solve the linear program in polynomial time, we begin our work towards a rounding algorithm for MULTICUT. A nice way to visualize a solution to the LP relaxation is as a system of *pipes*. Given the optimal solution x to LP 11.1, imagine each edge $e \in E$ as a pipe with *length* x_e and *cross-sectional area* c_e. The quantity $c_e x_e$ is thus the volume of the pipe represented by edge e.

What does the linear program compute under this interpretation? It determines pipe lengths x_e that minimizes the total volume of the pipe network $\sum_{e \in E} c_e x_e$ subject to the constraint that (1) pipe lengths are non-negative and (2) there is at least 1 unit length of pipe along all paths connecting s_i to t_i. Our dependence on pipe lengths, and certainly how we constructed the separation oracle, seem to point us towards using the optimal solution x of our LP to formalize a notion of *distance* between the vertices of G.

We define the following distance function $d_x : V \times V \to \mathbb{R}_{\geq 0}$ where $d_x(u, v)$ is the shortest path distance between u and v on G with each edge $e \in E$ weighted by x_e. Certainly, d_x is a metric as the shortest path distance is symmetric, positive semi-definite, and abides by the triangle inequality. Using this, the first constraint of LP 11.1 now holds if and only if

$$d_x(s_i, t_i) \geq 1 \qquad\qquad \forall i = 1, \ldots, k$$

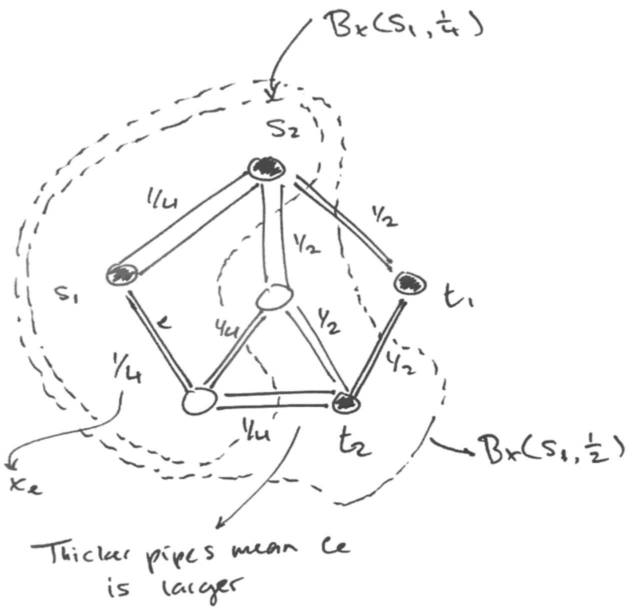
Another reason why this distance is useful is because we can use d_x to cut a set of vertices around any vertex $v \in V$. Define $\mathcal{B}_x(s_i, r)$ as the ball of radius r centered at s_i measured by distance d_x as the following

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

Let us see an example of these definitions.

11.1.4 An Example

Consider the following pipe network. The labels on the pipe represent x_e for the corresponding edge.



$\mathcal{B}_x(s_1, \frac{1}{4})$
s_2
$1/4$
$1/2$
$1/2$
s_1
t_1
e
$1/4$
$1/2$
$1/4$
$1/2$
$1/4$
t_2
$\mathcal{B}_x(s_1, \frac{1}{2})$
x_e
Thicker pipes mean c_e is larger

Notice the following about this system:

(1) The assignment of x_e's is feasible. All paths between (s_1, t_1) and (s_2, t_2) have at least distance one.

(2) The ball $\mathcal{B}_x(s_1, \frac{1}{4})$ contains the two source vertices s_1 and s_2.

(3) The ball $\mathcal{B}_x(s_1, \frac{1}{2})$ contains a source and sink pair s_2 and t_2.

The goal of MULTICUT is to find $F \subseteq E$ that separates each (s_i, t_i) pair. One potential strategy is to pick balls around each s_i and add edges crossing the boundary of the ball to our cut F. Doing so places s_i into its own partition, but we will need to be careful with how we choose the radius. If we choose a radius too large, such as $r = \frac{1}{2}$ in statement (3) above, then the ball may contain another source-sink pair causing that partition to be infeasible. Whereas choosing $r = \frac{1}{4}$ seemed reasonable as it separated both s_1 and s_2 from their respective sinks. Let us use this intuition to design the rounding algorithm.

11.2 The LP Rounding Algorithm

Our algorithm will construct the cut F by creating partitions around s_i's defined by the distance induced by the optimal solution to the LP relaxation d_x. By the above discussion, it should be that $r < \frac{1}{2}$, but perhaps an auspicious choice of r will allow us to control the cost of our cut. For now, we leave r as a parameter for our algorithm.

