CS 6550: Randomized Algorithms **Spring 2019**

Lecture 19: #DNF and Network Unreliability

March 26, 2019

Lecturer: Eric Vigoda *Scribes: Sherry Sarkar, Samarth Wahal*

19.1 #DNF

19.1.1 Problem Statement

Definition 19.1 *A boolean formula F is said to be in Conjunctive Normal Form (CNF) if it is the conjunction of clauses $C_1 \wedge \ldots \wedge C_m$ where each clause is a disjunction of literals.*

Definition 19.2 *A boolean formula F is said to be in Disjunctive Normal Form (DNF) if it is the disjunction of clauses $C_1 \vee \ldots \vee C_m$ where each clause is a conjunction of literals.*

Example 19.3 *The following is a DNF formula:*

$$(x_2 \wedge x_3 \wedge \overline{x_1}) \vee (x_5) \vee (\overline{x_2} \vee \overline{x_4} \vee x_3)$$

Satisfying a DNF formula F is easy since we can simply satisfy all literals in a particular clause C_i to make it true. A harder problem is computing $N(F)$ — the number of satisfying assignments to F. This problem is called #DNF. Note that this problem is $\#P$-complete where $\#P$ is the counting analog of NP. This implies that it is unlikely to solve this problem exactly. We will provide a Fully Polynomial Randomized Approximation Scheme (FPRAS) to solve #DNF. In particular, the FPRAS takes as input a DNF F, $\epsilon > 0$, and $\delta > 0$ and outputs OUT s.t. $\Pr(|OUT - N(F)| \leq \epsilon N(F)) \geq 1 - \delta$. The running time is $poly(n, m, 1/\epsilon, \log(1/\delta))$. Our FPRAS will make use of the Monte Carlo method.

19.1.2 The Monte Carlo Method

The Monte Carlo method estimates $|S|$ for a given set S. This approach involves choosing a trivial set Ω s.t. $S \subseteq \Omega$ and $|\Omega|$ is known. We generate i.i.d. samples $X_1, \ldots, X_t$ from Ω. Then we construct indicator random variables Y_i for $i = 1, \ldots, t$ as follows:

$$Y_i = \begin{cases} 1 & \text{if } X_i \in S \\ 0 & \text{if } X_i \notin S \end{cases}$$

Note that $\mu = \mathbb{E}[Y_i] = \Pr(X_i \in S) = \dfrac{|S|}{|\Omega|}$. Let $Y = \dfrac{1}{t}\sum_{i=1}^{t} Y_i$. Then $\mathbb{E}[Y] = \mu$. We output an estimator $\hat{Y} = |\Omega| Y$. So, $\mathbb{E}[\hat{Y}] = |\Omega|\mu = |S|$. We would like that, for inputs $\epsilon, \delta > 0$, $\Pr(|\hat{Y} - |S|| \leq \epsilon|S|) \geq 1 - \delta \Rightarrow \Pr(|\hat{Y} - |S|| > \epsilon|S|) \leq \delta$. Using Chernoff bounds, we have the following:

$$\begin{aligned}
\Pr(|\hat{Y} - |S|| > \epsilon|S|) &= \Pr\left(\left|\frac{|\Omega|}{t}\sum_{i=1}^{t} Y_i - |\Omega|\mu\right| \geq \epsilon|\Omega|\mu\right) \\
&= \Pr\left(\left|\sum_{i=1}^{t} Y_i - t\mu\right| \geq \epsilon t\mu\right) \\
&\leq 2e^{-\epsilon^2 t\mu/3}
\end{aligned}$$

Therefore, we have that $t \geq \frac{3}{\mu\epsilon^2} \ln\left(\frac{2}{\delta}\right)$. Observe that we need $\mu = \Omega\left(\frac{1}{poly(x)}\right)$ where x is the input size otherwise t is huge. That is, if $|S| << |\Omega|$, then this is a bad scheme.

