

## Lecture 2: January 9, 2003

Lecturer: Eric Vigoda

Scribe: Murali Krishnan Ganapathy

In this lecture, we will consider the problem of calculating the probability that a given network fails, and give a FPRAS (Fully Polynomial Randomized Approximation Scheme) for the same. The work on reliability is Karger, see [1, 2], and the earlier work on  $\#DNF$  is Karp and Luby [4, 3].

## 2.1 Problem Statement

Fix an undirected graph  $G$  and  $0 \leq p \leq 1$ . Let  $H$  be a graph obtained by deleting edges of  $G$ , independently with probability  $p$ . Let  $FAIL_G(p)$  denote the probability that the graph  $H$  obtained as above is disconnected. The network reliability problem is to calculate  $FAIL_G(p)$  given inputs  $G$  and  $p$ .

## 2.2 Overview

For an undirected graph  $H$ , let  $MINCUT(H)$  denote its min-cut. First observe that a graph  $H$  is disconnected iff  $MINCUT(H) = 0$ . If  $G$  is an undirected graph and  $H$  is obtained by deleting some edges of  $G$ , then  $H$  is disconnected iff the deleted edges contain some cut of  $G$  (not necessarily a min-cut). Hence, we have that

$$FAIL_G(p) = \Pr(E(G) \setminus E(H) \text{ contains a cut of } G) \quad (2.1)$$

where  $E(G)$  denotes the edge set of  $G$ .

Since, calculating the exact value of  $FAIL_G(p)$  is  $\#P$ -complete, we will try to develop an FPRAS for the same, i.e. there is an algorithm  $\mathcal{A} \ni \forall \epsilon > 0, \forall \delta > 0$  and all inputs  $(G, p)$ , we have

$$(1 - \epsilon)FAIL_G(p) \leq \mathcal{A}(G, p) \leq (1 + \epsilon)FAIL_G(p), \text{ with probability } \geq 1 - \delta$$

and the running time of  $\mathcal{A}$  is polynomial in the input size,  $1/\epsilon$  and  $\log(1/\delta)$ . Here the input size is  $O(|G| + \log(1/p))$ .

One simple way to solve this problem is to keep generating random  $H$ 's, and observe how many of them turn out to be disconnected. For this to work, we need to generate at least  $1/FAIL_G(p)$  random  $H$ 's, and then some if we want to keep the error down. Since  $FAIL_G(p)$  can be exponentially small, this algorithm will not cut it.

If  $FAIL_G(p)$  is “polynomially small”, we do as suggested above. In the alternate case, we show that if in equation (2.1), we keep only the terms corresponding to “small cuts”, then the right hand side, is a good enough approximation to  $FAIL_G(p)$ .

In the rest of this lecture, we make the terms “polynomially small” and “small cuts” more precise and show that the algorithm does indeed work as advertised. For the rest of this lecture  $(G, p)$  denotes a fixed input to our algorithm. Similarly, we fix some  $\epsilon > 0$  and  $\delta > 0$ . Let  $C := MINCUT(G)$ . This is calculated by using Karger’s algorithm discussed in the previous lecture. Finally put  $q := FAIL_G(p)$  and observe that

$$q = \text{FAIL}_G(p) \geq \Pr(E(G) \setminus E(H) \text{ contains a specific cut}) \geq p^C.$$

Now we consider two cases depending on whether  $p^C$  is greater or smaller than  $n^{-4}$ . This defines “polynomially small”.

### 2.3 $p^C \geq n^{-4}$

In this case, we show that the Monte Carlo method discussed before actually works. Put  $\ell = \frac{3n^4}{\epsilon^2} \log(2/\delta)$ . All in all, we do  $\ell$  trials, and in each trial we create a random  $H$  by tossing coins of bias  $p$ , once for each edge of  $G$ , and then check if the  $H$  generated is connected. Finally, we output the fraction of the  $H$ 's which were connected.

