CS 466/666 Algorithm Design and Analysis . Spring 2026

Lecture 2: Isolating Cuts

We study a new method to reduce the global minimum cut problem to the minimum s-t cut problem. This is based on an algorithm for the isolating cuts problem, which has found many applications recently.

Isolating Cuts

The minimum isolating cuts problem was introduced in [LP20] to solve the deterministic min-cut problem.

Input: A weighted undirected graph $G=(V,E,w)$ where each edge $e \in E$ has weight $w(e)$, and a subset of terminal vertices $R \subseteq V$.

Output: For each terminal vertex $v \in R$, find a cut $S_v \subseteq V$ with $v \in S_v$ and $R-v \subseteq V-S_v$ such that $w(\delta(S_v)) := \sum_{e \in \delta(S_v)} w_e$ is minimized.

(In words, for each v, find a minimum cut that isolates v from the vertices in R.)

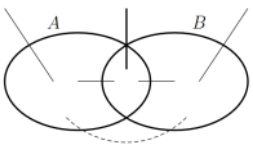
One nice property of the isolating cuts is that the cut $\{S_v\}_{v \in R}$ can be assumed to be vertex disjoint. This is a consequence of the proof, but can also be seen directly using the submodularity of cuts in undirected graphs (which will also be used in the proof).

Submodular function: A function $f: 2^V \to \mathbb{R}$ is submodular if for every pair A, B of subsets of V, it holds that $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$.

This is an important concept in combinatorial optimization and is useful in different areas. One well-known example is the cut function of undirected graphs.

Lemma: Let $G=(V,E,w)$ be an edge-weighted undirected graph. Let $d_w(S) := w(\delta(S))$. Then $d_w(A) + d_w(B) \geq d_w(A \cap B) + d_w(A \cup B)$ for all $A, B \subseteq V$.

Proof (by picture)

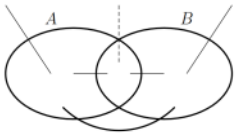


Check that each edge contributes to the LHS as much as to the RHS. □

For undirected graphs, the cut function d_w is symmetric, i.e. $d_w(S) = d_w(V-S)$. Applying the above lemma to the complements gives the following.

Lemma: Let $G=(V,E,w)$ be an edge-weighted undirected graph. Let $d_w(S) := w(\delta(S))$.

Then $d_w(A) + d_w(B) \geq d_w(A-B) + d_w(B-A)$ for all $A, B \subseteq V$.



One can also checks this directly by a picture.

Claim The minimum isolating cuts $\{S_v\}_{v \in R}$ can be assumed to be vertex disjoint.

Proof Let S_u and S_v be minimum isolating cuts for u and v, with $S_u \cap S_v \neq \phi$.

Note that $S_u - S_v$ and $S_v - S_u$ are also isolating cuts for u and v respectively.

Therefore, by the second lemma above, $d_w(S_u) + d_w(S_v) \geq d_w(S_u - S_v) + d_w(S_v - S_u) \geq d_w(S_u) + d_w(S_v)$.

So, equalities must hold throughout, and thus $S_u - S_v$ and $S_v - S_u$ are also minimum isolating cuts for u and v, and they are vertex disjoint.

Applying this "uncrossing" argument repeatedly will give pairwise vertex-disjoint minimum isolating cuts. □

This shows that the total description size of the minimum isolating cuts is $|V|$, and gives hope that there is a fast algorithm to compute the minimum isolating cuts.

The main result in this lecture is that we can compute minimum isolating cuts in $\log |V|$ max-flow calls.

Theorem (Isolating cuts) For a subset $R \subseteq V$, there is an algorithm that computes that computes the minimum isolating cuts for R using $\lceil \log_2 |R| \rceil + 1$ calls to min s-t cut on weighted graphs with $O(|V|)$ vertices and $O(|E|)$ edges, plus $\tilde{O}(|E|)$ deterministic time outside of the calls.