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

  1. Solve the linear program relaxation 11.1 for $x$ and set $F = \varnothing$.
  2. For $i = 1, \dots, k$ do:
       - If $s_i, t_i$ are not separated in $(V, E - F)$, then choose $\mathcal{B}_x(s_i, r)$
       - $F = F \cup \partial(\mathcal{B}_x(s_i, r))$ where $\partial(\mathcal{B}_x(s_i, r))$ denotes the edges crossing the boundary of $\mathcal{B}_x(s_i, r)$
       - Remove vertices in $\mathcal{B}_x(s_i, r)$ and all adjacent edges from $G$.
  3. Return $F$
```

We first demonstrate that this returns a feasible multicut.

Claim 11.3. *For $r \in [0, \frac{1}{2})$, algorithm 1 returns a feasible multicut.*

Proof. At each iteration of algorithm 1, we add edges to F separating a new partition surrounding s_i. For sake of contradiction, suppose the algorithm returns an infeasible multicut. There must exist an iteration i such that the partition removed contains a source-sink pair of vertices. Suppose at iteration i, the algorithm removed $\mathcal{B}_x(s_i, r)$. Because x is feasible and $r < \frac{1}{2}$, $d_x(s_i, t_i) \geq 1$ implying $t_i \notin \mathcal{B}_x(s_i, r)$. Consequently, there must exist $j \neq i$ such that $s_j, t_j \in \mathcal{B}_x(s_i, r)$. However, this implies that the following distances are bounded.

$$d_x(s_j, s_i) < \frac{1}{2} \qquad\qquad d_x(s_i, t_j) < \frac{1}{2}$$

By the triangle inequality, we have the following.

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

However, this contradicts feasability of x as it must be that $d_x(s_j, t_j) \geq 1$. We conclude that the algorithm returns a feasible multicut. □

11.3 Analyzing the Approximation-Ratio

Let us direct our attention towards bounding the approximation-ratio for algorithm 1 by showing.

Theorem 11.4. *Algorithm 1 returns a $4\ln(k+1)$-approximation for* MULTICUT

It will be useful for us to develop more machinery to discuss volume in our pipe system interpretation of the linear program relaxation as well as the cost of the partitions added to F. Let us define the following.

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

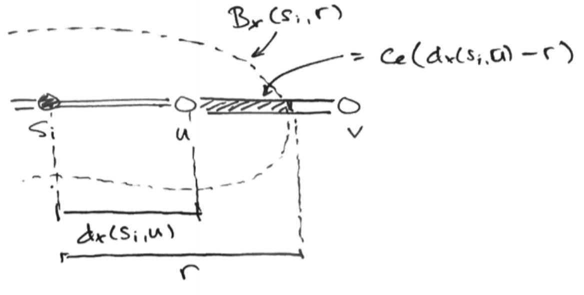
$$V^* = \sum_{e \in E} c_e x_e$$

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

(2) Define $V_x(s_i, r)$ to be the volume of pipes within the ball of radius r with an added $\frac{V^*}{k}$ term. More precisely, $V_x(s_i, r)$ is given by the following.

$$V_x(s_i, r) = \frac{V^*}{k} + \underbrace{\sum_{e=(u,v):u,v\in\mathcal{B}_x(s_i,r)} c_e x_e}_{\text{Total volume of pipe contained in the ball}} + \underbrace{\sum_{e=(u,v):u\in\mathcal{B}_x(s_i,r),v\notin\mathcal{B}_x(s_i,r)} c_e(r - d_x(s_i, u))}_{\text{Volume of pipe contained within the boundary}}$$

Together, the summations give the volume for the entire ball $\mathcal{B}(s_i, r)$. To visualize, consider the diagram:



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

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

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

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

We are now ready to analyze the cost of our cut.

11.3.1 Region Growing

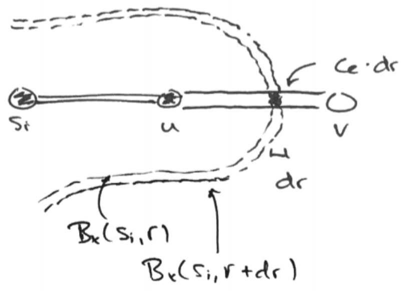
Consider a the ball $\mathcal{B}(s_i, r)$ removed during an iteration of algorithm 1. It is uncertain how we can handle the cost of cut $\partial(\mathcal{B}(s_i, r))$ directly, but suppose we could *charge* the cost of the cut to the volume of the ball. That is to say, we would like to discover α such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

$$c_x(s_i, r) \le \alpha \cdot V_x(s_i, r) \tag{11.2}$$

The first step toward this is to observe the following.

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

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 11.2 would then reduce to the following

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

But the fraction on the RHS 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 $r \in [0, \frac{1}{2})$

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

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

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

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

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

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 *region growing.* To summarize, we have demonstrated

Theorem 11.5. *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 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 later on in the notes and also demonstrate how r can be found in polynomial time. For now suppose we have theorem 11.5. We can now complete our analysis of the approximation ratio.