For  $i = 1, \dots, \ell$ , set  $X_i$  equal to 1 if the  $i$ 'th trial resulted in a disconnected graph and 0 otherwise, and put  $X = \sum_i X_i$ . Since  $\Pr(X_i = 1) = \text{FAIL}_G(p) = q$ ,  $E(X) = q\ell$ . Since the  $X_i$ 's are independent, by Chernoff's bound we have,

$$\begin{aligned} \Pr(|X - E(X)| \geq \epsilon E(X)) &\leq 2 \exp(-\epsilon^2 q \ell / 3) \\ &\leq 2 \exp(-\epsilon^2 n^{-4} \frac{n^4}{\epsilon^2} \log(2/\delta)) \\ &= 2 \exp(-\log(2/\delta)) \\ &= \delta \end{aligned}$$

Each trial involves generating the random  $H$ , and testing it for connectedness. Checking connectedness takes time  $O(|G|)$ . Generating the random  $H$  requires time  $O(|G| \log(1/p))$ , assuming that we only have access to unbiased coins. So, total time is  $O(\ell |G| \log(1/p))$ , which is polynomial in the size of the input,  $1/\epsilon$  and  $\log(1/\delta)$ .

### 2.4 Enumerating small cuts

In this section we consider the problem of enumerating the “small cuts”. This will be used as a subroutine to solve the main problem at hand. The algorithm is a variation of Karger's algorithm to find a cut of size  $\leq \alpha C$ , where  $\alpha > 1$  is a parameter:

1. Pick a random edge and contract it (eliminating self loops and keeping multiple edges).
2. Repeat step 1, till there are  $k := \lceil 2\alpha \rceil$  meta-vertices left.
3. Choose a random subset  $T$  of these  $k$ -vertices.
4. If the cut defined by  $T$  and its complement, has size  $\leq \alpha C$ , output it.

Fix a cut  $\mathcal{S}$  of size  $\leq \alpha C$ . For the above algorithm to output  $\mathcal{S}$ , it must not choose any edge in  $\mathcal{S}$  during step 1, and in step 3, the random subset  $T$  should correspond to one side of the cut defined by  $\mathcal{S}$ . Since every vertex (of every intermediate graph) has degree  $\geq C$  (since  $C = \text{MINCUT}(G)$ ),

$$\begin{aligned} \Pr(\mathcal{S} \text{ survives till step 3}) &\geq \prod_{t=0}^{n-k-1} \left(1 - \frac{\alpha C}{(n-t)C/2}\right) \\ &= \frac{\binom{n-2\alpha}{n-k}}{\binom{n}{k}} \end{aligned} \tag{2.2}$$

$$\geq \frac{1}{\binom{n}{k}} \tag{2.3}$$

which, assuming  $k \geq 4$ , implies that

$$\begin{aligned} \Pr(\text{output} = \mathcal{S}) &\geq \frac{1}{\binom{n}{k}} 2^{-k+1} \\ &\geq 2 \left(\frac{k}{2en}\right)^k \\ &\geq n^{-k} \quad \text{for } \alpha > 2 \\ &= n^{-\lceil 2\alpha \rceil} \end{aligned}$$

Since  $\mathcal{S}$  was an arbitrary cut of size  $\leq \alpha C$ , we have that the number of cuts of size  $\leq \alpha C$  is  $\leq n^{\lceil 2\alpha \rceil}$ . This bound can be easily improved to  $n^{-2\alpha}$  if we replace the right hand side of equation (2.2) with  $\binom{k}{2\alpha} / \binom{n}{2\alpha}$ , whose definition would involve  $\Gamma$  functions.

In order to enumerate all “small cuts” with high probability, we repeat the above algorithm  $r := \theta \log(\theta/\delta)$  times, where  $\theta = n^{\lceil 2\alpha \rceil}$ . For a specific cut  $\mathcal{S}$ ,  $\Pr(\mathcal{S}$  was not the result of the  $i$ 'th trial)  $\leq 1 - 1/\theta$ . Hence the probability that  $\mathcal{S}$  was not the result any of the  $r$ -trials is bounded above by  $(1 - 1/\theta)^r$ . This implies that the probability that some “small cut” did not appear in any of the  $r$ -trials, is bounded above by  $\theta(1 - \theta^{-1})^{\theta \log(\theta/\delta)} \leq \delta$ .

