MATH 8500 Algorithmic Graph Theory, Spring 2017, OSU
Lecture 1: Min-Cut and k-Cut
Instructor: Anastasios Sidiropoulos
Scribe: Ario Salmasi

1 The Min-Cut problem

s-t Min-cut
Input: A graph $G = (V, E)$, $s, t \in V$, and a function $w : E \to \mathbb{R}_{\geq 0}$.
Question: Find $E' \subseteq E$, such that s and t lie in different connected components of $G \setminus E'$, minimizing $w(E')$.

One way to solve s-t Min-cut problem is to use Max-flows. Another important partitioning problem is called Min-cut, and is defined as follows.

Min-cut
Input: A graph $G = (V, E)$, and a function $w : E \to \mathbb{R}_{\geq 0}$.
Question: Find $E' \subseteq E$, such that $G \setminus E'$ has at least two connected components, minimizing $w(E')$.

Here is an application of algorithms for partitioning graphs. Suppose that you are given an image that has an object and a background. The goal is to distinguish the object from the background. One can think of the image as follows. You have a vertex for every pixel of the image and there is an edge between every two neighboring pixels. The weight of each edge (u, v) is the likelihood of u and v being in the same component (either the object or the background). The goal is to find a set of edges with minimum weight, partitioning object from the background.

Suppose that you are given an algorithm for the s-t Min-cut problem. One can use this algorithm to solve the Min-cut problem as follows. Find the s-t Min-cut for all pairs of vertices and return the minimum value. On the other hand, if you are given an algorithm for Min-cut, it is not clear how to get an algorithm for s-t Min-cut. We are going to discuss a randomized algorithm for Min-cut problem.

Algorithm 1 Karger's Algorithm for Min-cut

Require: An undirected unweighted connected graph G.
Ensure: A Min-cut of G.

1 While there exists at least 3 vertices,

  1.1 Pick an edge uniformly at random and contract it.

2 Return the remaining edges.

We need to define the contraction step. Given an edge $e = (u, v)$ in a multigraph, we *contract* e as follows. We delete all the edges between u and v, replace u and v with a new vertex uv, and replace all the neighbor edges of u and v with edges incident to uv (See Figure 1).

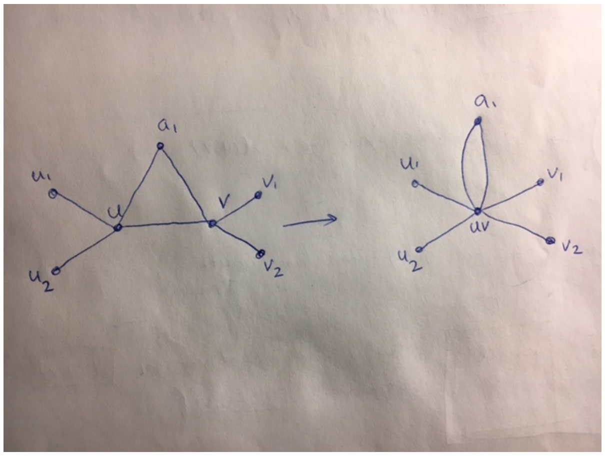

Figure 1: Contracting an edge (u, v)

Note that this algorithm can be extended to the weighted case, but you have to take edges with probability proportion to their weight.

Claim 1. *In any graph G, the size of the Min-cut is at most the minimum degree.*

Proof. Let v be the vertex of minimum degree. By deleting all the edges incident to v, we get a cut for G, and thus the size of the Min-cut should be at most the degree of v. □

Claim 2. *If the size of Min-cut is k, then we have $|E| \geq nk/2$.*

Proof. Since the size of Min-cut is k, every vertex must have degree at least k. Therefore, we have $|E| = \sum_{v \in V} \deg(v)/2 \geq \sum_{v \in V} k/2 = nk/2$. □

Now let C be some Min-cut in G. Let A be the event that the algorithm outputs C. For every $i \in \{1, 2, \ldots, n-2\}$, let A_i be the event that in iteration i, the algorithm does not pick an edge in C.

