1 Minimum cut trees

We can find a minimum cut in $f(n)$ for any pair of vertices using flows[1]. What if we wanted to find the minimum cut for all pairs of vertices? $O(n^2 f(n))$ is enough by running n^2 flows. But we can do better.

Definition 1.1. *A* Gomory-Hu tree *T of a weighted graph G is a weighted tree with vertex set $V(G)$ and the two components of $T - uv$ are the vertex sets of a minimum weight u-v cut for each $uv \in E(T)$.*

It is not clear that such a tree should even exist. But if it did, this suggests a nested/laminar structure to minimum cuts in a weighted graph.

First, suppose that such a tree exists. Then notice that we can indeed find the minimum weight cut for any pair of vertices, not just adjacent ones in T.

Lemma 1.2. *The weight of the minimum cut between any two vertices u, v is the (weight of the) minimum weight edge on the path from u to v in T.*

Proof. Any u-v cut separates some consecutive pairs of vertices on this path (it may separate many such pairs). So the minimum cut's weight cannot be below the minimum weight of an edge on the path.

It cannot be above either as the definition of the Gomory-Hu tree tells us all cuts on the path are u-v-cuts. □

Like in the odd cut theorem, we will (algorithmically) prove a containment lemma. This lemma will again suggest a recursive algorithm. This *uncrossing*/use of *laminarity* is an important concept and often used tool.

Lemma 1: Let $\delta(S)$ be a minimum s-t cut with $u \in S$. Then for any $u, v \in S$, there is a minimum u-v cut $\delta(W)$ with $W \subset S$.

The proof is also similar to the odd cut proof.

Proof. Again, take two minimum cuts S and U. Without loss of generality, $s \in U$. There are two cases to consider.

Case 1: $t \notin U$

So $w(\delta(S)) \leq w(\delta(V - S - U)$ and

$$\begin{aligned} w(\delta(S)) + w(\delta(U)) &\geq w(\delta(S \cap U)) + w(\delta(V - S - U)) \\ w(\delta(U)) &\geq w(\delta(S \cap U)) \end{aligned}$$

And $W = S \cap U$ is a minimum u-v cut contained in S.

Case 2: $t \in U$

So $w(\delta(S)) \leq w(\delta((V - S) \cap U)$ and

$$\begin{aligned} w(\delta(S)) + w(\delta(U)) &\geq w(\delta(S \cap (V - U))) + w(\delta((V - S) \cap U) \\ w(\delta(U)) &\geq w(\delta(S \cap (V - U))) \end{aligned}$$

[1] $f(n) = O(n^2 m) = O(n^4)$ using blocking flows; the best known algorithm runs in $O(nm) = O(n^3)$

And $W = S \cap (V - U)$ is a minimum u-v cut contained in S, as required. $\square$

Again, like in the odd cut case, this suggests a recursive algorithm.

We will construct a tree with nodes labelled by subsets of $V(G)$ that form a partition of $V(G)$.

The initial tree T is a single node labelled by $V(G)$. We then iteratively transform it into a Gomory-Hu tree.

Algorithm

Repeat until all nodes are labelled by a single vertex:

1. Pick any node t_L whose label L contains at least two vertices $s, t \in L$.
2. Construct the auxiliary graph G' obtained from G by contracting for each subtree of $T - t_L$, the union of labels in that subtree to a single vertex.
3. Find a min s-t cut $\delta(S)$ in G'. Say this cut has weight m_{st}.
4. Split t_L into two vertices t_S and $t_{L \setminus S}$.
5. Make each vertex adjacent to t_L adjacent to either t_S or $t_{L \setminus S}$ depending the side of $\delta(S)$ they are on.
6. Add an edge (in T) between $s \in S$ and $t \in L \setminus S$ of weight m_{st}.

In this description, edges remember what their endpoints were when we split a node. Alternatively, we could just track the labels (partition of $V(G)$) and the edges of the tree we found with their endpoints and weights.

We can then reassemble them into a tree.

Theorem 1.3. *This algorithm produces a Gomory-Hu tree.*

Proof. We need to check that for each edge $uv \in E(T)$, the two vertices of each component of $T - uv$ do form a minimum weight u-v cut.

Consider an arbitrary edge $uv \in E(T)$. At the time it was added uv to T by splitting a vertex t_{L^*}, a number of other edges are already present. To see that a minimum u-v cut contained in t_{L^*} is a minimum u-v cut overall, apply Lemma 1 repeatedly to each edge of T added before uv, in the order they were added. Each time, a vertex t_L is split in two and u, v are both contained in either t_S or $t_{L \setminus S}$, say it is S. By Lemma 1, there is a minimum u-v contained in S that is a minimum u-v contained in L.

Furthermore, splits after uv do not affect this cut as vertex sets for $T - uv$ are unchanged when we split another vertex. $\square$

1.1 Applications

We can also use this tree to find the minimum weight odd cut! Build the Gomory-Hu tree. Then take the min amongst all edges of the tree that give an odd cut.

(Exists because leaves are all odd cuts.)