

How easy is it to disconnect a network? This problem is formalized as the min-cut problem. It also arises as a subproblem in many optimization problems.

Input: Undirected graph $G = (v, E)$.

Output: Smallest set of edges whose removal separates the graph into at least two pieces.

Deterministic Algorithms: Fix a vertex as s and for all possible choices of other vertices as t , find the minimum $s - t$ cut. Clearly this must be the global min-cut; since the global min-cut must separate s from some t . Each st -min-cut takes time $O(n^3)$ for a total time of $O(n^4)$.

Randomized Algorithms: We will see a very simple randomized algorithm that returns the global min-cut with high probability in $O(n^4 \ln n)$ time. We will then refine it to get algorithms which run respectively in $O(n^3 \ln n)$ and $O(n^2 \ln^3 n)$ time. The last is a substantial improvement over the deterministic algorithm, and nearly optimal, since for a dense graph we would need $\Omega(n^2)$ time just to examine all edges.

This algorithm does not always return the correct answer, only most of the time. Such algorithms are called *Monte Carlo* algorithms. Algorithms that always return the correct answer but take a variable amount of time depending upon the random choices are in contrast called *Las Vegas* algorithms.

1 The basic algorithm

For the algorithm we treat G as a multigraph, i.e. allow parallel edges between the same pair of vertices. The core of the algorithm is as follows:

For $i = 1$ to $n - 2$

1. Pick a random edge (u, v) from the current graph.
2. Merge vertices u, v into a single vertex w , and remove the edge (u, v) . The vertex w will have edges to all erstwhile neighbours of u, v . Further, if u, v both had n_u, n_v edges to some vertex x , then w will have $n_u + n_v$ edges to x . If there were other edges connecting u to v , these would be removed from the graph. This operation will be called an edge contraction.

The edges that remain are returned, i.e. those edges in G from which these were derived.

1.1 Time Analysis:

It is an easy exercise to show that very simple data structures enable the algorithm to be executed in $O(n^2)$ time.

1.2 Probability of returning the correct answer

A key property of contractions is: A set of edges C is a cut after a contraction iff C constituted a cut before and the contracted edge did not belong to C . This is easily proved.

Let C denote the set of edges constituting a min-cut. The key observation is: if none of the edges in C are picked for contraction, then C will be returned. This is because C continues to remain a cut after every contraction. But after $n - 2$ contractions there are only 2 vertices, and hence just one cut. So it must be C . So we simply need to estimate the probability that C survives each contraction, i.e. no edge of C is picked.

Let E_i denote the event that the cut C survives the first i contractions. Let $k = |C|$.

Lemma 1 $\Pr[E_1] = 1 - \frac{2}{n}$. $\Pr[E_i|E_{i-1}] \geq 1 - \frac{2}{n-i+1} = \frac{n-i-1}{n-i+1}$.

Proof: . The first part is simply the special case with $i = 1$ of the second part, defining E_0 to be the event that survives 0 contractions, which happens with probability 1. Suppose C survives the first $i - 1$ contractions. Since no new cuts can arise as a result of a contraction, if C was a min-cut at the beginning, then it must remain a min-cut after $i = 1$ contractions.

But this implies that each vertex must have degree at least $k = |C|$; if not, cutting off that vertex from the rest of the graph would give a smaller cut. So after $i - 1$ contractions, there are $n - i + 1$ vertices of degree at least k , i.e. a total of $(n - i + 1)k/2$ edges.

C will survive if any of its k edges are not picked; this happens with probability at least $1 - \frac{k}{(n-i+1)k/2} = 1 - \frac{2}{n-i+1}$. ■

Lemma 2 *The probability that C survives j contractions is at least $\frac{(n-j-1)(n-j)}{n(n-1)} = \theta\left(\left(\frac{n-j}{n}\right)^2\right)$.*

Proof: $\Pr[E_j] = \Pr[E_j|E_{j-1}] \cdot \Pr[E_{j-1}] = \Pr[E_j|E_{j-1}] \cdot \dots \cdot \Pr[E_2|E_1] \cdot \Pr[E_1]$

But this last expression is simply at least:

$$\frac{n-j-1}{n-j+1} \cdot \frac{n-j}{n-j+2} \cdot \frac{n-j+1}{n-j+3} \cdots \frac{n-3}{n-1} \cdot \frac{n-2}{n} = \frac{(n-j-1)(n-j)}{n(n-1)}$$

Choosing $j = n - 2$ we get the probability that the core algorithm returns C with probability at least $2/n(n - 1)$.