Claim 3. *We have* $\Pr(A_1) \geq (n-2)/n$.

Proof. We have that $\Pr(A_1) = 1 - \Pr(\neg(A_1)) = 1 - |C|/|E| \geq 1 - 2/n = (n-2)/n$, as desired. □

Claim 4. *For every $i \in \{2, 3, \ldots, n-2\}$, we have* $\Pr(A_i | \cap_{j=1}^{i-1} A_j) \geq (n-i-1)/n-i+1$.

Proof. Since we contract an edge in every iteration of the algorithm, at iteration i, there are $n-i+1$ vertices left. Assuming $\cap_{j=1}^{i-1} A_j$, we have that the minimum cut in the current graph is the same as the original graph. This is because we have not picked any of the edges in C in the first $i-1$ iterations. Therefore, the min degree in the contracted graph is at least $|C|$ (counting parallel edges with multiplicities), and thus there are at least $|C|/2.(n-i+1)$ edges left. Therefore we have:

$$\begin{aligned} \Pr(A_i | \cap_{j=1}^{i-1} A_j) &= 1 - \Pr(\neg(A_i | \cap_{j=1}^{i-1} A_j)) \\ &\geq 1 - 2/(n-i+1) \\ &= (n-i-1)/n-i+1. \end{aligned}$$

□

Theorem 1. *The algorithm succeeds with probability at least* $2/n(n-1)$.

Proof. We have that $\Pr(A) = \Pr(\neg(A_2 | \cap_{j=1}^{n-3} A_j)). \Pr(\neg(A_3 | \cap_{j=1}^{n-2} A_j)). \ldots \Pr(A_1)$. Therefore we have

$$\begin{aligned}\Pr(A) &\geq \frac{n-(n-2)-1}{n-(n-2)+1} \times \frac{n-(n-3)-1}{n-(n-3)+1} \times \ldots \times \frac{n-2}{n} \\ &= \frac{1}{3} \times \frac{2}{4} \times \ldots \times \frac{n-2}{n} \\ &= \frac{2}{n(n-1)}.\end{aligned}$$

□

Claim 5. *If we run the algorithm* k *times, at least one execution succeeds with probability at least* $1 - (\frac{2}{n(n-1))})^k$.

Claim 6. *For every* $x \geq 1$, *we have that* $(1 - \frac{1}{x})^x \leq 1/e$.

Now let $k = \frac{n(n-1)}{2} \cdot \ln(n)$. Therefore we have that the failure probability is at most $(\frac{2}{n(n-1))})^k \leq (\frac{1}{e})^{\ln(n)} \leq 1/n$. Therefore by running the algorithm $k = \frac{n(n-1)}{2} \cdot \ln(n)$ times, the failure probability will be less than $1/n$. If we run the algorithm $k = 10 \cdot \frac{n(n-1)}{2} \cdot \ln(n)$, the failure probability will be less than $1/(n^{10})$.

2 The k-Cut problem

k-Cut

Input: A connected unweighted graph $G = (V, E)$, and an integer k.

Question: Find $E' \subseteq E$, such that $G \setminus E'$ has at least k connected components, minimizing $|E'|$.

A similar algorithm applies for this problem.

Algorithm 2 Karger's Algorithm for k-cut

Require: An undirected unweighted connected graph G, and an integer k.

Ensure: A k-cut of G.

1 While there exists at least $k+1$ vertices,

  1.1 Pick edge uniformly at random and contract it.

2 Return the remaining edges.

Theorem 2. *The success probability of Karger's algorithm for k-Cut is at least* $\frac{1}{n^{2k}}$.

Corollary 1. *For any fixed* $k \geq 2$, *k-Cut can be solved in randomized polynomial time. i.e. there exists an algorithm with running time* $n^{O(k)}$ *(which is polynomial for fixed* k*), that succeeds with high probability.*

Theorem 3. *k-Cut is NP-hard.*