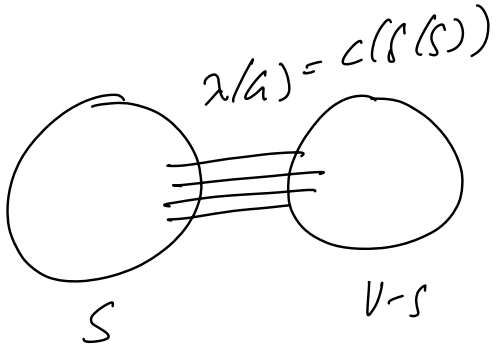
CS 598CSC: Topics in Graph Algorithms	Lecture date: 08/29/2024
Instructor: Chandra Chekuri	Scribe: CC

1 Global Mincut

Given a graph $G = (V, E)$ with edge capacities $c : E \to \mathbb{R}_+$, the global mincut (or simply mincut) value of G is defined as $\min_{S \subset V, S \neq \emptyset} c(\delta(S))$ where $\delta(S)$ is the set of edges with exactly one endpoint in S. For unweighted multi-graphs, this is also referred to as the edge connectivity and is often denoted by $\lambda(G)$.



It is the minimum number of edges whose deletion partitions the graph into two non-empty components. Mincut plays an important role in several applications, in particular it arises in LP relaxation for TSP. In addition, the structural and algorithmic understanding of mincuts plays an indirect role in several other problems.

How does one compute the mincut? A simple and standard way was to compute it was via $n-1$ s-t mincut computations; fix s arbitrarily and compute s-v mincut for each $v \in V - s$. Fairly recent work shows how one can do it with only poly-log maxflow computations — we will see this in the next lecture. Over the years, several (very) different algorithmic approaches have been developed for the mincut problem. One of the surprising ones is due to the work of Nagamochi and Ibaraki based on MA-orderings [NI92] who gave a combinatorial $O(mn+n^2 \log n)$-time algorithm that does not rely on flow at all; their approach generalizes to symmetric submodular functions. Hao and Orlin developed an approach to combine several flow computations together via the push-relabel method [HO94] (their approach also works for directed graphs). Karger developed elegant and powerful random contraction based algorithms for global mincut [Kar95], leading to many results. Two notable consequences are.

Theorem 1 (Karger and Stein [KS96])**.** *There is a randomized algorithm that runs in $O(n^2 \log n)$ time and outputs the mincut with high probability.*

Note that the algorithm is a Montecarlo one which means that we cannot be guaranteed that the mincut found is the correct one. Karger proved the following as a consequence of his contraction algorithm (see a later section for formal definitions and discussion).

Theorem 2 (Approximate Mincuts). *The number of α-approximate mincuts in a graph is at most $O(n^{2\alpha})$.*

Karger then developed another approach, via tree packing, to obtain a randomized near-linear time algorithm for mincut. He also was able to refine the bound on approximate mincuts via this approach.

Theorem 3 ([Kar00]). *There is a randomized algorithm that runs in time $O(m \log^3 n)$ and outputs the mincut with high probability.*

While the random contraction based algorithm is taught quite frequently due to its elegance and simplicity, the tree packing approach is more technical. More recently the tree packing approach has led to several new results. We will discuss the tree packing approach in this lecture. Williamson's book on network flows [Wil19] has a chapter on global mincut algorithms though it does not cover tree packing based approach

2 Tree packing based algorithm for mincut

Let $\tau(G)$ be the fractional tree packing value. We have seen that $\lambda(G)/2 \leq \tau(G) \leq \lambda(G)$. Suppose we can compute $\tau(G)$ exactly or approximately. It seems that it only gives us a 2-approximation to $\lambda(G)$. But we will see that it can provide more. A simple but crucial definition is the following.

Definition 1. *Let $T = (V, E_T)$ be a spanning tree of $G = (V, E)$. For an integer $k \geq 1$ we say T k-respects a cut $\delta(S), S \subset V$ if $|E(T) \cap \delta(S)| \leq k$.*

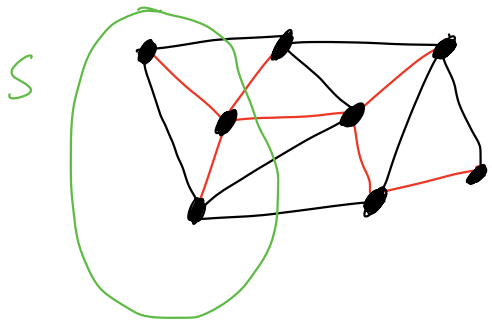


Figure 1: Spanning tree shown in red edges. T 2-respects the cut $(S, V - S)$.

