6.895 Randomness and Computation	March 8, 2006
Lecture 9	
Lecturer: Ronitt Rubinfeld	*Scribe: Kunal Agrawal*

1 Previous Lecture

We saw how to generate random satisfying assignments to a DNF formula. In this lecture, we see how to approximately count the number of satisfying assignments to a DNF formula.

2 Definitions

Definition 1 *$R \subseteq \{0,1\}^* \times \{0,1\}^*$ is **p-relation** if*

- $\forall (x,y) \in R, |y| = \text{poly}(|x|)$
- *$\exists$ p-time decision procedure for deciding $(x,y) \in R$*

Example: $R_{SAT} = \{(x,y)|x \text{ is a formula, } y \text{ is a satisfiable assignment}\}$

Definition 2 *$f : \{0,1\}^* \to \mathbb{N}$ ia in **#P** iff $\exists$ p-relation R s.t. $\forall x, f(x) = |\{y|(x,y) \in R\}|$*

In other words, a problem is in $\#P$ if and only if the number of witnesses for the problem are polynomial in the size of the problem. $\#P$-complete problems are the hardest problems in $\#P$. $\#CNF$, $\#DNF$ and $\#MATCHING$ are all $\#P$-complete problems.

Definition 3 *$\mathcal{A}$ is an **approximate counter** for f iff $\forall x, \Pr[f(x)/(1+\varepsilon) \leq \mathcal{A}(x) \leq f(x)(1+\varepsilon)] \geq 3/4$. If the runtime of $\mathcal{A}$ is $\text{poly}(|x|, 1/\varepsilon)$ then $\mathcal{A}$ is a "fully polynomial randomized approximation scheme" (FPRAS).*

Observation 4 *There is no FPRAS for $\#CNF$. If there were, then we could give a randomized algorithm to solve SAT in polynomial time.*

- *$SAT(x)$=yes $\implies \#CNF(x) > 0 \implies$ FPRAS$(x) > 0$, with probability $\geq 3/4$.*
- *$SAT(x)$=no $\implies \#CNF(x) = 0 \implies$ FPRAS$(x) = 0$, with probability $\geq 3/4$.*

Therefore, if we had an FPRAS for $\#CNF$, we could answer SAT correctly with probability at least 3/4. This implies that if there existed FPRAS for SAT, so we would know that $NP = RP$.

Definition 5 *R is a p-relation, $f : \{0,1\}^* \to \mathbb{N}$ s.t. $f(x) = |\{y|(x,y) \in R\}|$. M is a **uniform generator** for R if on any x*

- *M's runtime is $\text{poly}(|x|)$, and*
- *if $f(x) > 0$, then $\forall y$ s.t. $(x,y) \in R, \Pr[M \text{ outputs } y] = 1/f(x)$.*

Definition 6 *R is a p-reln, $f : \{0,1\}^* \to N$ s.t. $f(x) = |\{y|(x,y) \in R\}|$. M is a **almost uniform generator** for R if on any x, ε*

- *M's runtime $\text{poly}(|x|, 1/\varepsilon)$, and*
- *if $f(x) > 0$, then $\forall y$ such that $(x,y) \in R$, then $1/((1+\varepsilon)f(x)) \leq \Pr[M \text{ outputs } y] \leq (1+\varepsilon)/f(x)$.*

Here is the definition of self-reducibility given in the work of Jerrum, Valiant and Vazirani:

Definition 7 *A relation $R \subseteq \Sigma^* \times \Sigma^*$ is **self-reducible** iff*

1. *There exists a polynomial time computable function $g : \Sigma^* \to \mathcal{N}$ such that if xRy then $|y| = g(x)$.*

2. *There exists a polynomial time computable function $\psi : \Sigma^* \times \Sigma^* \to \Sigma^*$ and $\sigma : \Sigma^* \to \mathcal{N}$ satisfying*

   - *$\sigma(x) = O(\log |x|)$ for all $x \in \Sigma^*$*
   - *if $g(x) > 0$ then $\sigma(x) > 0$ for all $x \in \Sigma^*$*
   - *$|\psi(x, w)| \leq |x|$ for all $x, w \in \Sigma^*$*
   - *for all $x \in \Sigma^*$, $y = y_1, \ldots, y_n \in \Sigma^*$,*
     $< x, y_1, \ldots, y_n > \in R$ iff $< \psi(x, y_1, \ldots, y_{\sigma(x)}), y_{\sigma(x)+1}, \ldots, y_n > \in R$

The function g gives the length of the solutions to instances, and σ gives the granularity of solutions as follows: Given instance x and initial segment w of a witness y for x (i.e., $(x, y) \in R$), ψ gives an instance x' whose solutions are exactly those words which when concatenated with w form witnesses to x.

For the purposes of this lecture, self-reducibility means solving a problem by solving smaller problems. SAT is self-reducible. If we had an oracle P that solved a problem with $n-1$ variables, then you can use it to solve a problem with n variables, to solve $SAT(\Phi(x_1 \ldots x_n))$:

1. If $P(\Phi(0, x_2, \ldots x_n))$ = satisfiable, or $P(\Phi(1, x_2, \ldots, x_n))$ = satisfiable,

2. output satisfiable,

3. else output unsatisfiable.

3 FPRAS for $\#DNF$

Most graph and optimization problems are self-reducible. Today, we use this property to show prove a theorem of by Jerrum, Valiant and Vazirani.

Theorem 8 (Jerrum, Valiant, Vazirani) *If R is a self-reducible P-relation, there is an FPRAS for R iff there is a almost uniform generator (aug) for R.*

The theorem means that approximate counting is equivalent to approximate uniform generation. We are going to do the proof of SAT, and the only thing we use it is that SAT is self-reducible.

