3.6 Min-cut and network reliability

3.6.1 Min-Cut: Karger’s algorithm

We consider undirected simple (no loops or multiple edges) weighted graphs \(G\) on \(n\) vertices.

\[ w(i, j) \geq 0, \quad w(i, j) = w(j, i), \quad i \neq j \in \{1, \ldots, n\} \]

Write \(w(G) = \sum_{k<l} w(k, \ell)\).

A cut is a partition of the vertices into two non-empty subsets \(S, \bar{S}\); the weight of cut \((S, \bar{S})\) is

\[ w(S, \bar{S}) = \sum_{i \in S, j \in \bar{S}} w(j, i) \]

A min-cut is a cut of minimum weight over all possible cuts; the min-cut problem is that of computing the value of the min-cut. Usually we also mean that we want to output some cut of this value. This doesn’t make the problem much harder: if you know the min-cut value is \(c\), try removing some edge \(e\), of weight say \(w\). If the remaining graph has a cut of weight \(c - w\), then you can safely put this edge in the cut, and if the remaining graph has only cuts of weight \(> c - w\), then you can safely exclude the edge from the cut. Either way, you can just check the next edge, in the first case with the graph \(G - e\), in the second case with the graph \(G\). (This is known as a “self-reducibility” argument. We saw this idea last term when we were studying algorithms for perfect matching.)

Today: Randomized poly-time algorithm for min-cut. (Deterministic poly-time is known but is much more complicated.) Then we’ll combine these ideas (slightly extended) with the \#DNF approximation algorithm (also slightly extended), to give a FPRAS (will be defined below) for the network reliability problem.

Contrast: The max-cut problem is NP-complete.

**Definition 31.** Let \(\{i, j\}\) be an edge of \(G\). In the contraction of \(G\) by \(\{i, j\}\), \(G/\{i, j\}\), the vertices \(i\) and \(j\) are replaced by a single new vertex \((i, j)\), and for each \(v \notin \{i, j\}\) any edges \(\{i, v\}\) or \(\{j, v\}\) are replaced by the edge \(\{(i, j), v\}\), with the sum of the constituent weights; the edge \(\{i, j\}\) is removed; the rest of the graph remains unchanged.

With each contraction, the number of vertices of \(G\) decreases by one. There is a \(1 - 1\) correspondence between cuts of \(G\) that don’t separate \(i\) and \(j\), and cuts of \(G/\{i, j\}\). In particular, every cut in the graph \(G/\{i, j\}\) is a cut in \(G\). So min-cut(\(G/\{i, j\}\)) \(\geq\) min-cut(\(G\)).

Let \(c\) be the value of a min-cut of \(G\). In particular, the edges incident on any vertex of \(G\) sum to at least \(c\). This remains true of every vertex of \(H_t\) (because the min-cut is nondecreasing, as just noted), so

\[ w(H_t) \geq \frac{(n - t)c}{2} \quad \text{ (3.22)} \]

(the factor of two for counting weights from both ends).

**Theorem 32.** Let \((S, \bar{S})\) be a min-cut. The probability that Karger’s algorithm outputs a refinement of \((S, \bar{S})\) is at least \(\frac{\binom{n'}{t'}}{(n-t')^2}\)
CHAPTER 3. WEIGHTED SAMPLING

Algorithm 4: Karger’s Min Cut Algorithm \cite{25}

Input: Undirected simple graph $G$ with edge-weights $w$; desired size $n' \geq 2$.

Output: An $n'$-way cut $H_0 := G$

$t := 0$

while $t < n - n'$ do

Pick $(i, j) \in E(H_t)$ with probability $w(i, j)/w(H_t)$

$H_{t+1} := H_t / \{i, j\}$

$t ← t + 1$

Return the cut of $G$ corresponding to the $n' = n - t$ vertices in $H_t$.

Exercise: For $n' = 2$ this is tight.

Proof. As noted, $w(S, \bar{S}) = c$. $(S, \bar{S})$ is output by the algorithm if and only if none of the edges crossing this cut is contracted by the algorithm in its $n - 2$ iterations. Suppose that none of the edges in $(S, \bar{S})$ was contracted in $H_{t-1}$. Then

$$\Pr(\text{an edge of } (S, \bar{S}) \text{ is contracted in } H_t) = \frac{c}{w(H_{t-1})} \leq \frac{2}{n - t + 1}$$

where in the inequality we have applied (3.22). Therefore, for the output of the algorithm,

$$\Pr[\text{min-cut } (S, \bar{S}) \text{ coarsens the output } H_{n-n'}] = (1 - \frac{c}{w(H_0)}) \cdot \cdot \cdot (1 - \frac{c}{w(H_{n-(n'-1)})})$$

$$\geq (1 - \frac{2}{n}) \cdot \cdot \cdot (1 - \frac{2}{n'})$$

$$= \frac{n - 2}{n} \cdot \frac{n - 3}{n - 1} \cdot \cdot \cdot \frac{n' - 1}{n' + 1}$$

$$= \frac{n'(n' - 1)}{n(n - 1)}. \quad (3.23)$$

If this is run to completion ($n' = 2$), the bound is $\frac{1}{(\frac{1}{2})}$. This is therefore a lower bound on the probability of success, even if there is only one min-cut.

Corollary 33. Repeating Karger’s algorithm $O(n^2)$ times gives a probability bounded away from 0 of correctly outputting the min-cut value, and repeating the algorithm $O(n^2 \log n)$ times gives a probability bounded away from 0 that we observe all min cuts. (Which is even more than we required for this problem – but we’ll want the stronger property a little later.)

The second part of the corollary comes from the well-known:

**Coupon collector’s problem:** sample with repetition from $k$ kinds of coupons. How many trials until all kinds have been seen? (Was assigned as an exercise)

| 1 | 2 | 3 | … | k |

If all probabilities are $\frac{1}{k}$, then the expected number of trials is $\Theta(k \log k)$.

If all probabilities are at least $p$, then the expected number of trials is $O\left(\frac{1}{p} \log \frac{1}{p}\right)$.

It will be a homework problem that each contraction step can be implemented in time $O(n)$, and therefore that one trial of the algorithm runs in time $O(n^2)$. Consequently the time to success with constant probability is $O(n^4)$. Now let’s see a faster method.

Caltech CS150b 2019. 38