This gives us an algorithm which for any  $\alpha > 1, \delta > 0$ , enumerates all cuts of a graph of size  $\leq \alpha C$  with probability  $1 - \delta$  and running time  $O(rn^2)$ .

## 2.5 A FPRAS for #DNF

In this section, we consider the problem of estimating the number of satisfying assignments of a DNF formula, and its weighted version.

Let  $\Phi$  be a DNF formula with  $M$  clauses  $\theta_1, \dots, \theta_M$ . Let  $r_i$  be the number of literals appearing in  $\theta_i$ . Let  $N$  denote the total number of distinct variables in  $\Phi$ . Without loss of generality assume  $\Phi$  is satisfiable (checking this can be done in time  $O(\sum r_i)$ ). Let  $0 \leq p \leq 1$ , be given. Consider a random assignment with bias  $p$ . We wish to calculate the probability,  $f(\Phi)$ , of the random assignment satisfying  $\Phi$ . In case  $p = 1/2$ , it reduces to the problem of counting the number of satisfying assignments.

The obvious trick of choosing a random assignment (with required bias) and checking if it satisfies  $\Phi$  is out of question, since  $f(\Phi)$  can be exponentially small. Let  $w_i$  denote the probability that a random assignment satisfies the  $i$ 'th clause. If  $p_i$  and  $n_i$  denote the number of positive and negative literals appearing in  $\theta_i$ , then  $w_i = p^{p_i}(1-p)^{n_i}$ . Finally, put  $Z = \sum_i w_i$ . Consider the following algorithm:

1. Choose a random clause  $\theta_i$ , with probability of selecting  $\theta_i$  equal to  $w_i/Z$ .
2. Choose a random assignment  $\sigma$  satisfying  $\theta_i$ .
3. Calculate  $S = Z/N(\sigma)$ , where  $N(\sigma)$  is the number of clauses satisfied by  $\sigma$ .
4. Repeat steps 1-3  $t := (4M/\epsilon^2)$  times, and output the observed mean.

First we show that  $E(S) = f(\Phi)$ . For any assignment  $\sigma$ , let  $w(\sigma)$  denote the probability that a random assignment equals  $\sigma$ , i.e.  $w(\sigma) = p^a(1-p)^b$ , where  $a$  is the # of variables set to 1 and  $b = N - a$  is the # of variables set to 0.

$$\begin{aligned} \Pr(\text{choosing } \sigma \text{ in step 2} \mid \theta_i \text{ in step 1}) &= w(\sigma)/w_i \\ \implies E(S \mid \theta_i \text{ in step 1}) &= \sum_{\sigma: \sigma \models \theta_i} \frac{Z}{N(\sigma)} \frac{w(\sigma)}{w_i} \end{aligned}$$

This gives

$$\begin{aligned} E(S) &= \sum_i \frac{w_i}{Z} \sum_{\sigma \models \theta_i} \frac{Z}{N(\sigma)} \frac{w(\sigma)}{w_i} \\ &= \sum_i \sum_{\sigma \models \theta_i} w(\sigma)/N(\sigma) \\ &= \sum_{\sigma \models \Phi} \sum_{i: \sigma \models \theta_i} w(\sigma)/N(\sigma) \\ &= \sum_{\sigma \models \Phi} w(\sigma)/N(\sigma) \sum_{i: \sigma \models \theta_i} 1 \\ &= \sum_{\sigma \models \Phi} w(\sigma) \\ &= f(\Phi) \end{aligned}$$

Similarly

$$\begin{aligned} E(S^2) &= \sum_i \frac{w_i}{Z} \sum_{\sigma \models \theta_i} \frac{Z^2}{N(\sigma)^2} \frac{w(\sigma)}{w_i} \\ &= \sum_i \sum_{\sigma \models \theta_i} \frac{Z}{N(\sigma)^2} w(\sigma) \\ &= \sum_{\sigma \models \Phi} \frac{Z}{N(\sigma)} w(\sigma) \\ &\leq Z f(\Phi) \end{aligned}$$

the last inequality following from the fact that  $N(\sigma) \geq 1$  if  $\sigma \models \Phi$ . Now  $\text{Var}(S) = E(S^2) - E(S)^2 \leq (Z/f(\Phi) - 1)f(\Phi)^2 \leq (M - 1)f(\Phi)^2$ . To see that  $Z \leq Mf(\Phi)$ , define  $A_i$  to be the event that a random  $\sigma$  satisfies  $\theta_i$  and note that  $Z = \sum_i w_i = \sum_i \Pr(A_i) \leq M \Pr(\cup_i A_i) = Mf(\Phi)$ .