3.1 Reducing approximate uniform generation to approximate counting

For simplicity, first, we assume that we have an exact counter of $\#SAT$, and generate exact uniform generator. Then we relax this assumption to get an approximate generation from an approximate counter.

We use a ***self-reducibility tree*** and each node of this tree corresponds to the settings of a prefix[1]. The tree is shown in 3.1. At each node we get a formula, $\Phi_{b_1 \ldots b_j}(x_{j+1} \ldots x_n) = \Phi(b_1, \ldots b_j, x_{j+1} \ldots x_n)$. The leaves are the full assignments. If we had an oracle for counting the number of solutions on both the right and the left subtree of each node, then we can just do a biased random walk of the tree to generate a solution uniformly at random.

Let the current node be $(b_1 \ldots b_j)$,

[1]There was a question in class about what happens if two prefixes of a witness reduce to the same subproblem. This is not be a problem, since as long as the prefixes are different, the witness that would be output would be distinct at each leaf.

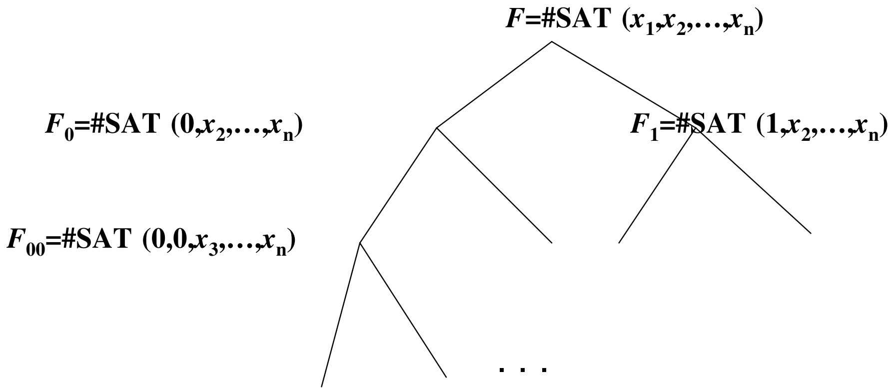

Figure 1: Self-Reducibility Tree

- Use the exact counter to compute $S_0 = F_{b_1..b_j,1}$ and $S_1 = F_{b_1..b_j,0}$
- Go left with probability $S_0/(S_0 + S_1)$ and right with probability $S_1/(S_0 + S_1)$.
- output the leaf you reach.

The output is correct because for any assignment $(b_1, ..., b_n) = b$ that the algorithm outputs,

- b is definitely a satisfying assignment.
- $$\Pr[\text{output } b] = \frac{F_{b_1}}{F}\frac{F_{b_1,b_2}}{F_{b_1}}\frac{F_{b_1,b_2,b_3}}{F_{b_1,b_3}}... = 1/F. \tag{1}$$

Now we prove that an approximate counter can be used in an approximate uniform generator using exactly the same algorithm. What if we were using an ε'-FPRAS for counting instead of an exact counter? The right hand side of the equation becomes: $(1/F)(1+\varepsilon')^n/(1-\varepsilon')^n \leq (1/F)(1/(1-\varepsilon)^{2n})$. Thus, if we want an ε-FPRAS for uniform generation, then it is sufficient to call the uniform counting ε'-FPRAS where $\varepsilon' < \varepsilon/2n$.

3.2 Reducing approximate counting to approximate uniform generation

Again, first, let us assume that we have a perfect uniform generator for R_{SAT}, and try to get an approximate counter. We will then relax the assumption and use the approximate generator. We look at the self-reducibility tree, and let $S_1 = F_1/F$. We are interested in computing the number of solutions to the formula, which is $F = F_1/S_1$. S_1 can be computed by sampling, that is, generating some uniformly distributed satisfying assignments (using our oracle for uniform genetation) and taking the ratio of the ones where $x_1 = 1$ and those with $x_1 = 0$. F_1 can be computed recursively.

Since we can only generate approximately uniform satisfying assignments, we can compute F_1 and S_1 approximately. Approximations are represented with hats on the variables.

1. For $b \in \{0, 1\}$. Let $\hat{S}_b$ = Number of samples beginning with b/total number of samples
2. Now recursively estimate F_b by $\hat{F}_b$.
3. Estimate F via $\frac{\hat{F}_b}{\hat{S}_b}$.

Unrolling the recursion gives us:

$$\begin{aligned}
\hat{F} &= \frac{\hat{F}_{b_1}}{\hat{S}_{b_1}} \\
&= \frac{\hat{F}_{b_1 b_2}}{\hat{S}_{b_1}\hat{S}_{b_1 b_2}} \\
&= \ldots \\
&= \frac{1}{\hat{S}_{b_1}\hat{S}_{b_1 b_2}\ldots\hat{S}_{b_1 b_2 \ldots b_n}}
\end{aligned}$$

If we had exactly uniform generators and no sampling errors, then the recursion can go either left or right, as long as the subtree had some satisfying assignment. But since we only have approximate generators, we recurse on the subtree which has more satisfying assignments. We need to estimate each $\hat{S}_{b_1 \ldots b_j}$ to within $\varepsilon/2n$ factor. Which gives $\hat{F} \leq (1+\varepsilon)F$ and $\hat{F} \geq 1/(1+\varepsilon)F$.

There is a slight technical problem with using Chernoff bounds in this case, which forces us to recurse on the side which we think has more satisfying assignments. The Chernoff bound says that:

$$\Pr[\hat{S}_b > (1+\delta)] \leq e^{-\delta^2 S_b m/3}$$

where m is the number of samples. When S_b is very small, then the number of samples needed to estimate the $\hat{S}_b$ upto reasonable accuracy gets bigger. Therefore, we need to go down the side with more satisfying assignments.