1.3 The complete basic algorithm

We simply run the core $cn^2 \ln n$ times. The probability that the C fails to survive is at most $(1 - (\frac{2}{n(n-1)}))^{cn^2 \ln n} \leq e^{\frac{-2cn^2 \ln n}{n(n-1)}} \leq n^{-2c}$.

2 Improvement 1

It may be seen that not all contractions are equally safe for the cut C . C survives the first contraction with probability $1 - 2/n$, but survives the $n - 2$ th contraction with a probability $1/3$. So this suggests that we avoid the bad contractions and try something else.

One idea is to stop after j contractions after j and then use the deterministic algorithm. The probability that the cut survives j contractions, from the discussion above is at least

$\frac{(n-j-1)(n-j)}{n(n-1)} = \theta(((n-j)/n)^2)$. This is the probability of surviving the entire algorithm, since the next part is deterministic. The deterministic part will take time $O((n-j)^4)$.

Picking $j = n - \sqrt{n}$ gives the probability of survival as $\theta(1/n)$, and the time for the second part as $O(n^2)$.

Thus now we need repeat this new algorithm only $cn \ln n$ times and pick the best to give high probability of getting an optimal cut. The total time is thus $O(n^3 \ln n)$.

3 Improvement 2

Since we know that the initial contractions are more reliable, we should boost the reliability of later contractions by repeating those more frequently than the initial ones. So for example, we could do some z contractions, and then do the remaining ones twice. But perhaps we should repeat more frequently those that are further down? This motivates a recursive algorithm:

1. Perform z contractions to produce a graph G' . The number z will be specified later.
2. For $i = 1$ to some y to be chosen later compute $C_i = \text{cut}$ obtained by solving G' .
Notice that we expect to get different C_i s because the random choices can be different.
3. Return smallest of all C_i s.

We give one choice for z, y . Other choices are explored in the exercises.

The probability that a min cut survives z contractions is about $((n-z)/n)^2$. So heuristically, we decide to do contraction till this probability drops to $1/2$. This requires us to choose $n-z = n/\sqrt{2}$. At this point, the number of vertices drops down to $n-z = n/\sqrt{2}$. We choose $y = 2$ giving the recurrence for the time $T(n)$ for solving the n node problem:

$$T(n) \leq O(n^2) + 2T(n/\sqrt{2})$$

It is easily seen that this evaluates to $T(n) = O(n^2 \log n)$.

To estimate the probability of the cut surviving, we need to understand how the recursion progresses. Since the number of vertices drops by a factor of $\sqrt{2}$ at each call, the recursion must have depth $2 \log n$.

Define P_d to be the probability that the correct answer is returned by call at height h given that the min-cut is alive at the time of making this call. For the correct answer to be returned, the cut must survive the first z contractions which happens with probability $1/2$. Given this, at least one of the recursive calls at the level $d-1$ should return the correct answer, which happens with probability $(1 - (1 - P_{d-1})^2)$. Thus we get

$$P_d = \frac{1}{2} \cdot (1 - (1 - P_{d-1})^2) = P_{d-1} - \frac{1}{2}P_{d-1}^2$$

This recurrence solves to $P_d \geq 1/(d+1)$, as follows. First consider $f(x) = x - x^2/2$. This function is increasing for $0 < x < 1$, and hence $1 > x \geq x_0 > 0 \Rightarrow f(x) \geq f(x_0)$.

Thus we get

$$P_d = P_{d-1} - \frac{1}{2}P_{d-1}^2 \geq \frac{1}{d} - \frac{1}{2d^2} \geq \frac{1}{d} - \frac{1}{d(d+1)} = \frac{1}{d}$$

We want the probability $P_{2 \log n}$, which is at least $\frac{1}{1+2 \log n}$. Thus if we repeat this $\Omega(\log^2 n)$ times, we will get failure probability to be $O(n^{-\Omega(1)})$.

4 Summary

The basic idea of using random contraction is just genius. The idea of repeating the core algorithm to boost the success probability should now become second nature to you.

After that, the ideas of using deterministic algorithms after the problem is reduced substantially and also the idea of selectively repeating the work may in some sense be considered somewhat natural developments. These improvement ideas will work in other algorithms too.

Exercise

1. Show how the basic algorithm core can be implemented in $O(n^2)$ time. Give the required data structures.
2. Suppose in the final algorithm we use $z = n/2$. For the new value of z what value of y should you choose? Note that we would like to still get the same total time and high probability, i.e. time of $O(n^2 \log^3 n)$ and failure probability $n^{-\theta(1)}$.