11.3.2 The Approximation Ratio

Recall theorem 11.4 which states that algorithm 1 produces a $4\ln(k+1)$-approximation for MULTICUT. We now produce the proof assuming $r < \frac{1}{2}$ has been chosen according to theorem 11.5.

Proof. At each iteration of algorithm 1, we add $F_i = \partial(\mathcal{B}_x(s_i, r))$ to F. Let us also denote V_i to be the volume of pipes contained in the ball $\mathcal{B}_x(s_i, r)$, that is

$$V_i = V_x(s_i, r) - \frac{V^*}{k}$$

For iterations i where no ball is removed, as s_i may have been separated from t_i from a previous iteration, we consider $F_i = \varnothing$ and $V_i = 0$. For our choice of r, theorem 11.5 gives the cost of F_i.

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

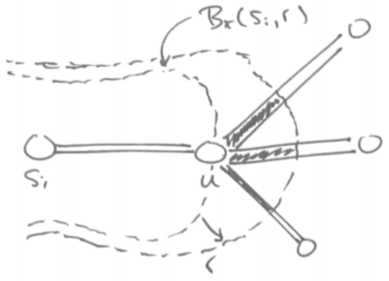
Since $\mathcal{B}_x(s_i, r)$ is removed from G at every iteration along with all adjacent edges, $F_i \cap F_j = \varnothing$ for $i \neq j$. Additionally, the volume measured by V_i will not overlap with that measured by V_j for $i \neq j$. The total cost of F returned by the algorithm is the following.

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

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

Finally, we revisit our the continuity of $F(r) = \ln(V_x(s_i, r))$ and ask where $F(r)$ could be discontinuous. Notice that 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 11.5.

Proof. 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 *expected value* of $\frac{c_x(s_i,r)}{V_x(s_i,r)}$ to be what we want, then there must exist an actual choice of r where the bound holds deterministically. Using the fact that the support of the expectation has to be non-empty in this fashion is also known as the *probabilistic method.*

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 $[0, \frac{1}{2})$.

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

$$
\begin{aligned}
&= 2 \cdot \sum_{j=1}^{\ell} \Big(\ln(V_x(s_i, r_{j+1}^-)) - \ln(V_x(s_i, r_j)) \Big) \\
&\le 2 \cdot \sum_{j=1}^{\ell} \Big(\ln(V_x(s_i, r_{j+1})) - \ln(V_x(s_i, r_j)) \Big)
\end{aligned}
$$

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

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

Thus the expectation is bounded by $2\ln(k+1)$ hence there must be an r achieving $\frac{c_x(s_i,r)}{V_x(s_i,r)} \le 2\ln(k+1)$. 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}. To find a point where this ratio is minimized requires to only check each $r_1, \ldots, r_\ell$. Because $\ell \le n$, a linear number of computations suffice. □

11.4 Final Remarks

There were two key ideas highlighted by these notes: (1) the method of constructing a metric from a feasible solution to some LP and (2) growing regions for which we can charge the cost of the boundary to its volume.

Constructing a metric from an LP feasible solution can be traced to a paper by Leighton and Rao [4][5] where they use this technique to construct low-cost graph partitions for various cut problems. The idea of region growing also appears in this paper for approximating a problem known as uniform sparsest cut. Another result by Arora, Rao, Vazirani [1] uses these two ideas to round a semidefinite program for uniform sparsest cut, where they are able to produce an $O(\sqrt{\log n})$-approximation.

Other domains where region growing has found success is in finding random spanning trees of a graph G. Here, one wishes to produce a spanning tree of G sampled uniformly at random from the set of all spanning trees of G. A fast primitive for solving this problem has a number of applications in graph sparsification, and constructing efficient solvers for certain linear systems. Kelner and Madry are able to find a fast algorithm using region growing to decompose G into a collection of sparsely connected cuts [3].

On the topic of multicut, algorithm 1 was first provided by Garg, Vazirani, and Yannakakis [2]. It is not known if this algorithm achieves the optimal approximation ratio. Under the Unique Games Conjecture, one only knows of the following result.

Theorem 11.6. *Assuming the Unique Games Conjecture, there does not exist an α-approximation for* MULTICUT *for any $\alpha \ge 1$ 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 $\mathsf{P} = \mathsf{NP}$. Even improving the constant of 4 would lead to an algorithm that is not currently known!

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] 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.

[3] 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.

[4] 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.

[5] 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.