Now let  $S_1, \dots, S_t$  denote the observed values during the run of the algorithm, and let  $\hat{S}$  be the observed mean, i.e.  $\hat{S} = (\sum_i S_i)/t$ . Clearly,  $E(\hat{S}) = E(S) = f(\Phi)$  and  $Var(\hat{S}) = Var(S)/t \leq (M-1)f(\Phi)^2/t$ . Now applying Chebychev's inequality, we have

$$\Pr\left(|\hat{S} - f(\Phi)| \geq \epsilon f(\Phi)\right) \leq \Pr\left(|\hat{S} - f(\Phi)| \geq \frac{\epsilon\sqrt{t}}{\sqrt{M-1}} \sqrt{Var(\hat{S})}\right) \leq \frac{M-1}{\epsilon^2 t} \leq 1/4$$

To reduce the error probability to less than  $\delta$ , we repeat the above procedure  $\log(1/\delta)$  times and output the median of all the observed means.

## 2.6 $p^C \leq n^{-4}$

Now back to the original network reliability problem. In the case when  $q = FAIL_G(p)$  is small, we use equation 2.1, and consider only the terms on the right hand side corresponding to "small cuts". We now show that that one can choose  $\alpha > 1$ , so that if we restrict ourselves to only those terms which correspond to cuts of size  $\leq \alpha C$ , then the error we introduce is small. Then we enumerate all small cuts and approximate the probability using the FPRAS for #DNF.

Choose  $\zeta \geq 2$  such that  $p^C = n^{-(2+\zeta)}$ . Let  $H$  be a random graph obtained from  $G$  by removing some edges (in the prescribed manner). Then

$$\begin{aligned} \Pr(\text{some cut of size } \geq \alpha C \text{ fails in } H) &\leq \int_{\beta \geq \alpha} n^{2\beta} p^{\beta C} d\beta \\ &\leq \int_{\beta \geq \alpha} n^{-\zeta\beta} d\beta \\ &= O(1) * n^{-\zeta\alpha} \end{aligned}$$

Setting  $\alpha = 2 - \ln(\epsilon/O(1))/\zeta \ln n$ , we see that  $\Pr(\text{large cut failing in } H) \leq \epsilon n^{-2\zeta} \leq \epsilon p^C \leq \epsilon q$ . Therefore, by ignoring all "large cuts" (i.e. those of size  $> \alpha C$ ), we calculate  $FAIL_G(p)$  to within a multiplicative factor of  $1 \pm \epsilon$ .

Now, we enumerate all the "small cuts", using the algorithm from Section 2.4. Let  $\mathcal{C}_1, \dots, \mathcal{C}_r$  be the "small cuts". Define a DNF formula  $\Phi$  over the variables  $\{x_e\}_{e \in E(G)}$  as follows. For each cut  $\mathcal{C}_i$ , define the clause  $\theta_i = \bigwedge_{e \in \mathcal{C}_i} \bar{x}_e$ . Set  $x_e$  to true if the edge appears in  $H$ , and false otherwise. Then  $\theta_i$  evaluates to true exactly when the cut  $\mathcal{C}_i$  fails in  $H$ . Hence  $\Phi$  evaluates to true iff some "small cut" fails in  $H$ . So, we are left with the problem of evaluating the probability that a random assignment of truth values to  $x_e$  (true with prob  $p$ ) satisfies  $\Phi$ . This is solved using the FPRAS of section 2.5.

