6.5420 Randomness and Computation March 17, 2026

Lecture 11

Lecturer: Ronitt Rubinfeld *Scribe: Nipun Dour*

In this lecture, we study approximate counting and its connection to uniform generation. We begin with the definition of an FPRAS and the observation that an FPRAS for #SAT would give a randomized polynomial-time algorithm for SAT. We then compare exact counting to approximate counting on some natural problems, and use the DNF case to show how uniform generation together with downward self-reducibility leads to an approximate counting algorithm. Finally, we prove the reverse case while introducing the Jerrum-Valiant-Vazirani theorem.

1 Approximate Counting

Definition 1 (FPRAS)**.** *Given formula ϕ, $\varepsilon > 0$, let $z = \#\phi$ be the number of satisfying assignments to ϕ. A fully polynomial randomized approximation scheme* (FPRAS) *outputs a value y such that*

$$\frac{z}{1+\varepsilon} \leq y \leq z(1+\varepsilon)$$

with probability at least $3/4 = \delta$ (sometimes LHS is $z(1-\varepsilon)$).

Hope: runtime polynomial in $|\phi|$ and $1/\varepsilon$ In Problem 1 of Pset 1, algorithm that satisfies this

$$\text{poly}\left(|\phi|, \frac{1}{\varepsilon}, \log\frac{1}{\delta}\right),$$

so the failure probability can be exponentially small while keeping polynomial runtime.

Proposition 2. *If there is an* FRPAS *for* #SAT*, then there is a randomized polynomial-time algorithm for* SAT*.*

Proof. Given a formula ϕ, call the FPRAS on ϕ with any fixed $\varepsilon > 0$. Let the output be y.

- If $y > 0$, output Satisfiable.
- If $y = 0$, output Unsatisfiable.

If ϕ is satisfiable, then $z = \#\phi \geq 1$, so

$$y \geq \frac{1}{1+\varepsilon} > 0.$$

Hence the algorithm outputs Satisfiable.

If ϕ is unsatisfiable, then $z = \#\phi = 0$, so the approximation guarantee forces

$$0 \leq y \leq 0(1+\varepsilon),$$

and therefore $y = 0$. Hence the algorithm outputs Unsatisfiable. □

2 Exact Versus Approximate Counting

Below are the complexity of exact counting and approximate counting on several natural problems.

Problem	Decision/Existence	Exact counting	Approximate counting
CNF satisfiability	NP-complete	#P-complete	NP-hard
DNF satisfiability	P	#P-complete	P
Perfect matchings	P	#P-complete	P
Spanning trees	P	polynomial time	P

Remark. Note that there are no simple global trends here, for example, it took decades of work to prove approximate counting in perfect matching is P.

3 Approximate Counting for #DNF

Our polynomial-time approximate counting algorithm uses two components:

1. a uniform generator for satisfying assignments of a DNF, and
2. downward self-reducibility.

3.1 Downward self-reducibility

Definition 3 (dsr)**.** *A counting problem is* downward self-reducible *if one can compute the solution by solving smaller subproblems and putting together the answers in polynomial-time computation.*

Proposition 4. #DNF *is downward self-reducible.*

Proof.

$$\#\phi(x_1,\ldots,x_n) = \#\phi(x_1 = \mathrm{F},\ldots,x_n) + \#\phi(x_1 = \mathrm{T},\ldots,x_n)$$

Each of the two subproblems is a #DNF subproblem, now on $n-1$ variables. □

Example 5. *Consider*

$$\#(x_1\bar{x}_2 \vee x_1x_2 \vee \bar{x}_2) = \#(\bar{x}_2) + \#(\bar{x}_2 \vee x_2 \vee \bar{x}_2) = 1 + 2 = 3.$$

where the first term is the $x_1 = \mathrm{F}$ case and the second term is the $x_1 = \mathrm{T}$ case.

