CSE 632: Analysis of Algorithms II: Randomized Algorithms **Spring 2024**

Lecture 17: #DNF

Lecturer: Zongchen Chen

1 #DNF and FPRAS

A Boolean formula f is in disjunctive normal form (DNF) if it is an OR of ANDs, e.g.,

$$f = (x_1 \wedge \neg x_3 \wedge x_4) \vee (\neg x_2 \wedge x_3) \vee (x_2).$$

It is easy to find a satisfying assignment for a formula in DNF: Take any clause, and find an assignment that satisfies all literals in that clause. For example, $(x_1, x_2, x_3, x_4) = (\mathsf{T}, \mathsf{T}, \mathsf{F}, \mathsf{T})$ is a satisfying assignment to the formula f above where the first and third clauses are satisfied.

Given a Boolean formula f in DNF with n variables $x_1, \dots, x_n$ and m clauses $c_1, \dots, c_m$, our goal is to find

$$N(f) := \#\text{ of satisfying assignments of } f.$$

Exact counting for #DNF is #P-complete, which is conjectured to be unsolvable in $\text{poly}(n, m)$ time. We want to have an approximate counting algorithm for #DNF.

Given a Boolean formula f in DNF and an approximation parameter $\varepsilon \in (0, 1)$ as inputs, a randomized algorithm is called a *fully polynomial-time randomized approximation scheme* (FPRAS) if it outputs $\hat{N}$ satisfying

$$\Pr\left((1-\varepsilon)\hat{N} \le N(f) \le (1+\varepsilon)\hat{N}\right) \ge \frac{3}{4}$$

in time $\text{poly}(n, m, 1/\varepsilon)$.

Remark 1.
1. An FPRAS gives the strongest form of approximation. It can achieve $1 \pm \varepsilon$ approximation for any $\varepsilon \in (0, 1)$, but the running time depends (inverse polynomially) on ε.
2. The running time of an FPRAS is fully polynomial in $n, m, 1/\varepsilon$. Note, $n^3 m^{2/\varepsilon}, e^{1/\varepsilon} n^2 m^3, (nm)^{\log(1/\varepsilon)} = (1/\varepsilon)^{\log(nm)}$ are *not* fully polynomial.
3. We can boost the success probability of an FPRAS to $1-\delta$ for any (input) failure probability $\delta \in (0, 1)$. This can be achieved by running $O(\log(1/\delta))$ trials of the FPRAS, and taking the median of all outputs (same analysis as the "median of means" approach by Chernoff bounds).

2 Monte Carlo Method

Let Ω be a finite universe with "simple" structure such that we know $|\Omega|$ and can generate random elements from Ω. Let $S \subseteq \Omega$ be a subset for which we want to estimate the size $|S|$. For the #DNF problem, we may define Ω to be the set of all truth assignments, so that $|\Omega| = 2^n$ and we can easily sample an assignment from Ω uniformly at random; and we define $S \subseteq \Omega$ to be the set of satisfying assignments, so $N(f) = |S|$. To estimate $|S|$, the Monte Carlo algorithm generates a few independent samples from Ω uniformly at random and finds the fraction of samples that lie in S which approximates $|S|/|\Omega|$.

Let $\alpha = |S|/|\Omega|$, and observe that $\mathbb{E}[Y_i] = \Pr(X_i \in S) = |S|/|\Omega| = \alpha$ for each i. So, we have $\mathbb{E}[\hat{Y}] = \alpha$ and $\mathbb{E}[\hat{N}] = |\Omega|\mathbb{E}[\hat{Y}] = \alpha|\Omega| = |S|$. This shows that $\hat{N}$ is an unbiased estimator. By the Chernoff bounds,

Algorithm 1 General Monte Carlo Algorithm

1: Generate t independent and uniformly random samples from Ω, denoted by $X_1, \ldots, X_t$
2: **for** $i = 1$ to t **do**
3: $\quad Y_i \leftarrow \begin{cases} 1, & \text{if } x_i \in S \\ 0, & \text{o/w} \end{cases}$
4: **end for**
5: $\hat{Y} \leftarrow \frac{1}{t}\sum_{i=1}^{t} Y_i$ ▷ Fraction of samples in S
return $\hat{N} = |\Omega|\hat{Y}$

