**E0 249: Approximation Algorithms** January 31, 2022

Week 5
Lecture 1: Min s-t cut

*Instructors: Arindam Khan, Anand Louis* *Scribe: Shirish Gosavi*

# 1 Introduction

In this lecture we will look at the minimum weight $s$-$t$ cut problem, the LP-relaxation of the problem along with a randomized rounding algorithm to find the mincut and the dual LP which is the max $s$-$t$ flow problem. The multiway cut is a generalization of the minimum $s$-$t$ cut problem. It will be discussed in detail in the next lecture.
We will begin the section **2 Min s-t cut** with the definition of **Metric(V,d)**. *Metrics* turn out to be a useful way of thinking about graph problems involving cuts. We will try to explore connections between cuts in graphs and properties of *metrics* to design approximation algorithms.
In the section **3 Dual of the LP**, we will present the dual of the LP for the min $s$-$t$ cut problem. The dual is the LP for the max $s$-$t$ flow problem.
In the section **4 The mincut polytope**, we will show integrality of the solution of LP for the min $s$-$t$ cut problem.
Some general references used are [Vaz13] and [WS11].

# 2 Min *s-t* cut

**Definition 1** (**Metric(V,d)**). *A metric (V, d) on a set of vertices V gives a distance $d_{uv}$ for each pair of vertices $u, v \in V$ such that three properties are obeyed [WS11]:*

1. $d_{uv} = 0$ *if and only if* $v = u$.
2. $d_{uv} = d_{vu}$ *for all* $u, v \in V$.
3. $d_{uv} \leq d_{uw} + d_{wv}$ *for all* $u, v, w \in V$.

We start with designing an approximation algorithm to find **min *s-t* cut**. The problem is defined as follows.

**Definition 2** (**Min *s-t* cut**). *Given a graph G = (V, E, w), where $w : E \to \mathbb{R}^+$ and vertices $s, t \in V$, compute the least weight set of edges to "cut" such that s and t are disconnected.*

Throughout the discussion, we assume that the underlying graph is undirected.

We start with the integer program formulation as the first step. We use the following notations:

- Variable $x_e$ for each edge $e \in E$ is used to indicate whether $e$ is in the cut. The variable $x_e$ will be set to 1 if the corresponding edge $e$ is in the cut, and will be set to 0 otherwise.
- $P$ denotes the set of paths from $s$ to $t$.

The integer program formulation will be as follows:

$$\min \quad \sum_{e \in E} w_e \cdot x_e$$

subject to

$$\sum_{e \in q} x_e \geq 1 \quad \forall q \in P \tag{1}$$

$$x_e \in \{0, 1\} \quad \forall e \in E \tag{2}$$

The objective function is $\sum_{e \in E} w_e \cdot x_e$ and it indicates the cost of a cut. Only weights of edges in the cut contribute to the sum as $x_e$ is 1 only for such edges and is 0 for all other edges. We want to minimize the objective function.
The constraint (1) captures the requirement that every path between $s$ and $t$ has to have at least one edge in common with the cut. The constraint (2) enforces the condition that every $x_e$ must take one of two values 0 and 1.

Since finding optimal solution of integer program is NP-hard, we use LP-relaxation. In the integer program given above, the constraint 1 is linear, but the constraint 2 is not. We change the non-linear constraint (2) to:

$$0 \leq x_e \leq 1 \quad \forall e \in E \tag{3}$$

Further, we are trying to minimize the objective function and the coefficients of variables in the primal are all non-negative, we don't have to explicitly mention that $x_e \leq 1$. Therefore, the modified constraint 3 would look like:

$$x_e \geq 0 \quad \forall e \in E \tag{4}$$

The complete LP-relaxation will be as follows:

$$\min \quad \sum_{e \in E} w_e \cdot x_e$$

subject to

$$\sum_{e \in q} x_e \geq 1 \quad \forall q \in P \tag{5}$$

$$x_e \geq 0 \quad \forall e \in E \tag{6}$$

We can use ellipsoid method to solve the above LP problem in polynomial time. To check feasibility of a solution, we need to verify satisfiability of both the constraints. Checking constraint (6) is easy as the number of edges is finite. However, checking the first constraint (5) can be tricky as the number of paths can be very large, even exponential in some cases (for example, complete graphs). To verify the constraint (5), we can find a shortest path between $s$ and $t$ using $x_e$ as edge lengths. If the length of the shortest path found is greater than or equal to 1, then the constraint is satisfied (because non-zero length would imply that the path has at least one edge belonging to the cut). If the shortest path has length greater than or equal to 1, all paths between $s$ and $t$ would have length greater than or equal to 1 and this would mean each path has at least one edge belonging to the cut. Thus, the constraint (5) can be verified in polynomial time.

