COMPSCI 638: Graph Algorithms October 2, 2019

Lecture 11

Lecturer: Debmalya Panigrahi *Scribe: Kevin Sun*

1 Overview

In this lecture, we begin studying an area of graph algorithms known as network reliability. For this problem, we will give a polynomial-time algorithm based on Monte Carlo sampling and a problem known as DNF counting.

2 An FPRAS for Network Reliability

In this problem, we are given a graph $G = (V, E)$, and we are told that every edge e fails (i.e., disappears) independently with some probability p_e. Without loss of generality, we can assume $p_e = 1/2$ by replacing each edge e with a carefully defined "bundle" of edges such that the probability they all fail is p_e, while the probability that a single edge fails is $p = 1/2$.

Our goal is to compute $F(G, p)$, the probability that G disconnects if each edge fails independently with probability $p = 1/2$. If our estimate must be completely accurate, then this problem is equivalent to returning the probability that the graph does not fail, i.e., $1 - F(G, 1/2)$.

However, the complexity of this algorithm is known as #P-hard, which means we cannot expect to return an exact answer in polynomial time. Thus, we must settle for an approximation, but approximating $F(G, p)$ is *not* equivalent to approximating $1 - F(G, p)$.

In this section, we will give an algorithm due to Karger [Kar01] that is a fully-polynomial time randomized approximation scheme (FPRAS). Such an algorithm has an input accuracy parameter $\epsilon > 0$ and, for this problem, returns a value $\tilde{F}$ such that

$$(1 - \epsilon)F(G, 1/2) \le \tilde{F} \le (1 + \epsilon)F(G, 1/2)$$

with probability at least 3/4. (This success probability can be made arbitrarily close to 1 by repeating the algorithm a small number of times.) Furthermore, the running time of this algorithm must be polynomial in both n and $1/\epsilon$. For simplicity, we let $f = F(G, 1/2)$ denote the failure probability of the graph, so we are seeking a $(1 \pm \epsilon)$-approximation of f.

2.1 The case where $f \ge 1/n^4$

For this case, we use an idea known as Monte Carlo (MC) sampling: in each experiment, remove each edge of G with probability 1/2; repeat this experiment N times for some N. Then return the proportion of experiments that resulted in G becoming disconnected. The following lemma shows that if f is sufficiently large, then it suffices to choose some N that is $O(n^4/\epsilon^2)$.

Lemma 1. *If $f \ge 1/n^4$, then $O(n^4/\epsilon^2)$ rounds of MC sampling yields a $(1 \pm \epsilon)$-approximation of $f = F(G, 1/2)$ with probability at least* 3/4.

Proof. For $i = 1, \dots, N$, let X_i be a random variable equal to 1 if the i-th trial resulted in G becoming disconnected and 0 otherwise. Then $X_i = 1$ with probability f, so the expectation of $\sum_{i=1}^{N} X_i$ is Nf. By a Chernoff bound (see Lecture 9), we have

$$\Pr\left(\sum_{i=1}^{N} X_i \notin (1-\epsilon, 1+\epsilon)Nf\right) \le 2\exp\left(-\frac{\epsilon^2 Nf}{3}\right).$$

If we set $N = c/\epsilon^2 f$ for a suitable constant c, then the above quantity is at most $3/4$. Since $f \ge n^4$, this implies that the value of N is $O(n^4/\epsilon^2)$. $\square$

So we have successfully handled the case where $f \ge 1/n^4$, but our algorithm does not know the value of f. To address this, let λ denote the minimum cut value of G. Then $f \ge 1/2^\lambda$ because if the λ edges of a minimum cut fail, then G becomes disconnected. Thus, it suffices for the algorithm to check if $1/2^\lambda \ge 1/n^4$ for the purposes of Lemma 1.

2.2 The case where $f < 1/n^4$

The high-level idea is this: the probability that a large cut fails is tiny because it contains many edges. In fact, it is so tiny that the probability *any* large cut fails is negligibly small. So it suffices to focus on the small cuts, and the cut-counting lemma (see Lecture 8) tells us G does not have many small cuts. Finally, we estimate the probability that a small cut fails via a reduction to a problem known as DNF counting.