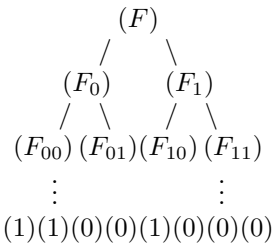
3.2 DSR tree

We can organize this recursion as a binary tree. Write

$$F \triangleq \#\phi(x_1,\ldots,x_n), F_0 \triangleq \#\phi(x_1 = \mathrm{F}, x_2,\ldots,x_n), F_1 \triangleq \#\phi(x_1 = \mathrm{T}, x_2,\ldots,x_n),$$

$$F_{00} \triangleq \#\phi(x_1 = \mathrm{F}, x_2 = \mathrm{F},\ldots,x_n), F_{01} \triangleq \#\phi(x_1 = \mathrm{F}, x_2 = \mathrm{T},\ldots,x_n),\ldots$$

Then every node F_b splits as the sum of its two children $F_b = F_{b0} + F_{b1}$.



At depth n, there are no variables left in the formula, so each leaf is either 1=T or 0=F.

Example 6. *A visualization of the previous example in a DSR tree is*

$$\begin{array}{ccc} & \#(x_1\bar{x}_2 \vee x_1x_2 \vee \bar{x}_2) = 1+2=3 & \\ & \swarrow \quad \searrow & \\ \#(\bar{x}_2) = 1+0=1 & & \#(\bar{x}_2 \vee x_2 \vee \bar{x}_2) = 1+1=2 \\ \swarrow \quad \searrow & & \swarrow \quad \searrow \\ \#\mathrm{T}=1 \quad \#\mathrm{F}=0 & & \#\mathrm{T}=1 \quad \#\mathrm{T}=1 \end{array}$$

3.3 Recursive estimation idea

The algorithm estimates the fraction of satisfying assignments that move to a chosen child. Define S_1 to be the fraction of satisfying assignments with $x_1 = \mathrm{T}$.

$$S_1 \triangleq \frac{F_1}{F} \Rightarrow F = \frac{F_1}{S_1}$$

More generally, if we follow a path $b_1, b_2, \ldots, b_n$, we define

$$S_{b_1\cdots b_i} \stackrel{\text{def}}{=} \frac{F_{b_1\cdots b_i}}{F_{b_1\cdots b_{i-1}}}.$$

Main Insight: For DNF, we can estimate S_1 via sampling We first uniformly generate k satisfying assignments. Then compute $\tilde{S}_1 \leftarrow (\#\text{ with } x_1 = \mathrm{T})/k$. Note we don't need perfect uniform generation for this sampling estimate, but we can for DNF from previous lecture.

By recursing on the subproblem,

$$F = \frac{F_{b_1}}{S_{b_1}} = \frac{F_{b_1b_2}}{S_{b_1}S_{b_1b_2}} \cdots = \frac{1}{\prod_{i=1}^n S_{b_1\cdots b_i}}.$$

3.4 Potential difficulties and larger-child

1. If $F_{b_1\cdots b_i} = 0$, this doesn't work.
2. Is the approximation of $S_{b_1\cdots b_i}$s good enough? (We only get additive estimates from sampling.)

Claim 7. *If we always choose $b_i \in \{0,1\}$ such that*

$$F_{b_1\cdots b_i} \geq F_{b_1\cdots b_{i-1}\bar{b}_i}.$$

Then the path always reaches a satisfying leaf.

Idea: Estimate each $S_{b_1\cdots b_i}$ to within $\frac{\varepsilon}{8n}$ additive error Using Chernoff bounds, we need only poly$(\frac{2n}{\varepsilon}, \log 8n)$ samples to get probability of error $< \frac{1}{4n}$, union bound over i to get $< \frac{1}{4}$. If $1 \geq r \geq \frac{1}{2}$, we can bound multiplicative error

$$r + \frac{\varepsilon}{8n} = r\Big(1 + \frac{\varepsilon}{8nr}\Big) \leq r\Big(1 + \frac{\varepsilon}{4n}\Big)$$
$$r - \frac{\varepsilon}{8n} = r\Big(1 - \frac{\varepsilon}{8nr}\Big) \geq r\Big(1 - \frac{\varepsilon}{4n}\Big)$$