19.1.3 Applying the Monte Carlo Method to #DNF

We can adapt the approach outlined above to the problem of counting satisfying assignments to a DNF formula F containing m clauses and n literals. First, we set $S = N(F)$. An obvious choice for Ω is the set of all possible assignments of the n literals. So, $|S| = |N(f)|$ and $|\Omega| = 2^n$. We can generate assignments $\sigma_1, \ldots, \sigma_t$ uniformly at random from Ω. Now, we can construct indicator random variables Y_i for $i = 1, \ldots, t$ as follows:

$$Y_i = \begin{cases} 1 & \text{if } \sigma_i \text{ satisfies } F \\ 0 & \text{if not} \end{cases}$$

Note that $\mathbb{E}[Y_i] = \Pr(\sigma_i \text{ satisfies } F) = \frac{N(F)}{2^n}$. Letting $Y = \frac{1}{t}\sum_{i=1}^{t} Y_i$, we have that $\mathbb{E}[Y] = \frac{N(F)}{2^n}$. We output an estimator $\hat{Y} = 2^n Y$. So, $\mathbb{E}[\hat{Y}] = N(F)$. From our running time analysis in section 19.1.2, we can see that if $\frac{N(F)}{2^n} \geq \frac{1}{poly(n)}$, then we have an FPRAS. However, if $N(F) << 2^n$, then this is a bad scheme. Thus, we will need to come up with a better choice for our sample space Ω.

19.1.4 Choosing a Better Sample Space

Instead, we consider the following multi-set as our sample space

$$\Omega = \sum_{i=1}^{m} S_i$$

where S_i is the set of assignments which satisfy clause i. Note that we can easily calculate the size of S_i — if there are j variables in clause i, then the number of satisfying assignments for clause i is 2^{n-j}. We also note that the set of all satisfying assignments of f is $S = \bigcup_{i=1}^{m} S_i$. Therefore,

$$|\bigcup_{i=1}^{m} S_i| \leq \sum_{i=1}^{m} |S_i| \leq m|S|$$

where the last inequality follows since each assignment can satisfy up to m clauses. With the above, we have proven that

$$|S| \leq |\Omega| \leq m|S|$$

which shows us that we have a good sample space.

Next, we understand how to sample from the multi-set Ω. We relabel the elements of Ω to instead be tuples

$$\Omega := \{(i, \sigma) : \sigma \in S_i, 1 \leq i \leq m\}.$$

We sample from Ω by first sampling i with probability proportional to how many satisfying assignments satisfy clause i, and then by sampling the actual assignment. In particular, we choose i with probability $\frac{|S_i|}{\sum_{j=1}^{m} |S_j|} = \frac{|S_i|}{|\Omega|}$. To sample the actual assignment, we simply decide the value of each literal not in clause i independently and uniformly at random. So, we pick the assignment with probability $\frac{1}{|S_i|}$. Therefore, we have that for a fixed (i, σ) tuple:

$$\Pr(\text{Picking } (i, \sigma)) = \frac{|S_i|}{|\Omega|} \frac{1}{|S_i|} = \frac{1}{|\Omega|}$$

i.e. uniform over Ω. Using this sampling technique, we generate t i.i.d samples $(i_1, \sigma_1), \ldots, (i_t, \sigma_t)$ from Ω. Let V be the set of all tuples (i, σ_i) s.t. clause i is the first clause satisfied by σ_i whenever σ_i is a satisfying assignment to F. Note that $|V| = |S| = N(F)$ since any $\sigma \in S$ satisfies a least indexed clause l which implies that $(\sigma, l) \in V$, and for any $(\sigma, l) \in V$, $\sigma \in S$ since it satisfies some clause l in F satisfies F. Now, we construct indicator random variables Y_j for $j = 1, \ldots, t$ as follows:

$$Y_j = \begin{cases} 1 & (j, \sigma_j) \in V \\ 0 & (j, \sigma_j) \notin V \end{cases}$$

So, we have the following:

$$\mu = E[Y_j] = \frac{|V|}{|\Omega|} = \frac{N(F)}{|\Omega|} = \frac{|S|}{|\Omega|} \geq \frac{1}{m}$$

Let $Y = \dfrac{1}{t}\displaystyle\sum_{i=1}^{t} Y_i$. Then $\mathbb{E}[Y] = \dfrac{|S|}{|\Omega|}$. We output an estimator $\hat{Y} = |\Omega| Y$ which implies that $\mathbb{E}[\hat{Y}] = |S| = N(F)$. Finally, from the running time analysis in section 19.1.2, we have the following:

$$\frac{3m}{\epsilon^2} \ln\left(\frac{2}{\delta}\right) \geq \frac{3}{\mu\epsilon^2} \ln\left(\frac{2}{\delta}\right)$$

By setting $t \geq \dfrac{3m}{\epsilon^2} \ln\left(\dfrac{2}{\delta}\right)$, we satisfy that $\Pr\left(|\hat{Y} - N(F)| \leq \epsilon N(F)\right) \geq 1 - \delta$. We also have that t is polynomial in m, $\dfrac{1}{\epsilon}$, and $\ln\left(\dfrac{1}{\delta}\right)$. Since our sampling step takes polynomial time as well, we have an FPRAS for #DNF.