More formally, we begin by proving the following lemma. Recall that we are in the case where f is small, i.e., $1/2^\lambda \le f < n^{-4}$. So we can define $\delta > 2$ such that $1/2^\lambda = n^{-(\delta+2)}$.

Lemma 2. *Let δ be as defined above. Then there exists some α and constant c such that*

$$\Pr(\textit{any cut of size at least } \alpha\lambda \textit{ fails}) \le c \cdot \frac{1}{n^{\alpha\delta}} < \epsilon f.$$

Proof. First recall the cut-counting lemma, which states that there are at most $n^{2\alpha}$ cuts with size at most $\alpha\lambda$. Thus, by the definition of δ, we have

$$\Pr(\text{any cut of size exactly } \alpha\lambda \text{ fails}) \le \frac{1}{2^{\alpha\lambda}} \cdot n^{2\alpha} = \frac{1}{n^{(\delta+2)\alpha}} \cdot n^{2\alpha} = \frac{1}{n^{\alpha\delta}}.$$

The cut counting lemma allows us to sum over cuts with at least $\alpha\lambda$ edges without significantly increasing the value of the right hand side. So for simplicity, and without loss of generality, we now ignore the constant c. By setting $\alpha = 2 + \log_n(1/\epsilon)/2$ and using $\delta > 2$, we get

$$\frac{1}{n^{\alpha\delta}} = \frac{\epsilon^{\delta/2}}{n^{2\delta}} \le \frac{\epsilon}{n^4} < \epsilon f. \qquad \square$$

So Lemma 2 allows us to effectively ignore cuts with greater than $\alpha\lambda$ edges where $\alpha = 2 + \log_n(1/\epsilon)$. That is, let $f_s = \Pr(\text{any cut of size at most } \alpha\lambda \text{ fails})$. Then Lemma 2 implies $f \le f_s + \epsilon f$, so f_s is a $(1 \pm \epsilon)$-approximation of f. Thus, a $(1 \pm \epsilon)$-approximation of f_s is a $(1 \pm \epsilon)^2 \approx (1 \pm 2\epsilon)$-approximation of f. So for the rest of this section, we seek a $(1 \pm \epsilon)$-approximation of f_s.

Reduction to DNF counting: To estimate f_s, the number of cuts we must consider is at most $n^{2\alpha} = n^4/\epsilon$, where α was defined in the proof of Lemma 2. Thus, in fully polynomial time, we can list these cuts as $C_1, C_2, \ldots, C_k$ where $k \leq n^4/\epsilon^2$ using the Karger-Stein algorithm (see [KS96]).

Let x_e be a random variable that indicates whether or not edge e fails, and consider the variable $Y_i = \wedge_{e \in C_i} x_e$. Notice that Y_i indicates whether or not the cut C_i has failed, i.e., all of its edges have failed. Now consider the random variable

$$Z = \bigvee_{i=1}^{k} Y_i = Y_1 \vee Y_2 \vee \cdots \vee Y_k,$$

which indicates whether or not an α-minimum cut has failed, so f_s is precisely the probability that $Z = 1$ given that every x_e is 1 with probability $1/2$. Furthermore, the total size of Z is polynomial because $k \leq n^4/\epsilon^2$ and each Y_i contains at most $|E|$ variables.

Let $m' \leq m$ denote the number of distinct variables in Z, so that there are exactly $2^{m'}$ possible assignments. Since x_e is 1 with probability $1/2$, every assignment is equally likely. Thus, it suffices to count the number of assignments such that $Z = 1$; the value of f_s is that number divided by $2^{m'}$.

DNF counting: So the final piece of our FPRAS is solving the following problem, known as disjunctive normal form (DNF) counting: given a DNF formula $Z = \vee_i t_i$ containing m terms and n variables, count the number of assignments such that Z is true. (Note that in our case, we actually have a special case of this problem because our DNF formula has no negations.)

