6.841: Advanced Complexity Theory Fall 2012

Lecture 22— November 29, 2012

Prof. Dana Moshkovitz *Scribe: Liz Simon*

1 Overview

In this lecture we begin talking about counting. An example of a counting problem would be to find the number of satisfying assignments a formula has. First, we will discuss $UniqueSAT$ and the Valiant-Vazirani Theorem, and then $\oplus SAT$. Then we will show that approximate counting is in BPP^{NP}.

Next time we will cover Toda's Theorem which says that $PH \subseteq P^{\#P}$, which shows that the power to count exactly is enough to solve any problem in the polynomial hierarchy.

2 Main Section

2.1 Unique-SAT

Definition: $UniqueSAT$ is a promise problem, meaning that we are given a guarantee on the input, which distinguishes between the following two classes:

- $U_{YES} = \{\phi \mid \phi \text{ has 1 satisfying assignment }\}$
- $U_{NO} = \{\phi \mid \phi \text{ has 0 satisfying assignments }\}$

The $UniqueSAT$ problem is: Given $\phi \in U_{YES} \cup U_{NO}$, distinguish between $\phi \in U_{YES}$ and $\phi \in U_{NO}$. If ϕ has more than one satisfying assignment, there is no guarantee on the output.

2.2 Valiant-Vazirani Theorem

This theorem was proved by Valiant and Vazirani in 1986[1]. The theorem states that there is a randomized reduction from SAT to $UniqueSAT$. The reduction is as follows:

- $\phi \in SAT \implies R(\phi) \in U_{YES}$ with probability $\geq \frac{1}{p(n)}$, where $p(n)$ is a polynomial in n
- $\phi \notin SAT \implies R(\phi) \in U_{NO}$ always

Notice that this does not seem very good since $\frac{1}{p(n)}$ is very small, but this result is very useful for many other results in complexity. Any improvement on this result would cause an improvement in many other results as well.

2.2.1 Proof of Valiant-Vazirani Theorem

The idea of the proof is to construct a reduction in such a way that the new formula is the same as the old but includes an extra term which uniquely identifies one satisfying assignment. In order to uniquely identify one satisfying assignment, we will hash the assignments. So we get $R(\phi) = \phi \wedge (h(x) = \vec{0})$, where $h : \{0,1\}^n \to \{0,1\}^m$ is a randomly selected hash function. However, for this to work, we need the hash function to not have collisions, so that $h(x) = \vec{0}$ for only one satisfying assignment. We will pick $h(x)$ randomly from a family $\mathcal{H}$, to be discussed next.

Crucial Point: Choosing the correct size of the set to be hashed to, 2^m, is very important. If we choose m to be too large, nothing will be mapped to $\vec{0}$. If we choose an m that is too small, there will be collisions so multiple things might get mapped to $\vec{0}$.

We pick a value for m at random from $0, 1, \cdots, n-1$. This is where the $\frac{1}{p(n)}$ comes from because the probability that we chose the correct value for m is $\frac{1}{n}$.

Definition 1. *A pairwise independent hash family $\mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}^m\}$ has the property that $\forall x \neq y \in \{0,1\}^n, a, b \in \{0,1\}^m, \Pr(h(x) = a \wedge h(y) = b) = (\frac{1}{2^m})^2$.*

Examples of pairwise independent hash families:

1. $\mathcal{H}$ = all functions
2. $\mathcal{H} = \{Ax + b \mid A \in \{0,1\}^{m \times n}, b \in \{0,1\}^m\}$, where the matrix-vector multiplication is performed over $\mathbb{F}_2$.