However, we are using relaxed LP instead of the original integer program to get the solution in polynomial time and we may get fractional values for $x_e$ variables. Let $\boldsymbol{x}$ be an optimal solution to the fractional LP. For an edge $(i, j) \in E$, we can view $x_{(i,j)}$ as the "distance" between the vertices $i$ and $j$. For vertices $i, j \in V$, define *d(i,j)* to be the length of the shortest path between $i$ and $j$.
The $d(i, j)$ definition meets all the conditions required to qualify it as a semi-metric as shown below:

1. *d(i,i)* $= 0$ is obvious. However, *d(i,j)* could be 0 for some $i \neq j$. Therefore, *d(i,j)* satisfies the first condition of metric definition partially, but meets the first condition for semi-metric completely.
2. $d(i, j) = d(j, i)\ \forall i, j \in V$ is obvious as we are dealing with undirected graphs.
3. $\forall i, j, k \in V, d(i, j) + d(j, k) \geq d(i, k)$ . If not, we can find a shorter path *i-j-k* with cost *d(i,j)+d(j,k)* which is smaller than *d(i,k)*. This will contradict the definition of *d(i,j)*.

Even though technically *d(i,j)* is a semi-metric and not a metric, the difference is not going to affect the rest of the discussion. Therefore, we simply treat it as a metric and write the following lemma.

**Lemma 3.** *d is a metric on V.*

We use randomized rounding to convert the fractional LP solution to a feasible solution for the original integer program. We can use the following Algorithm 1 for randomized rounding.
The set of edges returned by the Algorithm 1 may not form a valid cut. It is because, there may be an *s-t* path such that none of its edge gets selected in the sample even though it has some edge(s) with non-zero value(s).
The probability of not getting a valid cut can be reduced to any desired value by executing the Algorithm 1 multiple times. We can use an alternative Algorithm 2 for rounding.

**Algorithm 1:** Algorithm 1 for Min s-t cut

1. Sample each edge $e \in E$ with probability $x_e$.
2. Output the set of edges sampled.

**Algorithm 2:** Algorithm 2 for Min s-t cut

1. Sample $r \in [0, 1)$ uniformly at random.
2. Let $A_r \stackrel{def}{=} \{i \in V : d(s, i) \leq r\}$.
3. Output the set of edges leaving $A_r$.

Note that the Algorithm 2 always outputs a feasible cut. This is because *d(s,s)* $= 0$ and for any $0 \leq r < 1$, *d(s,s)* is less than or equal to *r*. On the other hand, the shortest distance between $s$ and $t$ is greater than or equal to 1. If not, then there will be a path from $s$ to $t$ with total cost less than 1 and the solution $\boldsymbol{x}$ will not be a feasible solution as the constraint 5 will be violated. Therefore, $s \in A_r$ and $t \notin A_r$.

Now we will show that the Algorithm 2 actually outputs a mincut, that is, it is 1-approximation algorithm.

First we will prove the following lemma.

**Lemma 4.** Pr[*the vertex pair* $(i, j)$ *is cut*] $\leq d(i, j)$

*Proof.* Without loss of generality, assume that that $d(s, i) \leq d(s, j)$. The probability Pr[the vertex pair $(i, j)$ is cut] can be computed using the following equation:

$$\Pr[\text{the vertex pair } (i, j) \text{ is cut}] = \Pr[d(s, i) \leq r \text{ and } d(s, j) > r]$$

It may be noted that, all the *d(u,v)* values are in the range [0, 1] and we are choosing $r$ uniformly from [0, 1). We want $r$ to be present in the interval [*d(s,i), d(s,j)*].
Therefore,

$$\Pr[d(s, i) \leq r \text{ and } d(s, j) > r] = d(s,j) \text{ - } d(s,i) \tag{7}$$

As already discussed, $d$ is a metric and hence satisfies triangle inequality. Therefore,

$$d(s,i) + d(i,j) \geq d(s,j) \tag{8}$$

Combining 7 and 8, we get:

$$\Pr[d(s,i) \leq r \text{ and } d(s,j) > r] \leq d(i,j) \tag{9}$$

That is,

$$\Pr[\text{the vertex pair } (i,j) \text{ is cut}] \leq d(i,j)$$

Hence proved. □

If (i,j) is an edge in $E$, $d(i,j) \leq x_{(i,j)}$ by definition of *d(i,j)*. Therefore, applying the lemma 4 to an edge $(i,j) \in E$, we can infer that

$$\Pr[\text{ the vertex pair } (i,j) \text{ is cut}] \leq x_{(i,j)}$$

Let *Algcut* denote cut output by the algorithm 2 and *w(Algcut)* be the the weight of *Algcut*.

$$E[w(Algcut)] = \sum_{e \in E} w_e \cdot \Pr[e \in Algcut] \leq \sum_{e \in E} w_e \cdot x_e$$

But $\sum_{e \in E} w_e \cdot x_e$ is nothing but the objective function value of the relaxed LP solution.
Therefore,

$$E[w(Algcut)] \leq LP \tag{10}$$

We also have the following inequalities:

$$LP \leq mincut \tag{11}$$

$$mincut \leq w(Algcut) \tag{12}$$

The inequality 11 follows from the fact that $LP$ is the solution of the LP-relaxation and the inequality 12 holds because *mincut* is optimal solution and *Algcut* is the solution obtained after rounding.
From the inequalities 10, 11 and 12, we get:

$$LP = mincut = E[w(Algcut)] = LP$$

Since every cut output by the algorithm 2 has weight $\geq mincut$, if even one of them has weight $> mincut$, then to satisfy $\boldsymbol{mincut = \mathbf{E}[w(Algcut)]}$, some cut output by the algorithm 2 must have weight $< mincut$, which is not possible. Therefore, the Algorithm 2 always outputs a *mincut*.

