6.841/18.405J: Advanced Complexity Theory	**Wed, Apr 09, 2003**
Lecture 16	
Instructor: Madhu Sudan	*Scribe: Daniel Corson*

Today

Average-case complexity

- permanent
- DNP-completeness

Distributional complexity

Optimization problems may be easier to solve practically than previous theoretical treatment would indicate, because:

- near-optimal solutions may be easier to find than optimal ones, and
- random instances may be easier to solve than worst-case instances.

(Optimizing the bin-packing problem, for example, is presumably hard, but we know that we can easily find solutions with a constant number more than the optimal number of bins, and it may even be easy if we only allow one more bin than the optimal number.)

So for the purposes of distributional complexity, our instances are pairs (Π, D) where Π is a computational problem (often a language like SAT) and D is a distribution, which we will treat more specifically later. It is important, of course, that the distribution matches the application.

The big question is if it is easy to solve Π over the distribution D. We would like to allow that very rare instances are say exponentially hard, for then we would still certainly say that the problem is well-solved for (almost?) all practical purposes, but we also don't want to make our definition so broad that problems with undecidable instances are considered well-solved. In the end, we choose the following. A problem and distribution pair $(\Pi, D) \in$ AVG-P (informally, it is easy) means there exists an algorithm $\mathrm{A}(\cdot,\cdot)$ (first $\cdot$ input, second $\cdot$ confidence) such that for all instance lengths n and confidences δ, $\Pr_{x \in D, |x|=n}[\mathrm{A}(x,\delta) \text{ solves } \Pi] \geq 1-\delta$, and $\mathrm{A}(x,\delta)$ runs in time polynomial in both $|x|$ and $\frac{1}{\delta}$. (We also notate $\{x \in D : |x| = n\}$ by D_n).

The distributional complexity of SAT on various distributions is something interesting that we may come back to later in the course.

The permanent

For $n \in \mathbb{N}$, $[n]$ denotes $\{1, 2, \cdots, n\}$ and S_n denotes $\{\text{bijections } \pi : [n] \to [n]\}$. The permanent of an $n \times n$ matrix M is given as $\mathrm{perm}(M) = \sum_{\pi \in S_n} \prod_{i \in [n]} M_{i,\pi(i)}$. Notice when M is a $\{0,1\}$-matrix that the permanent counts the perfect matchings of the (n,n)-bipartite graph whose (condensed, because it is bipartite) adjacency matrix is M. We know that the (non-distributional) problem of calculating the permanent of a $\{0,1\}$-matrix is #P-hard [Valiant].

The following is a rare instance where we have been able to say anything about implications of distributional complexity on classical complexity class relationships. It is a worst case to average-case reduction!

Theorem 1 (Lipton). *If the problem of computing permanents of uniformly distributed* $\{0,1\}$-*matrices is in* AVG-P, *then* $\mathrm{P}^{\#P} \subseteq BPP$.

We will use without proof the infamous Chinese remainder theorem (CRT), which states that given relatively prime integers $p_1, \ldots, p_n$, and the values of some integer a mod p_i for $i \in [n]$, we can determine via a polynomial time computation a mod $\prod_{i \in [n]} p_i$.

Proof. Let B be our Avg-P algorithm. We'll construct a BPP one. Given a matrix A, we will pick n different primes p all with $n+1 < p \leq n^{10}$. Then we will determine $\text{perm}(A)$ mod p for each one. By CRT, we can determine from this $\text{perm}(A)$, since there are no more than $n!$ possible values (think of the correspondencse to perfect matchings in (n,n)-bipartite graphs). But we will do the picking in a funny way, in order to maintain uniform distribution. We will pick uniformly distributed random integers from $[n^{10}]$ until we have n distinct "good ones", and those that are too small or are composite, we will end up ignoring (though we will still go through the same motions with *all* the randomly choosen numbers, so it is always clear that we are never biasing our uniform distribution, which is important if we wish to be allowed to use B, though when we cover distributional problem reductions, we will be able to see that it did not actually matter, which makes intuitive sense anyway, since doing computations and throwing them away without looking is unlikely to help you solve problems).

So, how do we determine each $\text{perm}(A)$ mod p? Pick a random matrix $R \in \mathbb{Z}_p^{n \times n}$, then construct the $n+1$ matrices $M_i = A + iR \pmod{p}$, $i \in [n+1]$. Notice that, individually, the M_i are uniformly distributed (seeing any *one* of them tells you nothing about A) as long as p is prime and greater than $n+1$. Also notice that $\text{perm}(M_x)$ depends on x as a polynomial $g(x)$ of degree n. So we can run B to determine $g(x) = \text{perm}(M_x)$ for all $x \in [n+1]$ (with high probability, of course; we'll take care of that accounting at the end). We have enough values of g to determine it's coefficients, so we can in this way obtain $g(0) = \text{perm}(M_0) = \text{perm}(A \bmod p) = \text{perm}(A) \bmod p$.

And remember if p is less than $n+1$ or composite, we can do the same things anyway, just don't count on any of it to work (which we don't, since we ignore the result).

So if we pick k p's total in the beginning, then run B $n+1$ times for each k, and B has error probability δ, then we know the total error probability is $1 - k(n+1)\delta$, which is sufficiently high for our purposes. $\square$

Distributions, reductions, and DNP-completeness

We wish to define a distributional analogue to the notion of reductions. Allowing all statistical distributions to be problem distributions makes defining such an analogue less useful because it allows contrived adversorial distributions that would never come up in practice which trivialize the theory. So instead we restrict our attention to a more feasible class of distributions, but we obviously don't want to lose the uniform distribution, nor what it can become under polynomial transformation. We call P-sampleable a distribution that is spat out by some polynomial-time algorithm which is fed a uniform distribution. From now on, we only work with such distributions.

Before we define reductions (which will only fully happen next time), we need another definition. Where D' and D are (P-sampleable) distributions, for D' to α-dominate D ($\alpha \geq 1$) means that for all x, $\Pr[x \text{ accords to } D'] \geq \frac{1}{\alpha} \Pr[x \text{ accords to } D]$.

Even now, before we have a fully general distributional reduction theory, we can see that if D' α-dominates D, then if an algorithm B solves Π on D' with probability $1 - \delta$, then it also solves Π on D with probability $1 - \alpha\delta$.

Next time

- use this to formulate a finer definition of reduction
- show that a particular problem is DNP-complete