1 Global Min-Cut

In the previous sections we have used network flow to find s-t cuts of minimum value (capacity) in a graph. When the capacities are all unit this corresponds to the fewest number of edges which disconnect the source s from the sink t. We now look a variation of this problem which asks for a global cut using the minimum number of edges to disconnect a graph. In this section we give an example of a class of algorithms known as randomized algorithms to solve the global min-cut problem.

Given an undirected graph $G(V, E)$, a **global min-cut** is a partition of V into two subsets (A, B) such that the number of edges between A and B is minimized. Note that the value of the global min-cut is the minimum over all possible s-t cuts. This suggests one solution to the problem of finding the global min-cut: compute the min s-t cut for all pairs $s, t \in V$. This algorithm requires $O(n^2)$ calls to a min s-t cut (max s-t flow) solver. Note that we can adapt a Ford-Fulkerson based flow algorithm to an undirected graph by introducing two opposing directed edges for each undirected edge. A faster scheme can be implemented by noting that any node s must appear in one of A or B, meaning we can reduce the number of min s-t cut solves by a factor of n: we may fix s and find min s-t cut for all $t \in V$. This solution still requires $O(n)$ calls to a min s-t cut solver.

Historically, the intuition was that global min-cut was a harder problem than s-t cut. David Karger was able to show that this is not the case with a randomized algorithm.

Algorithm 1 Karger's Randomized Global Min-Cut

repeat
  Choose an edge $\{u, v\}$ uniformly at random from E.
  Contract the vertices u and v to a super-vertex w.
  Keep parallel edges but remove self-loops.
until G has only 2 vertices.
Report the corresponding cut.

Theorem 1 *The probability that the algorithm finds the minimum cut in G is at least $2/n^2$.*

Proof: Let F be the set of edges in a global min-cut and suppose $|F| = k$. Let E_i be the event that the algorithm does not contract an edge from F in step i. At step 1, since the minimum cut has value k, the degree of each vertex must be at least k. This together with the fact that the sum of the degrees is twice the number of edges implies that $|E| \geq kn/2$.

Then the probability that the first edge picked does not belong to F is therefore:

$$\begin{aligned} Pr(E_1) &= 1 - \frac{k}{|E|} \\ &\geq 1 - \frac{2}{n} \end{aligned}$$

Similarly,

$$\begin{aligned} Pr(E_2|E_1) &\geq 1 - \frac{2}{n-1} \\ Pr(E_i|E_1 \cap E_2 \cap \ldots) &\geq 1 - \frac{2}{n-i+1} \end{aligned}$$

Combining these to get the total probability of success gives

$$\begin{aligned} Pr(\text{success}) &= Pr(E_1 \cap E_2 \cap \ldots \cap E_{n-2}) \\ &\geq Pr(E_1)Pr(E_2|E_1)\ldots Pr(E_{n-2}|E_1 \cap \ldots E_{n-3}) \\ &\geq (1-\frac{2}{n})(1-\frac{2}{n-1})\ldots(1-\frac{2}{3}) \\ &\geq 2\frac{(n-2)!}{n!} \geq \frac{2}{n(n-1)}, \end{aligned}$$

So the algorithm will succeed at with probability at least $2/n^2$. ■

The probability of finding a min-cut seems low at face value, and goes to zero as $n \to \infty$. However, if we run the algorithm t times and output the smallest min-cut found over all runs, the probability of success will be at least

$$1-(1-\frac{2}{n^2})^t.$$

So if we set $t = cn^2$ for some constant c, the probability of failure will be at most $(1-\frac{2}{n^2})^{cn^2} \leq e^{-2c}$. Thus to ensure the probability of failure is smaller than some fixed constant ϵ, it is only necessary to run the algorithm $\frac{1}{2}\log(\frac{1}{\epsilon})n^2$ times.