# 3 Dual of the LP

$$\min \quad \sum_{e \in E} w_e \cdot x_e$$

subject to

$$\sum_{e \in q} x_e \geq q \quad \forall q \in P \tag{13}$$

$$x_e \geq 0 \quad \forall e \in E \tag{14}$$

Introduce multipliers $f_q$ for each constraint.
The dual can be written as:

$$\max \quad \sum_{q \in P} f_q$$

subject to

$$\sum_{q: e \in q} f_q \leq w_e \quad \forall e \in E \tag{15}$$

$$f_q \geq 0 \quad \forall q \in P \tag{16}$$

The dual represents the LP for max-flow problem. $f_q$ is the amount of flow through path $q \in P$, and $w_e$ is the capacity of each edge.
Thus, dual (= max-flow problem) corresponds to the primal (= min $s$-$t$ cut problem).
By LP duality, we know that max $s$-$t$ flow = min $s$-$t$ cut.

# 4 The mincut polytope

Let $e_1, ..., e_m$ be an arbitrary ordering of the edges. The LP solution can be viewed as $x \in \mathcal{R}^m$. For a set of vertices $S$, let $\delta(S)$ denote the set of edges leaving $S$ and let $\mathbb{I}_{\delta(S)} \in \{0,1\}^m$ denote its indicator vector.
Sort vertices in increasing order of *d(s,x)*, let vertex $i$ denote the $i^{th}$ vertex in this ordering and let $S_i$ be the set of the first $i$ vertices in this ordering. For each $i \in [n-1], S_i$ is a feasible s-t cut.

**Lemma 5.** *Let* $y_i \stackrel{def}{=} d(s,i)$. *Then* $x = \sum_{l=1}^{n-1} (y_{l+1} - y_l) \mathbb{I}_{\delta(S_l)}$.

*Proof.* Fix an edge $(i,j) \in E$. Wlog, assume $y_j \geq y_i$. Then $x_{(i,j)} = y_j - y_i$. This can be justified as follows:

1. $x_{(i,j)} > y_j - y_i$ : It means there is a shorter path from $i$ to $j$ than the direct edge between the two vertices. In such a case, we can bypass edge $(i,j)$ altogether in all the shortest paths. But then the optimal solution would not have given non-zero $x_{(i,j)}$ value corresponding to the edge $(i,j)$. If $x_{(i,j)}$ is 0, then it cannot be greater than $y_j - y_i$ as $j$ is appearing after $i$ in the sorted order based on the shortest distances from $s$.
   Therefore, this scenario is not possible.

2. $x_{(i,j)} < y_j - y_i$ : In this case, we can get a shorter path $s$-$i$-$j$ from $s$ to $j$ with cost less than $y_j$. Therefore, this scenario is also not possible.

We can write

$$x_{(i,j)} = y_j - y_i = \sum_{l=i}^{j-1} (y_{l+1} - y_l)$$

We note the following:

1. $(i,j) \notin \delta(S_l)$ for $l \leq (i-1)$
   If $(i,j) \in \delta(S_l)$ for some $l \leq (i-1)$, the $j^{th}$ vertex must appear before $j^{th}$ position in the sorted sequence.

2. $(i,j) \notin \delta(S_l)$ for $l \geq j$
   If $(i,j) \in \delta(S_l)$ for some $l \geq j$, the $i^{th}$ vertex must appear after $i^{th}$ position in the sorted sequence.

Therefore,

$$(i,j) \in \delta(S_l) \text{ for } i \leq l \leq j-i$$

It means,

$$\langle x, \mathbb{I}_{(i,j)} \rangle = \sum_{l=1}^{n-1} (y_{l+1} - y_l) \langle \mathbb{I}_{\delta(S_l), \mathbb{I}_{(i,j)}} \rangle$$

□

Lemma 5 implies that $x$ is a convex combination of cuts.
$\sum_{l=1}^{n-1} (y_{l+1} - y_l) = y_n - y_1 = d(s,t) - d(s,s) = 1$
If $x$ is a vertex solution of the LP, then it can not be written as a convex combination of two feasible solutions. Then $y_{l+1} - y_l$ is non-zero for only one value of $l \in [n-1]$.
Therefore, $y_l \in \{0, 1\}$ $\forall l \in [n]$.

# References


[Vaz13] V.V. Vazirani. *Approximation Algorithms*. Springer Berlin Heidelberg, 2013.

[WS11] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*, page 469–484. Cambridge University Press, 2011.