Now suppose we chose m such that $2^i \leq \#$ satisfying assignments $< 2^{i+1}$, where $m = i + 2$, which will happen with probability $\sim \frac{1}{n}$. For all $x \in \{0,1\}^n$, define the event U_x to be that x is the unique satisfying assignment that h maps to $\vec{0}$ (h may map other strings to 0, but those cannot be satisfying assignments). Then

$$\Pr(U_x) \geq \Pr(h(x) = \vec{0}) - \sum_{y \neq x, \text{ is sat. assgn.}} \Pr(h(x) = \vec{0} \wedge h(y) = 0) \geq \frac{1}{2^m} - \frac{N}{2^{2m}} \geq \frac{1}{2^{m+1}},$$

where N is the number of satisfying assignments. We want to find $\Pr(\exists x \ U_x)$. Since these are disjoint events for each x,

$$\Pr(\exists x U_x) = \sum_{x \text{ is sat. assgn.}} \Pr(U_x) \geq \frac{N}{2^{m+1}} \geq \frac{1}{8}$$

. So with probability $\sim \frac{1}{8n}, \phi \in SAT \implies R(\phi) \in U_{YES}$. If $\phi \notin SAT, R(\phi) \in U_{NO}$ because there are no satisfying assignments.

No one knows how to get rid of the $\frac{1}{n}$ factor, but if we talk about $\oplus SAT$ instead of $UniqueSAT$ we can do much better.

2.3 Parity-SAT

Definition 2. $\oplus SAT = \{\phi \mid$ *The number of satisfying assignments to ϕ is even* $\}$.

2.3.1 Valiant-Vazirani for $\oplus SAT$

We can modify the Valiant-Vazirani reduction above to achieve the following for $\oplus SAT$:

- $\phi \in SAT \implies R'(\phi) \in \oplus SAT$ with probability $\geq \Omega(\frac{1}{n})$
- $\phi \notin SAT \implies R'(\phi) \notin \oplus SAT$ always

2.3.2 Proof of Valiant-Vazirani for $\oplus SAT$

The reduction is as follows: given a formula ϕ, we first perform the $UniqueSAT$ reduction $\phi \mapsto R(\phi)$, which with probability $\Omega(\frac{1}{n})$ will give a formula that's uniquely satisfiable (if ϕ was satisfiable to begin with). Then, $R'(\phi)$ will be the formula $(y \wedge R(\phi)(x)) \vee (\overline{y} \wedge \vec{x} = \vec{0})$. Note that the satisfying assignments for $R'(\phi)$ are:

- $y = 1, \vec{x}$ satisfying for $R(\phi)$.
- $y = 0, \vec{x} = \vec{0}$

If $R(\phi)$ has one satisfying assignment, then $R'(\phi)$ has two satisfying assignments, and so $R'(\phi) \in \oplus SAT$. If $R(\phi)$ has no satisfying assignments, then $R'(\phi)$ has one satisfying assignment, and so $R'(\phi) \notin \oplus SAT$. Since these two are the only possibilities with probability $\Omega(\frac{1}{n})$, the theorem holds.

2.3.3 Amplified Valiant-Vazirani

We can improve the $\Omega(\frac{1}{n})$ probability for the $\oplus SAT$ reduction significantly:

- $\phi \in SAT \implies R''(\phi) \in \oplus SAT$ with probability $\geq 1 - \exp(-n)$
- $\phi \notin SAT \implies R'(\phi) \notin \oplus SAT$ always

2.3.4 Proof of Amplified Valiant-Vazirani

Let $k = n^2$. Invoke the reduction R' k times, each with independent randomness, and relabel the variables so that every output formula is using a new set of variables. Then we get $\phi_1, ...\phi_k$ on disjoint sets of variables. Even though R' has a low probability of succeeding, we only need it to succeed once, so invoking it many times will give us the result we want.

Define $\phi = \bigwedge_{i=1}^{k} \phi_i$. Then the number of satisfying assignments for ϕ is $\prod_{i-1}^{k}$(# satisfying assignments to ϕ_i). Notice that if even one term in the above product is even, the product will be even. So if the number of satisfying assignments for ϕ_i is odd for all i, then $\phi \notin \oplus SAT$. If for some i the number of satisfying assignments to ϕ_i is even, then $\phi \in \oplus SAT$.

The probability that $R''(\phi) \in \oplus SAT$ given that $\phi \in SAT$ is $1 - (1 - \Omega(\frac{1}{n}))^{n^2} \approx 1 - e^{-n}$, because each reduction attempt is independent of one another and $(1 - \frac{1}{n})^n \approx \frac{1}{e}$.