From the proof of Theorem 1 we may see that the probability of failure (contracting an edge of F) is much greater for later steps of the algorithm. In the last step alone we can only guarantee a successful contraction $1/3$ of the time. It would seem that we can improve the success probability with little extra work by repeating the last few steps of Karger's algorithm and returning the best cut found. This intuition is valid; in Karger's original paper[1], he outlines a recursive algorithm based on increasingly retrying later contraction steps that runs even faster than Algorithm 1 to achieve any constant failure probability. While the "vanilla" Karger's algorithm outlined in these notes requires $O(n^2)$ runs of an $O(n^2)$ contraction process for an overall complexity of $O(n^4)$, with relatively little effort this running time may be improved to $O(n^2\log^2(n))$.

We also note that in the proof of Theorem 1, we are computing the probability that a particular min-cut survives the contraction process. We prove that for any cut of minimum size, there is at least a $\frac{2}{n(n-1)}$ chance that it is produced by the contraction process. Considering the fact that this is true for all min-cuts, we have the following corollary:

Corollary 1 *The number of min-cuts in G is $\leq n(n-1)/2$.*

By considering a cycle on n vertices, we see that this bound is tight. The proof technique of Theorem 1 may be extended even further to bound the number of near-minimum cuts of a graph. An α-min-cut is a cut of size $\leq \alpha \cdot$ (size of the min-cut). Using a similar argument, we may show that the probability that any α-min-cut survives the contraction process is $\geq n^{-2\alpha}$. This implies the following bound:

Corollary 2 *The number of α-min-cuts in G is $\leq n^{2\alpha}$.*

Finally, we note that we may adapt Karger's algorithm to enumerate all of these min-cuts or near-min-cuts of G with high probability. For example, if we run the contraction process given in these notes $2n^2\log n$ times, the probability that we miss any particular min-cut is $\leq 1/n^4$. So if we keep track of all of the smallest cuts we see along the contraction process, the probability that we miss any min-cuts at all is $\leq n(n-1)/2 \cdot 1/n^4 = O(1/n^2)$. That is, increasing the run-time by a multiplicative factor of $O(\log n)$ allows us to guarantee with high probability that we see all of the min-cuts.

[1]Karger D. and Stein C. *A New Approach to the Minimum Cut Problem.* `http://www.columbia.edu/~cs2035/courses/ieor6614.S09/Contraction.pdf` (1996)

2 Network Reliability

One application of the above bounds on the number of min-cuts/near-min-cuts is to prove guarantees in network reliability. Consider a graph $G(V, E)$ and assume a one-time-step reliability model where each edge $e \in E$ fails, i.e. is removed from the graph, independently with probability p. What is $Pr[FAIL(G)]$, where $FAIL(G)$ is the event that G becomes disconnected as a result of the edge failures?

Writing $|V| = n$, there are exactly $N = 2^{n-1} - 1$ cuts of G (this is the number of ways to partition V into two non-empty subsets). G will become disconnected through edge removals iff for some cut, all edges that cross the cut are removed. Define $E_1, ..., E_N$ to be the sets of edges crossing each of the cuts of G, sorted so that $|E_1| \leq |E_2| \leq ... \leq |E_N|$.

For each $e \in E$, we may define the random variable X_e such that

$$X_e = \begin{cases} 1 & \text{if } e \text{ fails} \\ 0 & \text{otherwise} \end{cases}.$$

Then

$$FAIL(G) = F = C_1 \vee C_2 \vee ... \vee C_N$$

where $C_i = \bigwedge_{e \in E_i} X_e$. This expression for $FAIL(G)$ is an example of a logical formula written in disjunctive normal form (DNF): it is a formula consisting of the disjunction of one or more conjunctions of Boolean variables.

2.1 Counting the Number of Solutions of a DNF

The network reliability question can then be reposed as a problem of counting the number of solutions of a DNF. That is, how many of the 2^m total assignments to $X_1, ..., X_m$ satisfy the formula F? Call this number of satisfying assignments $\#F$. Counting $\#F$ explicitly is difficult; for each clause C_i it is simple to compute $S_i = \{\text{the set of assignments satisfying } C_i\}$, we may even note that $|S_i| = 2^{m-|E_i|}$. We wish, however, to know

$$\#F = |S_1 \cup ... \cup S_N|$$

and the over-counting associated with having assignments belong to multiple S_i's is a source of much difficulty. Indeed, the exact counting problem is #P-hard, but estimation schemes exist.