Why can't we estimate  $FAIL_G(p)$  for all graphs by enumerating small cuts? I.e., do we need to partition the algorithm into cases based on the size of the min-cut? Consider a line with  $n$  vertices and  $p = 1/2$ . Hence,  $FAIL_G(p) = 1 - 2^{-n} \approx 1$ . However, small cuts give a poor estimate. We need to consider cuts of size  $\Theta(n)$  to get a good estimate.

## 2.7 Putting it all together

Combining all the previous discussion the final algorithm is:

1.  $C = \text{MINCUT}(G)$
2. If  $p^C > n^{-4}$  goto step 3, else goto step 4
3. **(Monte Carlo)** Using  $\ell := \frac{3n^4}{\epsilon^2} \log(2/\delta)$  trials, estimate  $\text{FAIL}_G(p)$  to within  $1 \pm \epsilon$  with probability  $1 - \delta$  and end program.
4. Calculate  $\alpha = 2 - \ln(\epsilon/O(1))/\zeta \ln n$ , where  $\zeta$  satisfies  $p^C = n^{-2-\zeta}$ .
5. Using the extension of Karger's min-cut algorithm described in Section 2.4, enumerate all cuts of size  $\leq \alpha C$ , with probability  $1 - \delta$ .
6. Construct the DNF  $\Phi$  as described in Section 2.6.
7. Using FPRAS of section 2.5 evaluate  $f(\Phi)$  to within a multiplicative error of  $1 + \epsilon$  with probability  $1 - \delta$ .
8. Output  $f(\Phi)$

This algorithm calculates  $\text{FAIL}_G(p)$  to within a multiplicative error of  $(1 \pm \epsilon)$  with probability  $(1 - \delta)^2$ . Each step takes polynomial time w.r.t its input,  $1/\epsilon$  and  $\log(1/\delta)$ . Composing all this we get an algorithm whose running time is polynomial in  $|G|, |\log p|, 1/\epsilon$  and  $|\log \delta|$ , and the cumulative multiplicative error is bounded by  $(1 \pm \epsilon)$  with error probability bounded by  $1 - (1 - \delta)^2$ .

## 2.8 Open problems

Is there a simpler, more natural algorithm for network reliability. For example, here is a proposed random walk approach. Let  $\Omega$  denote the disconnected subgraphs of our input graph  $G = (V, E)$ , where the elements of  $\Omega$  are subsets of  $E$ . The following Markov chain has stationary distribution uniformly distributed over  $\Omega$ . From  $X_t \in \Omega$ , choose an edge  $e$  uniformly at random from  $E$ . Set  $X' = X \setminus \{e\}$  with probability  $1/2$  and  $X' = X \cup \{e\}$  otherwise. If  $X' \in \Omega$ , set  $X_{t+1} = X'$ , otherwise set  $X_{t+1} = X_t$ . Can we prove that this chain converges quickly to its stationary distribution? An efficient method to generate (almost) uniform samples yields an efficient approximate counter.

Another project is to devise a deterministic approximation algorithm for  $\text{FAIL}_G(p)$ . Karger [2] showed that his approach can be efficiently derandomized when  $\text{FAIL}_G(p) \leq 1/n^4$ . However it is open how to derandomize the case  $\text{FAIL}_G(p) > 1/n^4$ . The high-level idea for small failure probabilities is the following. There is an efficient deterministic algorithm to list all small cuts via max-flow computations. Then  $\text{FAIL}_G(p)$  can be estimated via inclusion-exclusion where only the first few terms are needed for a sufficient estimate.

## References

- [1] D. R. Karger. A randomized fully polynomial time approximation scheme for the all-terminal network reliability problem. *SIAM J. Comput.*, 29(2):492–514, 1999.
- [2] D. R. Karger. A randomized fully polynomial time approximation scheme for the all-terminal network reliability problem. *SIAM Rev.*, 43(3):499–522, 2001. Reprint of *SIAM J. Comput.* **29** (1999), no. 2, 492–514.
- [3] R. M. Karp and M. Luby. Monte Carlo algorithms for the planar multiterminal network reliability problem. *J. Complexity*, 1(1):45–64, 1985.

- [4] R. M. Karp, M. Luby, and N. Madras. Monte Carlo approximation algorithms for enumeration problems. *J. Algorithms*, 10(3):429–448, 1989.