19.2 The Network Unreliability Problem

The **Network Unreliability Problem** is as follows: we have an undirected graph $G = (V, E)$ and some parameter $0 \leq p \leq 1$. We run through all edges of the graph and delete each edge with probability p. Let H denote the resulting sub-graph. We define

$$FAIL_G(p) = \Pr(H \text{ is disconnected})$$

We want to create an FPRAS to find $FAIL_G(p)$.

We could use a trivial scheme: run the experiment multiple times and check whether the graph is disconnected. This is the same as performing a Monte Carlo simulation. Let $\mu = FAIL_G(p)$. If we run the experiment $t \geq \mathcal{O}(\frac{1}{\epsilon^2\mu} \log \frac{1}{\delta})$ and we say

$$Y_i = \begin{cases} 1 & \text{if } H_i \text{ is disconnected} \\ 0 & \text{if not} \end{cases}$$

Then, $\mathbb{E}(Y_i) = \mu$. Let the size of the min cut of G be c. All edges of this cut vanish (and therefore leave the graph disconnected) with probability p^c. Therefore, if $\mu > p^c > n^{-4}$, then we can run the trivial scheme and get a running time of $\mathcal{O}(\frac{n^4}{\epsilon^2} \log \frac{1}{\delta})$ [Karger]. However, if $p^c < n^{-4}$, we will have to use something other than the trivial scheme.

Recall that when deriving Karger's algorithm, the probability of finding a single min cut was $\frac{1}{n^2}$ which in turn implied there were less than n^2 min cuts in any given graph G. Similarly, if we consider the number of cuts of size αc where $\alpha \geq 1$, then

$$\Pr\left(\text{cut of size at most } \alpha c \text{ is found}\right) < \frac{1}{n^{2\alpha}}$$

since we can just run Karger's algorithm down to 2α vertices. Therefore, the number of cuts of size at most αc is $n^{2\alpha}$.

Intuitively, when calculating the probability H is disconnected, the size of "large" cuts do not matter. If we let $\alpha = 2 + \ln(\frac{2}{\epsilon})$, then we say that cuts with size greater than αc do not matter (theorem 2.9 and 2.10 [Karger]). We enumerate all cuts with size less than or equal to αc and we say H is disconnected if at least one of these cuts is satisfied.

At this point, we apply #DNF. We construct a DNF formula f where the edges are the variables x_i and the cuts are the clauses. Then,

$$x_i = \begin{cases} 1 & \text{with probability } p \\ 0 & \text{with probability } 1-p \end{cases}$$

which would mean

$$\Pr\left(f \text{ is satisfied}\right) \leq FAIL_G(p)$$

(the value is within $\frac{\epsilon}{2}$ of $FAIL_G(p)$). Note that the probability that f is satisfied is

$$\sum_{\sigma:\sigma \text{is a sat. assign}} p^{pos(\sigma)}(1-p)^{neg(\sigma)}$$

where $pos(\sigma)$ and $neg(\sigma)$ indicate the number of clauses in f which are satisfied and the number of clauses in f which are not, respectively. This can be calculated using a variant of the #DNF scheme in the previous section. Since our #DNF scheme runs in polynomial time, this clever scheme also runs in polynomial time.

In summary, our FPRAS for estimating $FAIL_G(p)$ is as follows: find c, the size of the minimum cut of G. If $p^c > n^{-4}$, then run the experiment $\mathcal{O}(\frac{1}{\epsilon^2 \mu} \log \frac{1}{\delta})$ times and calculate the proportion of times in which the resulting sub-graph H is disconnected. Else, use Karger's algorithm to enumerate all cuts of size at most αc and construct a corresponding DNF formula. The probability the DNF formula is satisfied is a polynomial time close estimate for $FAIL_G(p)$ according to section 19.1.

References

[1] D. R. Karger. A randomized fully polynomial time approximation scheme for the all-terminal network reliability problem. SIAM J. Comput., 29(2):492514, 1999.