Consider a mincut $(S, V - S)$ and an optimum tree packing given by $y^*_T, T \in \mathcal{T}$ where $\mathcal{T}$ is the set of spanning trees of G. Since the total value of the tree packing is at least $\lambda(G)/2 = c(\delta(S))/2$, not too many trees in the packing can cross S more than twice. We formalize this in the following technical lemma which is stated in a slightly more general form so that we can also work with approximate tree packings. We will use $p(T) = \frac{y_T}{\sum_T y_T}$ to deal with fractional packings.

Lemma 4. *Let $\delta(S)$ be a mincut of G and consider a $(1-\epsilon)$-approximate tree packing of G given by $y_T, T \in \mathcal{T}$. Let $p(T) = \frac{y_T}{\sum_T y_T}$ and let $\ell(T) = |E_T \cap \delta(S)|$ be the number of edges of T that cross the cut S. Let $q = \sum_{T:\ell(T)\leq 2} p_T$ be the fraction of the tree packing that 2-respects S. Then*

$$q \geq \frac{1}{2}(3 - \frac{2}{1-\epsilon}(1-1/n)).$$

In particular, if $\epsilon = 0$ then $q \geq \frac{1}{2} + \frac{1}{n}$ and if $\epsilon < 1/5$ then $q \geq 1/4$.

Proof. We have $\sum_T y_T \geq (1-\epsilon)\tau_{\text{frac}}(G)$ by assumption. We have seen that $\tau_{\text{frac}}(G) \geq \frac{n}{n-1}\lambda(G)/2$. Putting together we have

$$\sum_T y_T \geq (1-\epsilon)\frac{n}{n-1}\lambda(G)/2.$$

Let $S \subset V$ be a mincut. We have $1 = \sum_T p(T) = \sum_{T:\ell(T)\leq 2} p_T + \sum_{T:\ell(T)\geq 3} p_T$. Each tree T with $\ell(T) \geq 3$ uses up at least 3 edges from $\delta(S)$ and each tree T with $\ell(T) \leq 2$ uses up at least 1 edge from $\delta(S)$. Since the total capacity of $\delta(S)$ is $\lambda(G)$ and the tree packing is valid,

$$(\sum_T y_T)(q + 3(1-q)) \leq \lambda(G).$$

Combining with $\sum_T y_T \geq (1-\epsilon)\frac{n}{n-1}\lambda(G)/2$, we obtain

$$(q + 3(1-q)) \leq 2(1-1/n)\frac{1}{1-\epsilon}.$$

Simplifying yields the desired claim. ■

In other words, if the tree packing is sufficiently good approximation then a constant fraction of the trees in the packing will cross the mincut at most twice.

Exercise 1. *For any mincut $(S, V-S)$ and any exact tree packing, there is a tree T in the support of the packing such that T is 1-respecting with respect to S.*

2.1 Algorithm for mincut

Karger's original algorithm was more involved because there was no near-linear time approximation algorithm for tree packing that he could use as a black-box at that time. He used a form of sparsification and then applied an approximate tree packing algorithm on the sparsified graph which is quite a feat. In our description we will use the algorithm for approximate tree packing from [CQ17] as a black-box which simplifies the description.

1. Given $G = (V, E)$ with capacities $c : E \to \mathbb{R}_+$, compute a $(1-\epsilon_0)$-approximate tree packing for some fixed $\epsilon_0 < 1/5$. Let $y_T, T \in \mathcal{T}$ be the packing.
2. Pick a tree T at random from the packing where the probability of picking T is $p_T = y_T / \sum_T y_T$.
3. Find the cheapest cut $(S, V-S)$ in G such that T is 2-respecting with respect to $\delta(S)$.

The following is easy to see from Lemma 4.

Lemma 5. *The algorithm outputs the mincut of G with probability at least $1/4$.*

We can repeat the last two steps $\Theta(\log n)$-times to increase the probability of correctness to $(1 - 1/n^c)$ for any desired constant c. Now we analyze the running time. A key ingredient is Step 3. Karger showed that one can implement the step via a clever dynamic program coupled with link-cut tree data structure. It is worth reading.