For this problem, we state an algorithm given by Karp, Luby, and Madras [KLM89]. Consider the following matrix M with m rows and 2^n columns, whose (i, j)-th entry is $\times$ if term t_i is satisfied by the j-th assignment A_j (in some fixed order of terms and variables). Furthermore, the first $\times$ in every column (if it exists) is circled, so that it becomes $\otimes$. We refer to both $\times$ and $\otimes$ as "cross," and the latter is also a "circled cross." An example of this matrix is given in Table 1.

	A_1	A_2	$\cdots$	A_{2^n}
t_1		$\otimes$		
t_2		$\times$		$\otimes$
t_3	$\otimes$			
$\vdots$				
t_m	$\times$	$\times$		$\times$

Table 1: The matrix used in the KLM algorithm [KLM89]. Each row t_i corresponds to a term in the DNF formula, and each column A_j corresponds to a truth assignment to the variables.

Thus, counting the number of satisfying assignments is equivalent to counting the number of non-empty columns. However, the number of empty columns might be very large, so Monte Carlo sampling over the columns would require too many iterations.

This is why we circle a cross in every non-empty column: it allows us to sample *crosses* instead of columns. Let r denote the fraction of circled crosses among all crosses; the key observation is $r \geq 1/m$. We can calculate the number of crosses exactly, so the product of these values is an estimate on the number of circled crosses, i.e., non-empty columns.

Lemma 3. *In polynomial time, we can sample a cross uniformly at random and determine if it's circled.*

Proof. We first describe how to sample a cross uniformly at random. For each i, let $|t_i|$ denote the number of crosses in the row corresponding to t_i. Also, let s_i denote the number of variables in t_i. Then $|t_i| = 2^{n-s_i}$ because there are $n - s_i$ variables that do not influence the truth value of t_i, and each of them can be assigned either true or false.

Once we've computed $|t_i|$, we sample a cross as follows: first sample each row with probability proportional to $|t_i|$. Once we have a row, sample one of its crosses by setting each of the variables not in the row to 1 with probability $1/2$. This yields an assignment that satisfies the chosen row term, so the corresponding entry contains a cross; this step takes linear time.

To determine if a sampled cross is circled, we calculate every cell in the column corresponding to the cross in $O(mn)$ time. We then scan the column to see if the sampled cross is the first cross in the column. $\square$

Now we apply a Chernoff bound to bound the number of MC iterations we need. Again, the key observation is that $r \geq 1/m$. By combining this with Lemma 3, we've obtained an FPRAS for DNF counting, and hence, an FPRAS for the network reliability problem.

Lemma 4. *In $O(m/\epsilon^2)$ iterations of MC sampling of the crosses, we obtain a $(1 \pm \epsilon)$-approximation of r with probability $3/4$.*

Proof. For $i = 1, \ldots, N$, let $R_i = 1$ if the i-th sampled cross is circled, so our estimate of r is $\sum_i R_i/N$. Then by a Chernoff bound, we see that

$$\Pr\left(\frac{\sum_{i=1}^{n} R_i}{N} \notin (1 \pm \epsilon)r\right) \leq 2\exp\left(-\frac{\epsilon^2 N r}{3}\right).$$

If we set $N = c/\epsilon^2 r$ for a suitable constant c, then the above quantity is at most $3/4$. Since $r \geq 1/m$, this implies that the value of N is $O(m/\epsilon^2)$. $\square$

3 Summary

In this lecture, we gave an FPRAS for the network reliability problem. We saw that the cut counting lemma allowed us to focus on cuts with small value, reduced the problem to DNF counting, and gave an algorithm for this new problem.

References

[Kar01] David R Karger. A randomized fully polynomial time approximation scheme for the all-terminal network reliability problem. *SIAM review*, 43(3):499–522, 2001.

[KLM89] Richard M Karp, Michael Luby, and Neal Madras. Monte-carlo approximation algorithms for enumeration problems. *Journal of algorithms*, 10(3):429–448, 1989.

[KS96] David R Karger and Clifford Stein. A new approach to the minimum cut problem. *Journal of the ACM (JACM)*, 43(4):601–640, 1996.