16 Minimum Cuts

Let $G$ be an arbitrary graph, and let $s$ and $t$ be two vertices of $G$. An $(s, t)$-cut (or more formally, an $(s, t)$-edge-cut) is a subset $X$ of the edges of $G$ that intersects every path from $s$ to $t$ in $G$. A minimum $(s, t)$-cut is an $(s, t)$-cut of minimum size, or of minimum total weight if the edges of $G$ are weighted. (In this context, edge weights are normally called capacities.)

The fastest method to compute minimum $(s, t)$-cuts in arbitrary graphs is to compute a maximum $(s, t)$-flow and then exploit the classical proof of the maxflow-mincut theorem. In undirected planar graphs, however, this dependency is reversed; the fastest method to compute maximum flows actually computes minimum cuts first.

16.1 Duality: Shortest essential cycle

Let $\Sigma$ be an undirected planar map, each of whose edges $e$ has a non-negative capacity $c(e)$, and let $s$ and $t$ be vertices of $\Sigma$. The first step in our fast minimum-cut algorithm is to view the problem through the lens of duality. It is helpful to think of the source vertex $s$ and the target vertex $t$ as punctures or obstacles — points that are missing from the plane — and similarly to think of the corresponding faces $s^*$ and $t^*$ as holes in the dual map $\Sigma^*$. In other words, we should think of the dual map $\Sigma^*$ as a decomposition of the annulus $A = \mathbb{R}^2 \setminus (s^* \cup t^*)$ rather than a map on the plane or a disk. Without loss of generality, assume that $t^*$ is the outer face of $\Sigma^*$.

A simple cycle $\gamma$ in $\Sigma^*$ is essential if it satisfies any of the following equivalent conditions:

- $\gamma$ separates $s^*$ from $t^*$.
- $\gamma$ has winding number $\pm 1$ around $s^*$.
- $\gamma$ is homotopic in $A$ to the boundary of $s^*$.
- $\gamma$ is homotopic in $A$ to the boundary of $t^*$.

Each edge $e^*$ in the dual map $\Sigma^*$ has a cost or length $c^*(e^*)$ equal to the capacity of the corresponding primal edge $e$. Whitney’s duality between simple cycles in $\Sigma$ and and minimal cuts (bonds) in $\Sigma^*$ immediately implies the following:

**Lemma:** A subset $X$ of edges is a minimum $(s, t)$-cut in $\Sigma$ if and only if the corresponding set $X^*$ of dual edges is a minimum-cost essential cycle in $\Sigma^*$.

![Figure 1: A minimum $(s, t)$-cut in a planar graph is dual to a shortest essential cycle in the annular dual map.](image-url)
16.2 Crossing at most once

Now let $\pi$ be a shortest path in $\Sigma^*$ from any vertex of $s^*$ to any vertex of $t^*$. We can measure the winding number of any directed cycle $\gamma$ in $\Sigma^*$ by counting the number of times $\gamma$ crosses $\pi$ in either direction. We have to define “crossing” carefully here, because $\gamma$ and $\pi$ could share edges.

Suppose $\pi = p_0 \rightarrow p_1 \rightarrow \cdots \rightarrow p_k$, where $p_0$ lies on $s^*$ and $p_k$ lies on $t^*$. To simplify the following definition, we add two “ghost” darts $p_{-1} \rightarrow p_0$ and $p_k \rightarrow p_{k+1}$, where $p_{-1}$ lies inside $s^*$ and $p_{k+1}$ lies inside $t^*$. We say that a dart $qp_i$ enters $\pi$ from the left (resp. from the right) if the darts $p_{i-1} \rightarrow p_i$, $qp_i$, and $p_{i+1} \rightarrow p_i$ are ordered clockwise (resp. counterclockwise) around $p_i$. Symmetrically, we say that a dart leaves $\pi$ to the left (resp. to the right) if its reversal enters $\pi$ from the left (resp. from the right). The same dart can leave $\pi$ to the right and enter $\pi$ to the left.