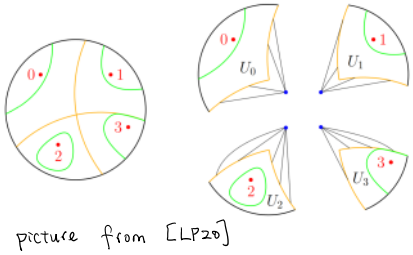
Algorithm

The first step of the algorithm is to compute vertex-disjoint sets $\{U_v\}_{v \in R}$ that contain $\{S_v\}_{v \in R}$

- Order the vertices in R from 0 to $|R|-1$, and label each vertex $i \in R$ by the binary string of i.
- For $0 \leq t \leq \lceil \log_2 |R| \rceil - 1$,

  Let $A_t \subseteq R$ be the vertices with the t-th bit being 0, and $B_t \subseteq R$ be those with t-th bit being 1.

  Compute a minimum A_t-B_t cut C_t (i.e. with $A_t \subseteq C_t$, $B_t \subseteq V - C_t$ and $w(\delta(C_t))$ minimized).
- For $v \in R$, let U_v be the component containing v in $G - \bigcup_t \delta(C_t)$, the graph obtained by removing all the edges in the union of $\delta(C_t)$.



picture from [LP20]

Each C_t can be computed using one min s-t cut call.

Observe that $\{U_v\}_{v \in R}$ are isolating cuts.

Claim $U_v \cap R = \{v\}$ $\forall v \in R$.

Proof Consider another vertex $u \in R$. Since $u \neq v$, their binary strings are differed in at least one bit, say t.

So, after removing the edges in $\delta(C_t)$, u and v are in different components in the remaining graph.

This applies to every $u \in R$, and thus v is not in the same component with any other $u \in R$. □

The key claim is that there is still an isolating cut S_v^* contained in U_v, by an uncrossing argument.

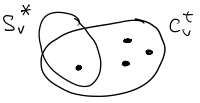
Claim. Let S_v^* be an inclusion-wise minimal minimum isolating cut of v. Then $S_v^* \subseteq U_v$.

Proof. In each iteration t, let $C_v^t = C_t$ if $v \in C_t$, and $C_v^t = V - C_t$ if $v \notin C_t$, i.e. C_v^t is the side of C_t containing v.

We will argue that $S_v^* \subseteq C_v^t$ for all t, and this implies that $S_v^* \subseteq \bigcap_t C_v^t$.

Note that S_v^* is connected, and so it is contained in U_v.

Suppose, by contradiction, that $S_v^* \not\subseteq C_v^t$



Note that $S_v^* \cap C_v^t$ is also an isolating cut.

Since S_v^* is an inclusion-wise minimal minimum isolating cut of v, $d_w(S_v^* \cap C_v^t) > d_w(S_v^*)$.

By submodularity, $d_w(S_v^*) + d_w(C_v^t) \geq d_w(S_v^* \cap C_v^t) + d_w(S_v^* \cup C_v^t) > d_w(S_v^*) + d_w(S_v^* \cup C_v^t)$,

This implies that $d_w(S_v^* \cup C_v^t) < d_w(C_v^t)$.

Note that $S_v^* \cup C_v^t$ is also a A_t-B_t cut as S_v^* does not contain any other vertex in R,

but this contradicts that C_v^t is a minimum A_t-B_t cut. □

Once we know that there is still a minimum isolating cut S_v^* contained in U_v, we can find S_v^* in a smaller graph where we contract all vertices in $V - U_v$ to a single vertex.

The second step of the algorithm is as follows.

- For each $v \in R$, construct the graph G_v by contracting all vertices in $V - U_v$ into a single vertex u.
- Return a minimum v-u cut in G_v as a minimum isolating cut of v in G.

For the correctness, any v-u cut in G_v is an isolating cut of v in G.

By the second claim above, a minimum v-u cut in G_v is a minimum isolating cut of v in G.