2.4 Approximate Counting

We'll show that given some boolean formula ϕ, distinguishing between the following two cases:

- The number of satisfying assignments for ϕ is $\geq 2^{k+1}$
- The number of satisfying assignments for ϕ is $\leq 2^k$

is in BPP^{NP}. This means that we can 2-approximate the number of satisfying assignments to a boolean formula in BPP^{NP}, because we can just try each possible value for k and then check if the number of satisfying assignments is less than or equal to 2^k until we find the one which gives us a 2-approximation.

Note: Any problem where you need to approximate the sum of nonnegative numbers will work this way. Once the numbers are allowed to be negative, it becomes much harder.

2.4.1 Algorithm for Approximate Counting

The idea of the algorithm is to hash the set of all assignments to a set of size less than 2^k. The NP oracle can give us satisfying assignments. The algorithm works as follows:

- Pick $h : \{0,1\}^n \to \{0,1\}^m$ at random from a pairwise independent hash family, with $m = k-5$. In the case that $k < 5$, just use brute force.
- Ask the prover for > 48 assignments $\vec{x}$ that satisfy ϕ and $h(\vec{x}) = \vec{0}$.

Notice that this algorithm is in BPP^{NP}. Now we just need to prove that the algorithm is correct.

2.4.2 Correctness of Algorithm

Lemma 3. *For any $S \subseteq \{0,1\}^n$, $|S| \geq \frac{4}{\epsilon^2} \cdot 2^m$,*

$$\Pr_{h \in \mathcal{H}} \left(\left| X - \frac{|S|}{2^m} \right| \geq \frac{\epsilon |S|}{2^m} \right) \leq \frac{1}{4},$$

where $X = \left| \{x \in S \mid h(x) = \vec{0}\} \right|$.

Proof. The idea is to bound the variance of X. Write $X = \sum_{x \in S} I_{h(x)=\vec{0}}$. We can bound the variance of this sum because $\mathcal{H}$ is a pairwise independent hash family, and the variance of a sum of pairwise independent variables is simply the sum of the variances. We get that

$$\mathrm{Var}(X) = |S| \left(\Pr(I_{h(x)=\vec{0}}) - \Pr(I_{h(x)=\vec{0}})^2 \right).$$

Then we can use Chebyshev's Inequality, $\Pr(|X - \mathbf{E}[X]| \geq \epsilon \mathbf{E}[X]) \geq \dfrac{\mathrm{Var}(X)}{\epsilon^2 \mathbf{E}[X]^2}$, and we get the stated lemma result. □

Now there are two cases:

1. $|S| \geq 2^{k+1}$, take $\epsilon = \frac{1}{4}$
   Then $|S| \geq \frac{4}{\epsilon^2} \cdot 2^m = 64 \cdot 2^m$
   With probability $\geq \frac{3}{4}$, the number of satisfying assignments hashed to 0 is $\geq 64(1 \pm \frac{1}{4}) > 48$
   If this is larger than 48 there are > 48 satisfying assignments and the prover can give them to us.

2. $|S| < 2^k$, take $\epsilon = \frac{1}{2}$
   Then there is some S' such that $S \subseteq S'$ and $|S'| = 2^k = 32 \cdot 2^m$
   With probability $\geq \frac{3}{4}$, the number of satisfying assignments hashed to 0 is $\leq 32(1 \pm \frac{1}{2}) \leq 48$
   So in this case there are not more than 48 satisfying assignments, so the prover cannot provide more than 48 satisfying assignments.

We can then amplify the $\frac{3}{4}$ probability by repetition to get an arbitrarily large probability of success.

2.5 Conclusion

In this lecture we proved the Valiant-Vazirani Theorem for $UniqueSAT$ and $\oplus SAT$, and then saw an algorithm in BPP^{NP} for approximate counting. Next time, we will go from approximate counting to exact counting.

References

[1] Valiant, L.; Vazirani, V. *NP is as easy as detecting unique solutions*, Theoretical Computer Science 47: 8593. 1986.