Theorem 6 ([Kar00])**.** *Given a graph G and a spanning tree T, there is a deterministic algorithm that computes a minimum 2-respecting cut with respect to T in $O(m \log^2 n)$ time.*

As we mentioned in the previous lecture, [CQ17] gave a deterministic algorithm to compute a $(1 - \epsilon)$-approximate tree packing in $O(\frac{1}{\epsilon^2} m \log^3 n)$-time. We fix ϵ to be a constant such as $1/5$, and hence the time for this step is $O(m \log^3 n)$. We only need to compute the packing once and apply the repetition for the second and third steps to boost the probability of success. Thus, if we do $O(\log n)$ repetitions, we obtain an $O(m \log^3 n)$-time algorithm that outputs the correct mincut with high probability.

3 Bounding the number of approximate mincuts

How many distinct mincuts can an undirected graph have? The following theorem was first shown by Dinitz and Karzanov, and Lomonosov [DKL76].

Theorem 7. *The number of distinct mincuts in an undirected graph is at most $\binom{n}{2}$.*

An n-cycle is the worst example with $\binom{n}{2}$ cuts. All the mincuts of a graph can be represented in a nice compact data structure called the *cactus* as was also shown in [DKL76]. See [FF09] for a short proof. See [Cun83] for a deeper result on decomposition submodular functions. The next two figures are taken from [FF09].

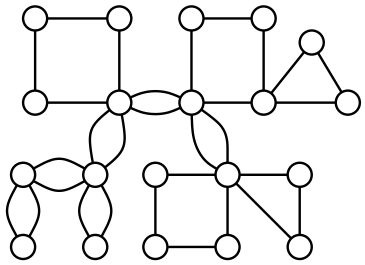


Figure 2: A cactus is a graph in which each edge is a single cycle (allowing for 2-edge cycles). Alternatively, it is a 2-edge-connected graph in which each block consists of a single cycle. This figure is from [FF09].

In contrast, the number of s-t mincuts can be very large, potentially exponential in n. We will be interested in approximate mincuts.

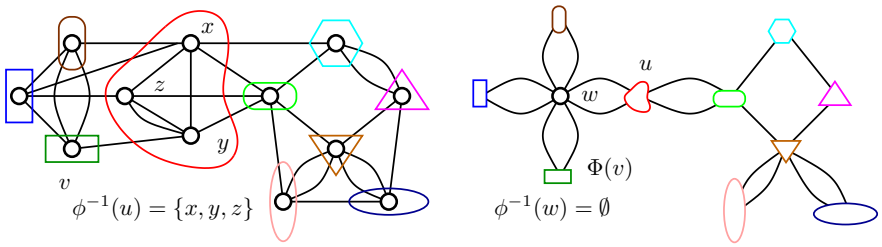


Figure 3: A graph and the cactus representation of its mincuts. This figure is from [FF09].

Definition 2 (α-approximate mincut)**.** *For $\alpha \geq 1$, a cut $S \subset V$ is an α-approximate mincut if $c(\delta(S)) \leq \alpha\lambda(G)$.*

Karger used tree packing to prove the following theorem.

Theorem 8 ([Kar00])**.** *The number of α-approximate mincuts in a graph is $O(n^{\lfloor 2\alpha \rfloor})$.*

We prove the preceding theorem via the description in [CQX20]. We will work with an optimum fractional tree packing $\{(T_i, y_i)\}$. Recall that an optimum fractional tree packing can be found via a solution to a linear program which has m non-trivial constraints, so we can assume that the support size of y (that is, the number of i such that $y_i > 0$) is at most m.

Consider an α-approximate cut $S \subset V$. Fix $h = \lceil \frac{2}{\alpha} \rceil$. Let $q_{h,\alpha}$ be the fraction of tree packing that h-respects $S \subset V$. Using a similar analysis as the one in Lemma 4, we can argue that

$$q_{h,\alpha} \geq \frac{1}{h}(1 - (2\alpha - \lfloor 2\alpha \rfloor))(1 - 1/n).$$

Moreover, at least one tree in the packing h-respects the cut. If we use this latter fact, then the total number of α-approximate cuts is at most $m \cdot n^h \leq m \cdot n^{\lfloor 2\alpha \rfloor}$. We can do better by noticing that $q_{h,\alpha} > 0$ is a fixed constant for any fixed α. Suppose N is the number of α-approximate mincuts. For any fixed α-approximate mincut, $q_{h,\alpha}$ fraction of the tree packing is h-respecting with respect to the cut. Fix a single tree. How many distinct h-respecting cuts can we obtain from T? We remove at most h edges from T to create at most $h + 1$ components. We can combine these components into two sides of a cut. Hence, each tree T can correspond to at most $2^{h+1}\binom{n}{n} \leq 2^{h+1}n^h$ cuts.