One workaround is to approximate $\#F$ using a Monte Carlo procedure. Define a random variable Z to be computed by drawing a random assignment $\mathbf{X}$ for $X_1, ..., X_m$, i.e. by drawing each $X_i \sim Bernoulli(1/2)$ independently, and setting

$$Z = \begin{cases} 0 & \text{if the formula } F \text{ is not satisfied} \\ 2^n & \text{otherwise} \end{cases}.$$

Then we have

$$\begin{aligned} \mathrm{E}[Z] &= \sum_{\mathbf{X}} Z(\mathbf{X}) Pr(\mathbf{X}) \\ &= \sum_{\mathbf{X} \text{ satisfies } F} 2^n \left(\frac{1}{2^n}\right). \\ &= \#F. \end{aligned}$$

We might hope to estimate $\mathrm{E}[Z]$ by sampling k copies of Z and using the central limit theorem to claim:

$$\mathrm{E}[Z] \approx \frac{1}{k} \sum_{i=1}^{k} Z_i = \frac{1}{k} \sum_{i=1}^{k} Z(X_1^i, ..., X_m^i),$$

but to achieve meaningful results this approach could require a very large number of trials k if the variance of Z is large. This will be the case if there are few satisfying assignments for F.

Instead of the basic Monte Carlo procedure described above, we may use a standard variance reduction technique called importance sampling to estimate $\#F$. The idea is to sample random assignments for $X_1, ..., X_m$ in a way that assigns nonzero probability to only the satisfying assignments of F; then we reweight our observations accordingly. Define the random variable Y by the following process:

- Choose a clause C_i with probability $\propto |S_i|$ (i.e. with probability $|S_i|/M$, where $M = \sum_{i=1}^{N} |S_i|$).
- Choose $\tau \in S_i$ uniformly at random.
- Compute $C(\tau) =$ the number of clauses satisfied by τ.
- Take $Y = \frac{M}{C(\tau)}$.

Claim 1 $\mathrm{E}[Y] = \#F$.

Proof: The probability that we choose a particular satisfying assignment τ for computing Y is:

$$Pr(\tau) = \sum_{i:\tau \text{ satisfies } C_i} \frac{|S_i|}{M} \cdot \frac{1}{|S_i|} = \frac{C(\tau)}{M}.$$

Then

$$\begin{aligned} \mathrm{E}[Y] &= \sum_{\tau \text{ satisfies } F} Y(\tau) Pr(\tau) \\ &= \sum_{\tau \text{ satisfies } F} 1 \\ &= \#F. \end{aligned}$$

■

The advantage of using Y is that we may bound its variance in comparison to its expectation. Noting that Y only takes on values in the range $[\frac{M}{N}, M]$, we may see that

$$\frac{\sigma(Y)}{\mathrm{E}[Y]} = \frac{\sqrt{\mathrm{E}[(Y - \mathrm{E}[Y])^2]}}{\mathrm{E}[Y]} \leq \frac{M - M/N}{M/N} = N - 1.$$

It turns out that we need only sample Y polynomially many times (in n and $1/\epsilon$ for an accuracy parameter ϵ) in order to estimate $\#F$ to a high degree of accuracy.

Claim 2 *Let Y_k denote the mean of k independent samples of Y. For any $\epsilon > 0$,*

$$Pr[|Y_k - \#F| \leq \epsilon \#F] \geq 3/4$$

for $k = 4(N-1)^2/\epsilon^2$.

