1 Max-flow Min-cut Theorem (LP version)

In this section, we give a proof of max-flow min-cut theorem using LP rouding. We first write down the LP relaxation for the problem of finding minimum s-t cut, and show how to round the LP to get a cut of value OPT_{LP}. This implies that there is a cut of the same value as that of maximum flow (since max-flow is the dual of min-cut).

Let $G = (V, E)$ be any graph with capacity function $c : V \times V \to \mathbb{R}$. We are interested in computing a cut $S \subseteq V$ such that $|\{s, t\} \cap S| = 1$ while minimizing $c(S) = \sum_{(u,v)\in E: u\in S, v\notin S} c(u, v)$. Let us first write down the LP relaxation for the problem of single source single sink cut. We have a variable $x(u, v)$ for each edge (u, v) in the graph G. The intended meaning of this variable is to indicate whether an edge (u, v) is cut.

$$
\begin{aligned}
&\text{(LP)} \\
&\qquad \min \quad \sum_{(u,v)\in E} c(u,v)x(u,v) \\
&\qquad \text{s.t.} \quad \sum_{(u,v)\in P} x(u,v) \geq 1 \text{ for all path } P \text{ from } s \text{ to } t \\
&\qquad\qquad x(u,v) \in [0,1]
\end{aligned}
$$

The set of constraints enforces that, for each path P connecting s and t, at least one edge must belong to the cut. Notice that this is a relaxation of the problem, because for each cut S separating s and t, any path P starting from s must cross from S to $V \setminus S$ at some point.

1.1 Solving this LP in polynomial time

Even though this LP has exponential number of constraints (i.e. the number of paths from s to t can be exponential), one can show a separation oracle for this LP as follows. Given a tentative solution x, we can compute shortest s-t path from s to t using the distance function x. That is, define the set $\mathcal{P}_v$ as the set of paths starting from s and ending at v. We can define $d_x(v) = \min_{P\in\mathcal{P}_v} \sum_{(u,w)\in P} x(u, w)$.

Claim 1.1. *Solution x violates some constraint in (LP) if and only if $d_x(t) < 1$.*

The value of $d_x(t)$ can be computed using any shortest path algorithm, so we have shown a separation oracle for (LP).

1.2 LP Rounding

Define a mapping of vertices onto the real line $\mathbb{R}$ as follows: Let $\phi(s) = 0$, and for each vertex $v \in V \setminus \{s\}$, we define $\phi(v)$ as the shortest path length (using metric $x(\cdot, \cdot)$) from s to v. It is clear from this definition that $\phi(t) = 1$. Now we are ready to describe the algorithm:

- Pick a random value $\Delta \in (0, 1)$
- Create a cut (A, B) where A contains all vertices v with $\phi(v) \leq \Delta$, and $B = \bar{A}$.

The following claim follows immediately from the fact that $\phi(s) < \Delta < \phi(t)$.

Claim 1.2. *The cut (A, B) is feasible, i.e. $s \in A$ and $t \in B$.*

Now the next claim argues that the cost of the resulting cut is low in expectation.

Claim 1.3. *The expected cost of the cut (A, B) is* OPT.

Proof. Let Y be a random variable that denotes the size of the cut $\sum_{(u,v)\in(A,B)} c(u, v)$. We can write this term as $\sum_{(u,v)\in E:\phi(u)<\phi(v)} c(u, v)Y_{uv}$ where Y_{uv} is an indicator random variable denoting whether (u, v) crosses the cut (A, B).

By linearity of expectation, we have $\mathbf{E}[Y] = \sum_{(u,v)\in E} c(u, v)\mathbf{E}[Y_{uv}]$. Since Y_{uv} is an indicator rv, we have $\mathbf{E}[Y_{uv}] = \mathbf{Pr}[(u, v) \text{ is cut}]$. Since $\phi(u) \leq \phi(v)$, the probability that (u, v) is cut is the same as the probability that "$u \in A$ and $v \in B$" which is again the same as "$\phi(u) \leq \Delta \leq \phi(v)$". This probability is at most $\phi(v) - \phi(u) \leq x(u, v)$ due to the fact that $\phi(\cdot)$ is a shortest path function.

All these imply that $\mathbf{E}[Y] \leq \sum_{(u,v)\in E} c(u, v)x(u, v) = \mathsf{OPT}$. □

Derandomization: Now we have an algorithm whose cost is OPT in expectation. What if we want a deterministic algorithm? Notice that this analysis implies that, there is a value $\Delta \in (0, 1)$ such that $A_\Delta = \{v : \phi(v) \leq \Delta\}$ and $B_\Delta = \{v : \phi(v) > \Delta\}$ is a cut of cost at most OPT. If we reorder the vertices in V such that $V = \{v_1, \ldots, v_n\}$ where $\phi(v_1) \leq \phi(v_2) \leq \ldots \leq \phi(v_n)$, then for any $\Delta \in (\phi(v_i), \phi(v_{i+1}))$, the cut (A_Δ, B_Δ) is the same. Therefore, there can be at most n different cuts, and one such cut must have its value at most OPT. Our algorithm can simply try all possible cuts.