We thus claim that the number of α-approximate cuts is at most $2^{h+1}n^h/q_{h,\alpha}$. Why? It is easier to think of the cut packing as consisting of N distinct trees (for N very large) where each tree T, $p(T) = 1/N$. As we argued each tree gives rise to at most $2^{h+1}n^h$ cuts at most. For each distinct α-approximate cut S, there are at least $q_{h,\alpha}N$ of the N trees are h-respecting with respect to it. Thus, if L is the number of distinct α-approximate cuts then $L \cdot q_{h,\alpha}N \leq 2^{h+1}n^h N$ which implies that $L \leq 2^{h+1}n^h/q_{h,\alpha}$, as desired.

See [BCW23] for a very different approach to bounding the number of approximate mincuts. The bound they achieve is $n^{4\alpha+2}$ which is weaker that what we proved, but the proof technique is different and elegant.

Deterministic algorithm: Karger's algorithm is a randomized near-linear time algorithm for global mincut. Designing a deterministic algorithm was a long-standing open problem. A very

recent algorithm obtained such an algorithm [HLRW24] (see the paper for the history and related work that preceded it). The precise running time is not stated since there are many log factors. Moreover, it appears that the running time depends on $\log W$ where W is the largest integer capacity. This is in contrast to the randomized algorithm of Karger which is strongly polynomial. The algorithm is quite involved. Is there a simple deterministic near-linear time algorithm for the global mincut?

References

[BCW23] Calvin Beideman, Karthekeyan Chandrasekaran, and Weihang Wang. Approximate minimum cuts and their enumeration. In *Symposium on Simplicity in Algorithms (SOSA)*, pages 36–41. SIAM, 2023.

[CQ17] Chandra Chekuri and Kent Quanrud. Near-linear time approximation schemes for some implicit fractional packing problems. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 801–820. SIAM, 2017.

[CQX20] Chandra Chekuri, Kent Quanrud, and Chao Xu. Lp relaxation and tree packing for minimum k-cut. *SIAM Journal on Discrete Mathematics*, 34(2):1334–1353, 2020.

[Cun83] William H Cunningham. Decomposition of submodular functions. *Combinatorica*, 3(1):53–68, 1983.

[DKL76] Efim A Dinitz, Alexander V Karzanov, and Michael V Lomonosov. On the structure of the system of minimum edge cuts of a graph. *Issledovaniya po Diskretnoi Optimizatsii*, pages 290–306, 1976.

[FF09] Tamás Fleiner and András Frank. A quick proof for the cactus representation of mincuts. *EGRES Quick Proof*, 2009. `https://egres.elte.hu/qp/egresqp-09-03.pdf`.

[HLRW24] Monika Henzinger, Jason Li, Satish Rao, and Di Wang. Deterministic near-linear time minimum cut in weighted graphs. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3089–3139. SIAM, 2024.

[HO94] JX Hao and James B Orlin. A faster algorithm for finding the minimum cut in a directed graph. *Journal of Algorithms*, 17(3):424–446, 1994.

[Kar95] David Ron Karger. *Random sampling in graph optimization problems*. PhD thesis, stanford university, 1995.

[Kar00] David R Karger. Minimum cuts in near-linear time. *Journal of the ACM (JACM)*, 47(1):46–76, 2000.

[KS96] David R Karger and Clifford Stein. A new approach to the minimum cut problem. *Journal of the ACM (JACM)*, 43(4):601–640, 1996.

[NI92] Hiroshi Nagamochi and Toshihide Ibaraki. Computing edge-connectivity in multigraphs and capacitated graphs. *SIAM Journal on Discrete Mathematics*, 5(1):54–66, 1992.

[Wil19] David P Williamson. *Network flow algorithms*. Cambridge University Press, 2019.