we deduce that

$$\begin{aligned}
\Pr\left(\left|\hat{N} - |S|\right| \geq \varepsilon|S|\right) &= \Pr\left(\left|\frac{|\Omega|}{t}\sum_{i=1}^{t} Y_i - |S|\right| \geq \varepsilon|S|\right) \\
&= \Pr\left(\left|\sum_{i=1}^{t} Y_i - t\alpha\right| \geq \varepsilon(t\alpha)\right) \\
&\leq 2e^{-\varepsilon^2(t\alpha)/3} && \text{(Chernoff Bounds)} \\
&\leq \frac{1}{4},
\end{aligned}$$

where the last inequality holds if we set $t = \left\lceil \frac{9}{\alpha\varepsilon^2} \right\rceil$. Hence, we have $t = \mathrm{poly}(n, m)$ when $\alpha = \Omega(\frac{1}{\mathrm{poly}(n,m)})$.

For the #DNF problem, we have $\alpha = \frac{N(f)}{2^n}$. Thus, a straightforward application of the Monte Carlo method Algorithm 1 does not give a polynomial-time algorithm when $N(f) \ll 2^n$, e.g., $N(f) \approx 2^{0.99n}$.

3 Better Monte Carlo Method for #DNF

For each clause c_i, let S_i be the set of assignments that satisfy c_i. Let $S = \bigcup_{i=1}^{m} S_i$ be the set of all satisfying assignments. Note that $N(f) = |S|$.

We shall pick the universe Ω in a smarter way to apply Algorithm 1. Let Ω be the multiset union of S_i's; specifically,

$$\Omega = \{(i, \sigma) \in [m] \times S : \sigma \in S_i\}.$$

The set S is technically not a subset of the universe Ω, but we can easily find a bijective mapping from S to a subset $S' \subseteq \Omega$ defined as

$$S' = \{(i, \sigma) \in [m] \times S : \sigma \notin S_j \text{ for } j = 1, \ldots, i-1 \text{ and } \sigma \in S_i\}.$$

In other words, S' consists of all pairs (i, σ) where c_i is the first clause satisfied by σ.

Observation 2.
1. $|S'| = |S| = N(f)$;
2. $S' \subseteq \Omega \subseteq [m] \times S$ and so $|S'| \leq |\Omega| \leq m|S'|$;
3. For each i, we can easily sample u.a.r. from S_i by satisfying all the literals in c_i and choosing a uniformly random assignment for the remaining variables;
4. For each i, if $|c_i| = k_i$ (i.e., the clause c_i contains k_i literals), then $|S_i| = 2^{n-k_i}$.

From the observations above, we can compute

$$|\Omega| = \sum_{i=1}^{m} |S_i| = \sum_{i=1}^{m} 2^{n-k_i},$$

and we can sample uniformly random elements from Ω by first sampling $i \in [m]$ with probability proportional to $|S_i|$, and then sampling σ u.a.r. from S_i. Algorithm 1 is then specified to the following version.

Algorithm 2 Monte Carlo Algorithm for #DNF

---

for $j = 1$ to t **do**

  Pick $i \in [m]$ with probability $\propto |S_i| = 2^{n-k_i}$; that is,

$$\Pr(\text{pick } i) = \frac{|S_i|}{|\Omega|} = \frac{2^{n-k_i}}{\sum_{i'=1}^{m} 2^{n-k_{i'}}}$$

  Sample σ u.a.r. from S_i ▷ Note, $\Pr(\text{pick } (i,\sigma)) = \frac{|S_i|}{|\Omega|} \cdot \frac{1}{|S_i|} = \frac{1}{|\Omega|}$

  $Y_j \leftarrow \begin{cases} 1, & \text{if } c_i \text{ is the first clause satisfied by } \sigma \\ 0, & \text{o/w} \end{cases}$

end for

$\hat{Y} \leftarrow \frac{1}{t}\sum_{j=1}^{t} Y_j$

  return $\hat{N} = |\Omega|\hat{Y}$

Notice that

$$\mathbb{E}\left[\hat{Y}\right] = \alpha = \frac{|S'|}{|\Omega|} \geq \frac{1}{m}$$

by Observation 2. Therefore, if we set $t = \left\lceil \frac{9}{\alpha\varepsilon^2} \right\rceil = O\left(\frac{m}{\varepsilon^2}\right)$, then it holds

$$\Pr\left(\left|\hat{N} - N(f)\right| \geq \varepsilon N(f)\right) \leq \frac{1}{4}.$$