A positive crossing between $\pi$ and $\gamma$ is a subpath of $\gamma$ that starts with a dart entering $\pi$ from the right and ends with a dart leaving $\pi$ to the left, and a negative crossing is defined similarly.

Intuitively, for purposes of defining crossings, we are shifting the path $\pi$ very slightly to the left, so that it intersects the edges of $\Sigma^*$ transversely. It follows that the winding number $\text{wind}(\gamma, s^*)$ is the number of darts in $\gamma$ that leave $\pi$ to the left, minus the number of darts in $\gamma$ that enter $\pi$ from the left.

**Lemma:** The shortest essential cycle in $\Sigma^*$ crosses $\pi$ exactly once.

**Proof:** We follow the same intuition that we used for shortest homotopic paths in the plane.

Let $\gamma$ be any essential cycle in $\Sigma^*$, directed so that $\text{wind}(\gamma, s^*) = 1$, that crosses $\pi$ more than once. Then $\gamma$ must have a negative crossing followed immediately by a positive crossing. It follows that $\gamma$ has a subpath $p_i \rightarrow q \rightarrow \cdots \rightarrow p_j$, where $p_i \rightarrow q$ leaves $\pi$ to the right, $r \rightarrow p_j$ enters $\pi$ from the right. Let $\gamma'$ be the cycle obtained from $\gamma$ by replacing this subpath with the subpath of $\pi$ from $p_i$ to $p_j$. Because $\pi$ is a shortest path, $\gamma'$ must be shorter than $\gamma$. We conclude that $\gamma'$ is not the shortest essential cycle in $\Sigma^*$. \qed

16.3 Slicing along a path

Now let $\Delta := \Sigma^* \setminus \pi$ denote the planar map obtained by slicing the annular map $\Sigma^*$ along path $\pi$. The slicing operation replaces $\pi$ with two copies $\pi^+$ and $\pi^-$. Then for every vertex $p_i$ of $\pi$, all edges incident to $p_i$ on the left are redirected to $p_i^+$, and all edges incident to $p_i$ on the left are redirected to $p_i^-$. The channel between two two paths $\pi^+$ and $\pi^-$ joins $s^*$ and $t^*$ into a single outer face. Thus, we should think of $\Delta$ as being embedded on a disk. Every face of $\Sigma^*$ except $s^*$ and $t^*$ appears as a face in $\Delta$.  

2
Figure 3: Any essential cycle that crosses $\pi$ more than once can be shortened.

Figure 4: Slicing along $\pi$.

For any index $i$, let $\sigma_i$ denote the shortest path in $\Delta$ from $p_i^+$ to $p_i^-$. The shortest essential cycle $\gamma$ in $\Sigma^*$ appears in $\Delta$ as one of the shortest paths $\sigma_i$. Thus, to compute the minimum $(s, t)$-cut in our original planar map $\Sigma$, it suffices to compute the length of every shortest path $\sigma_i$ in $\Delta$.

Figure 5: Slicing along $\pi$ transforms the shortest essential cycle into a shortest path between the clones of some vertex of $\pi$.

16.4 Algorithms

The simplest way to compute these $k$ shortest-path distances is to run Dijkstra’s algorithm at each vertex $p_i^+$. Assuming $\pi$ has $k$ edges, so there are $k + 1$ terminal pairs $p_i^\pm$, this algorithm runs in $O(kn \log n)$ time, which is $O(n^2 \log n)$ time in the worst case. We can reduce the running time to $O(kn \log \log n)$ using the faster shortest-path algorithm we described in a previous lecture note, or even to $O(kn)$ using a linear-time shortest-path algorithm.
Alternatively, we can compute all $k$ of these shortest paths in $O(n \log n)$ time using the multiple-source shortest-paths algorithm. This algorithm is faster in the worst case, but slower than the previous algorithm when $k$ is small. Even when $k = 2$, the MSSP algorithm could require $\Omega(n)$ pivots.

John Reif (1983) proposed a divide-and-conquer algorithm that beats both of these time bounds. Reif’s algorithm computes the median shortest path $\sigma_m$, where $m = \lfloor k/2 \rceil$, and then recurses in each component of the sliced map $\Delta \setminus \sigma_m$. One of these components contains the terminal pairs $p^+_0, p^+_1, \ldots, p^+_m$; the other contains the terminal pairs $p^-_m, p^-_{m+1}, \ldots, p^-_k$.