For the time complexity, note that $\sum_{v \in R} |E(G_v)| \leq 2|E(G)|$, as each edge appears in at most two small graphs

Therefore, the total time spent to compute a minimum v-u cut in G_v for all $v \in R$ is at most the time complexity to compute one min s-t cut in the original graph G.

Alternatively, we can also compute a minimum v-u cut in G_v for all $v \in R$ by setting up one max-flow computation in a "big" graph with $O(|V|)$ vertices and $O(|E|)$ edges. (Exercise.)

This completes the proof of the theorem.

Minimum Steiner Cut

Minimum Steiner Cut

We show a simple and interesting application of the minimum isolating cut algorithm to the minimum Steiner cut problem.

Input: A weighted undirected graph $G=(V,E,w)$ where each edge $e \in E$ has weight $w(e)$, and a subset of terminal vertices $R \subseteq V$.

Output: A cut $S \subseteq V$ with $S \cap R \neq \emptyset$ and $(V-S) \cap R \neq \emptyset$ such that $w(\delta(S))$ is minimized.

In words, a minimum cut that separates some terminal vertices.

Note that the global minimum cut problem is the special case when $R = V$.

If a minimum Steiner cut is an isolating cut, then the isolating cut algorithm will find it, the minimum Steiner cuts could contain many terminal vertices.

The idea is to do random sampling of the terminals.

Suppose a minimum Steiner cut S contains k vertices in R.

Let R' be a random subset of R where each vertex in R is in R' with probability $\frac{1}{k}$.

Then, we claim that the probability that S contains exactly 1 vertex in R' is a constant.

Indeed, this probability is $k(\frac{1}{k})(1-\frac{1}{k})^{k-1} \approx \frac{1}{e}$, and it can be boosted to $1-\frac{1}{\text{poly}(n)}$ by repeating $O(\log n)$ times.

If this happens, then we can find this minimum Steiner cut by finding minimum isolating cuts of R'.

We do not know k in advance, but we can try every k that is a power of 2.

To summarize, the following is the randomized algorithm for the minimum Steiner cut problem.

- For $i = 1$ to $\lfloor \log_2 n \rfloor$
  - Let $k = 2^i$
  - Let R' be a random subset of R where each vertex in R is in R' with probability $\frac{1}{k}$.
  - Use the isolating cut algorithm to find the minimum isolating cuts of R' in G.
- Return the minimum cut that we have found so far

By slightly modifying the calculation about the probability, we see that the success probability of one execution of this algorithm is at least a constant.

By running $O(\log n)$ independent executions, the success probability is at least $1 - \frac{1}{\text{poly}(n)}$.

Theorem There is a randomized algorithm that finds a minimum Steiner cut with high probability.

using $O(\log^3 n)$ max-flow computations plus $\tilde{O}(m)$ processing time.

This simple algorithm is a significant improvement over previous work on the minimum Steiner cut problem.

This can be derandomized to obtain a deterministic algorithm with similar running time [LP20].

Very recently, there is an almost linear-time algorithm for computing minimum s-t cut [CKLPPS22], and this implies an almost linear-time algorithm for the minimum Steiner cut problem as well.

Concluding remarks

Karger's near linear-time algorithm is very nice, but it is more ad-hoc and does not seem to generalize.

This new approach using isolating cuts works directly for the weighted setting, and recent works showed that it can be adapted to compute all-pairs min-cuts, vertex connectivity, hypergraphs, symmetric submodular function minimization, and so on.

And deterministic algorithms as well.

An interesting open question is to design a fast algorithm for global minimum-cut in directed graphs.

References

- [LP20] Li and Panigrahi, Deterministic min-cut in poly-logarithmic max-flows
- [CKLPPS22] Chen, Kyng, Liu, Peng, Probst Gutenberg, Sachdeva, Maximum flow and minimum-cost flow in almost-linear time.
- See also a TCS+ talk by Thatchaphol Saranurak about all-pairs minimum-cuts : a tutorial.