Proof: We use Chebyshev's inequality to write:

$$Pr[|Y_k - \mathrm{E}[Y_k]| \geq \epsilon \mathrm{E}[Y_k]] \leq \left(\frac{\sigma(Y_k)}{\epsilon \mathrm{E}[Y_k]}\right)^2 = \left(\frac{\sigma(Y)}{\epsilon\sqrt{k}\mathrm{E}[Y]}\right)^2 \leq \frac{1}{4}.$$

■

Claim 2 is the basis for a fully polynomial randomized approximation scheme (FPRAS) for computing $\#F$. An FPRAS, in general, is an approximation algorithm $\mathcal{A}$ to a problem f that satisfies

$$Pr[|A(x) - f(x)| \leq \epsilon f(x)] \geq 3/4$$

for each problem instance x and error parameter $\epsilon > 0$, and the running time of $\mathcal{A}$ is polynomial in $|x|$ and $1/\epsilon$. We note that in order to increase the success probability for any FPRAS from $3/4$ to $(1-\delta)$, for any $\delta > 0$, we need only increase the running time by a multiplicative factor of $O(\log(1/\delta))$. We achieve this by running the FPRAS $O(\log(1/\delta))$ times and outputting the median value.

To prove that this median achieves the desired error probability, we first describe the Chernoff bounds[2] on the tail distributions of sums of Bernoulli random variables. Let $X_1, ..., X_n$ be independent random variables with $X_i \sim Bernoulli(p_i)$. Let $X = \sum_{i=1}^{n} X_i$, then we have $\mathrm{E}[X] = \mu = \sum_{i=1}^{n} p_i$. Chernoff bounds allow us to estimate the tail probabilities that X will deviate far from its mean.

$$Pr\left[X > (1+\delta)\mu\right] < \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{\mu}$$
$$Pr\left[X < (1-\delta)\mu\right] < e^{-\frac{\mu\delta^2}{2}}$$

These inequalities are part of a larger class of bounds called concentration inequalities that provide estimates on how a random variable deviates from some central value.

Now imagine running $\mathcal{A}$ a total of $k = 24\log(1/\delta)$ times on some problem input x, obtaining outputs $y_1, ..., y_k$. Let m be the median of these k values. Let $X_i = \mathbb{1}\{|y_i - f(x)| \leq \epsilon f(x)\}$ be the indicator random variable for the output of the i-th run being successful. Note that $\mathrm{E}[\sum_{i=1}^{k} X_i] \geq (3/4)k$. Then

$$\begin{aligned}
Pr\left(|m - f(x)| \not\leq \epsilon f(x)\right) &\leq Pr\left(\sum_{i=1}^{k} X_i < k/2\right) \\
&\leq Pr\left(\sum_{i=1}^{k} X_i < \left(1 - \frac{1}{3}\right)\mathrm{E}\left[\sum_{i=1}^{k} X_i\right]\right) \\
&\leq \exp\left(-\frac{\mathrm{E}\left[\sum_{i=1}^{k} X_i\right](1/3)^2}{2}\right) \\
&\leq \exp\left(-\frac{(3/4)k}{18}\right) = \delta.
\end{aligned}$$

where the second-to-last inequality makes use of a Chernoff bound.

2.2 Estimating the Probability that a DNF is Satisfied

Before returning to network reliability, we must address the issue that estimating the probability that a DNF formula is satisfied (given a distribution D over assignments) is not quite the same as estimating the number of solutions it has. This is because certain satisfying assignments may be more likely to occur than others. However, we may adapt our counting FPRAS to get one for probability estimation. Let q_i denote the probability that clause C_i is satisfied, and set $M = \sum_{i=1}^{N} q_i$. Define the random variable Y by this process instead:

- Choose a clause C_i with probability q_i/M.

[2] See `http://en.wikipedia.org/wiki/Chernoff_bounds` for proofs and more information on the following inequalities.

- Choose $\tau \in S_i$ with probability $Pr_D(\tau)/q_i$.
- Compute $C(\tau) =$ the number of clauses satisfied by τ.
- Take $Y = Q/C(\tau)$.

Then $\mathrm{E}[Y] = Pr_D[F]$, our desired quantity; the proof that this random variable is indeed the basis of an FPRAS is similar to that of Claims 1 and 2.