Reif’s algorithm falls back on Dijkstra’s algorithm in two base cases. First, if $k \leq 2$, we can obviously compute each of the $k$ shortest paths directly. A more subtle base case happens when the “floor” and “ceiling” paths collide. Let $\alpha$ denote the boundary path from $p^+_0$ to $p^-_0$, and let $\beta$ denote the boundary path from $p^+_k$ to $p^-_k$. If $\alpha$ and $\beta$ share a vertex $x$, then for every index $i$ we have $dist(p^+_i, p^-_i) = dist(p^+_i, x) + dist(x, p^-_i)$; thus, instead of recursing, we can compute all $k$ shortest-path distances by computing a single shortest-path tree rooted at $x$. (This second base case is not necessary for the correctness of Reif’s algorithm, but it is necessary for efficiency.)

Let $T(n, k)$ denote the running time of Reif’s algorithm, where $k + 1$ is the number of terminal pairs and $n$ is the total number of vertices in the map $\Delta$. This function obeys the recurrence

$$T(n, k) = T(n_1, \lfloor k/2 \rceil) + T(n_2, \lceil k/2 \rceil) + O(n \log n).$$

where $n_1$ and $n_2$ are the number of vertices in the two components of $\Delta \setminus \sigma_m$. The second base case ensures that each vertex and edge of $\Delta$ appears in at most $O(1)$ subproblems at any level of the recursion tree. Thus, the total work at any level of recursion is $O(n \log n)$. The recursion tree has depth at most $O(\log k)$, so the overall algorithm runs in $O(n \log n \log k)$ time.

If we use a linear-time shortest-path algorithm instead of Dijkstra, the running time improves to $O(n \log k)$. This improvement was first described by Greg Frederickson in 1987, as one of the earliest applications of $r$-divisions.

\[1\] Without the second base case, it is possible for a constant fraction of the vertices to appear in a constant fraction of the recursive subproblems, leading to a running time of $O(kn \log n)$.
16.5 Sketch of FR-Dijkstra improvement

Frederickson held the record for fastest planar minimum-cut algorithm for almost two and a half decades; the record was finally broken in 2011 by two independent pairs of researchers, who ultimately published their result jointly: Giuseppe Italiano, Yahav Nussbaum, Piotr Sankowski, and Christian Wulff-Nilsen. Their $O(n \log \log n)$-time algorithm relies on an improvement to Dijkstra’s algorithm in dense distance graphs, proposed by Jittat Fakcharoenphol and Satish Rao in 2001, and now usually called FR-Dijkstra.

Recall from our earlier lecture on shortest paths that we can compute a dense distance graph for a nice $r$-division in $O(n \log r)$ time. The dense distance graph has $O(n/\sqrt{r})$ vertices—the boundary vertices of the pieces of the $r$-division—and $O(n)$ edges. So Dijkstra’s algorithm with a Fibonacci heap runs in $O(E + V \log V) = O(n + (n/\sqrt{r}) \log n)$ time.

FR-Dijkstra removes the $O(n)$ term from this running time. Specifically, within each piece of the $r$-division, the algorithm exploits the Monge structure in the boundary-to-boundary distances to avoid looking at every pair of boundary vertices. This is the same high-level strategy that we previously used with FR-Bellman-Ford, but with one significant difference: We do not know the relevant Monge arrays in advance. Instead, portions of each Monge array are revealed each time the Dijkstra wavefront touches the corresponding piece of the $r$-division.

I’ll discuss FR-Dijkstra in detail, along with the faster planar minimum-cut algorithm, in the next lecture.

16.6 Aptly Named Sir Not

- Global minimum cuts (dual to shortest weighted cycle)
- Deriving maximum flows from minimum cuts
- Minimum cuts in directed planar graphs (via shortest directed essential cycles)
- Maximum cuts (or minimum cuts with negative capacities)