Claim 8. *Using larger child, $S_i \geq \frac{1}{2}$, thus $\frac{F}{1+\varepsilon} \leq$ output $\leq F(1+\varepsilon)$.*

Proof. Since $S \geq \frac{1}{2}$, $S(1 - \frac{\varepsilon}{4n}) \leq \tilde{S} \leq S(1 + \frac{\varepsilon}{4n})$, so

$$\text{output} = \frac{1}{\prod_{i=1}^n \tilde{S}_{b_1\cdots b_i}} \leq \frac{F}{(1-\frac{\varepsilon}{4n})^n} \leq \frac{F}{(1-\frac{\varepsilon}{4})} \leq F(1+\varepsilon)$$
$$\text{output} = \frac{1}{\prod_{i=1}^n \tilde{S}_{b_1\cdots b_i}} \geq \frac{F}{(1+\frac{\varepsilon}{4n})^n} \geq \frac{F}{1+\varepsilon}$$

□

3.5 Recursive Algorithm

At each recursion level, estimate S_0 and S_1 from uniformly generating k satisfying assignments, let $b_1 \leftarrow \text{argmax}\{S_0, S_1\}$, and recurse on F_{b_1}.

Runtime

$$\underbrace{n}_{\#\text{ recursions}} \cdot \underbrace{\mathsf{poly}\left(\tfrac{\varepsilon}{4n}\right)^{-1} \cdot \left(\tfrac{1}{4n}\right)^{-1}}_{\#\text{ samples for } \frac{\varepsilon}{4n} \text{ additive error}} \cdot \underbrace{\mathsf{poly}(n)}_{\text{uniform generator}} = \mathsf{poly}\left(n, \frac{1}{\varepsilon}\right).$$

$$\Pr[\text{algorithm fails}] \leq \sum_{i=1}^{n} \Pr[\text{estimate at level } i \text{ is bad}] \leq n \cdot \frac{1}{4n} = \frac{1}{4}$$

4 The JVV Theorem: Uniform Generation and Counting

The approximate counting algorithm above works for *any* downward self-reducible problem as a poly-time (almost-)uniform-generation of solutions $\Rightarrow$ poly-time approximate counting of # solutions.

Theorem 9 (Jerrum–Valiant–Vazirani)**.** *For any problem in* NP *that is downward self-reducible,*

poly-time approx counting of # solutions $\iff$ *poly-time almost-uniform generation.*

We first work on the easier case.

Proposition 10. *(Perfect) counting for* #DNF *implies (perfect) uniform generation for* #DNF*.*

Proof. Recursive algorithm: at node $b_1 \dots b_i$ in the DSR tree, use the (perfect) counter to compute

$$r_0 = F_{b_1 \dots b_i 0}, \qquad r_1 = F_{b_1 \dots b_i 1}.$$

Go left with probability $\frac{r_0}{r_0 + r_1}$, and right otherwise.

Claim 11. *We always reach a satisfying leaf, since we never take a branch with* 0 *satisfying assignments underneath it.*

Claim 12. *For every satisfying assignment* $b_1 \cdots b_n$*,*

$$\Pr[\textit{output } b_1 \cdots b_n] = \frac{F_{b_1}}{F} \cdot \frac{F_{b_1 b_2}}{F_{b_1}} \cdots \frac{1}{F_{b_1 \cdots b_n}} = \frac{1}{F},$$

hence the output is uniformly distributed. □

4.1 What if we only have an approximate counter?

$$\Pr[\text{output } b_1 \cdots b_n] \leq \frac{1}{F}\left(\frac{1+\varepsilon'}{1-\varepsilon'}\right)^n \leq \frac{1}{F} \cdot \frac{1}{1-\varepsilon},$$

if we choose $\varepsilon' \leq \frac{\varepsilon}{2n}$. This is close to a uniform generation of satisfying assignments.