3.7 Network reliability problem

Given a simple (no loops or multiple edges) undirected, unweighted, connected graph G. (Actually, it's fine if the graph is not connected, but the problem is trivial then.) Each edge fails independently with probability p. (I.e., we do bond percolation on G, with parameter $1-p$.) Define

$$\mathrm{FAIL}(p) = \Pr[G \text{ is disconnected}].$$

Our problem is to compute FAIL(p). This is a #P-complete problem. So we aim for a multiplicative approximation.

In the regime that the network G is fairly unreliable, specifically, $\mathrm{FAIL}(p) \in n^{-O(1)}$, then this problem is easy: just simulate and test. (That will indeed be the "easy case" of the algorithm we present below.) The challenging case is when the network is actually pretty reliable. Then we are trying to estimate the probability of a rare failure event. This is, throughout engineering, a difficult problem.

For network reliability, there is an elegant FPRAS (recall definition 29) due to Karger [29], [2] which we'll now see.

In the first step of the algorithm, we compute the value of the min-cut of G and denote it c. Note that FAIL(p) is at least p^c.

Case 1: $p^c \geq n^{-4}$
Algorithm: Brute force Monte Carlo.
Repeatedly toss independent, bias-p coins to simulate edge failures, and check if graph is connected. With $\mathrm{poly}(n, 1/\varepsilon, \log 1/\delta)$ samples get an estimate that with prob. $\geq 1-\delta$ is in $\mathrm{FAIL}(p) \pm \varepsilon$.

Case 2: $p^c < n^{-4}$
Specifically set $\delta \geq 2$ to satisfy $p^c = n^{-2-\delta}$.

For $\alpha \geq 1$, call the cuts of size at most αc "α-light", and the remaining α-heavy. We restrict ourselves to α integer or half-integer.

We bound the contributions to FAIL(p) from 3-light cuts and 3-heavy cuts separately.

Lemma 35. *The number of α-light cuts is not greater than $n^{2\alpha}$ and there is a poly-time algorithm to list them all.*[3]

This is quite interesting, as it holds not because there are some obviously few "degrees of freedom".

Note, the combinatorial bound is essentially tight (again on the cycle).

Proof. We use an argument similar to before. Use the "abbreviated" form of the min-cut algorithm in which we run the loop only until the graph is left with $k = 2\alpha$ vertices. Then list all 2^{k-1} coarsenings of this multi-way cut. If $(S, \bar{S})$ is a "light" cut, then the probability that it survives as a coarsening is at least

$$\prod_{t=0}^{n-k-1} \left(1 - \frac{2\alpha c}{(n-t)c}\right) = \prod_{t=0}^{n-k-1} \frac{n-2\alpha-t}{n-t} \geq \frac{1}{\binom{n}{k}}.$$

Consequently there are no more than $\binom{n}{k} 2^{k-1} \leq n^{2\alpha}$ light cuts. If we run this process $n^{2\alpha} 2\alpha \log n$ times, we are highly likely (coupon collector) to have a complete listing of all light cuts. □

Lemma 36. *The sum of the probabilities of the failures of all "heavy" cuts is $o(p^c)$.*

[2]And see [30] for an in some ways improved algorithm.
Also, recently, Guo and Jerrum [23] obtained a FPRAS for the quantity $1 - \mathrm{FAIL}(p)$. That is, they can multiplicatively approximate the probability that the network is connected, even when that probability is very small.
[3]Bound recently slightly improved by Gupta, Lee and Li [24].

Therefore, we can ignore all of the "heavy" cuts, i.e., we don't need to compute the contributions to FAIL(p) from "heavy" cuts.

Proof. List the cuts as $C_1, \ldots$ in order of nondecreasing weight, which we denote $|C_k|$. By Lemma 35, $|C_k| \geq \frac{c \log k}{2 \log n}$.

Consequently $p^{|C_k|} \leq p^{\frac{c \log k}{2 \log n}}$ and applying $p^c = n^{-2-\delta}$, i.e., $\log n = -\frac{c}{2+\delta} \log p$, gives $p^{|C_k|} = p^{-\frac{(2+\delta)\log k}{2 \log p}} = p^{-\frac{2+\delta}{2} \log_p k} = k^{-\frac{2+\delta}{2}}$. Therefore the total contribution from heavy cuts is upper bounded by $\sum_{k \geq n^{2\alpha}} k^{-\frac{2+\delta}{2}} = \frac{2}{\delta} n^{-(2\alpha)(\delta/2)} = \frac{2}{\delta} n^{-\alpha\delta} = p^c \frac{2}{\delta} n^{2-(\alpha-1)\delta}$. Since $\delta \geq 2$ it is sufficient to take, say, $\alpha = 3$ in order to guarantee that this is $o(p^c)$. $\square$

Now we return to the cuts that matter, namely the light cuts. How to compute Pr[some light cut fails]?

We can write a DNF and then use the Karp-Luby-Madras algorithm to evaluate the probability that it is satisfied under a random assignment to the input variables.

Let $X_{i,j}$ be the event that edge (i, j) fails. The algorithm lets us construct a poly-size DNF formula describing the event that some light cut fails.

$$\Phi(\bar{x}) = \bigvee_{\text{light cuts}(S,\bar{S})} \Big(\bigwedge_{\substack{i \in S \\ j \in \bar{S}}} x_{i,j} \Big).$$