2.3 Application to Network Reliability Estimation

Recall: we wish to estimate the failure probability $Pr(F) = Pr(C_1 \vee C_2 \vee ... \vee C_N)$, where $C_i = X_{e_1^i} \wedge ... \wedge X_{e_{j_i}^i}$ denotes the event that the graph becomes disconnected across a particular cut as a result of edge failures. F, as it stands, is actually too large to straightforwardly apply the results of the previous sections. The number of terms in the DNF is $N = 2^{n-1} - 1$ which is exponential in the size of G. We present an alternative approach built around the following idea: only a polynomial number of "small" cuts are responsible for the majority of the total failure probability $Pr(F)$. This is because the failure probability $Pr(C_i)$ for a cut is $p^{|E_i|}$ where $|E_i|$ is the number of edges crossing the cut; this probability decreases exponentially with cut size.

For reasons that will become clear later, we consider now only the case where $Pr(F) \leq n^{-4}$. Separate the cuts into two groups: the set of "small" cuts defined to be the set of α-min-cuts of G, and "large" cuts which are all of the rest. We have the following bound on the failure probability of the large cuts:

Claim 3 *Let c denote the size of a minimum cut of G. Then the failure probability of a min-cut is $p^c \leq Pr(F) \leq n^{-4}$. Let $p^c = n^{-(2+\delta)}$ where $\delta \geq 2$. For any $\alpha \geq 1$,*

$$Pr(\textit{one of the "large" cuts fails}) = Pr(\textit{some cut of capacity} \geq \alpha c \textit{ fails}) \leq n^{-\alpha\delta}\left(1 + \frac{2}{\delta}\right)$$

Proof: We note from Corollary 2 of Section 7 that the number of cuts of size βc is at most $n^{2\beta}$. This implies that $|E_{n^{2\beta}}| \geq \beta c$ for any $\beta > 1$, i.e. $|E_k| \geq c \log k/(2 \log n)$. Let a be the index of the first large cut. Then we may write using the union bound:

$$\begin{aligned}
Pr(\text{some cut of capacity} \geq \alpha c \text{ fails}) &\leq \sum_{i \geq a} Pr(C_i) \\
&\leq \sum_{i=a}^{a+n^{2\alpha}} Pr(C_i) + \sum_{i > n^{2\alpha}} Pr(C_i) \\
&\leq n^{2\alpha} p^{\alpha c} + \sum_{i > n^{2\alpha}} p^{c \log i/(2 \log n)} \\
&\leq n^{-\alpha\delta} + \int_{n^{2\alpha}}^{\infty} k^{-(1+\delta/2)} \, \mathrm{d}k \\
&\leq n^{-\alpha\delta} + \frac{2}{\delta} n^{-\alpha\delta} = n^{-\alpha\delta}\left(1 + \frac{2}{\delta}\right)
\end{aligned}$$

■

By taking $\alpha = 2 - \frac{\log(\epsilon/2)}{2 \log n}$, the above claim and some algebraic manipulation allows us to say that

$$Pr(F) \geq Pr(\text{one of the "small" cuts fails}) \geq (1 - \epsilon) Pr(F).$$

Then using Karger's min-cut algorithm we may enumerate the α-min-cuts of G in polynomial time and construct a corresponding DNF formula F'. This formula has at most $n^{2\alpha} = O(n^4/\epsilon)$ clauses, and then we may use the results of the previous sections to accurately estimate the probability that F' is satisfied (i.e. Pr(one of the "small" cuts fails)) in polynomial time. Putting this all together gives an FPRAS for estimating network reliability in the case that $Pr(F) \leq n^{-4}$.

If $Pr(F) > n^{-4}$, we may resort to simple Monte Carlo sampling to estimate the failure probability. That is, we may disconnect each edge of G with probability p and then check to see if G is disconnected. If we repeat this experiment $O(\log n/(\epsilon^2 Pr(F))$ times and output the mean number of times G is disconnected, a simple application of Chernoff bounds guarantees that the result lies in

$$[(1-\epsilon)Pr(F), (1+\epsilon)Pr(F)]$$

with high probability.