6.841/18.405J: Advanced Complexity **Monday, April 7th, 2003**

Lecture 15

Instructor: Madhu Sudan *Scribe: Mike Neiman*

In this lecture, we will cover the following topics:

- PCPs and Inapproximability
- Approximability
- Average-Case Complexity

1 Probabilistically Checkable Proofs and Inapproximability

1.1 Probabilistically Checkable Proofs (PCPs)

We begin by recalling the definition of PCPs that was given in the last lecture. A (r, q)-restricted verifier V is a probabilistic polynomial time oracle machine that, given input x, uses at most $r(|x|)$ coin tosses and queries the oracle in at most $q(|x|)$ bits. A language $L \in$ PCP$[r, q]$ if $\exists$ (r, q)-restricted verifier V such that:

$$x \in L \Rightarrow \exists \Pi \; Pr[V^{\Pi}(x) \text{ accepts}] = 1$$

$$x \notin L \Rightarrow \forall \Pi \; Pr[V^{\Pi}(x) \text{ accepts}] \leq \frac{1}{2}$$

Let n be the length of the input. We have the following results:

- PCP[0,poly(n)]=NP
- PCP[poly(n),0]=co-RP
- PCP[poly(n),poly(n)]=NEXPTIME
- PCP[$O(log(n))$,$O(1)$]=NP
- PCP[$O(log(n))$,3]=NP

The first two results follow immediately from the definitions of the complexity classes, but the remaining results are non-trivial to prove. We won't prove any of these results here. References to proofs in the literature were given in the previous lecture.

1.2 Inapproximability of 3SAT

Let's take a closer look at the result that PCP[$O(log(n))$,3]=NP. Let V be a $(r, 3)$-restricted verifier for language $L \in$NP and let Π be an optimal proof (a proof with maximum accepting probability) for V, where $r = O(log(n))$. For each random string, V performs a "test" based on 3 bits of the proof Π. For example:

$$00 \cdots 00 \longrightarrow \Pi_1 = 1 \Rightarrow \Pi_2 = \Pi_5$$

$$00 \cdots 01 \longrightarrow \Pi_2 = 1 \Rightarrow \Pi_3 = 1 \text{ or } \Pi_4 = 0$$

Each test can be converted to a 3CNF formula with 8 clauses on at most 7 variables. For each random string s, this gives a 8-clause 3CNF ϕ_s. We now set

$$\Phi_x = \bigwedge_{s \in \{0,1\}^r} \phi_s$$

Note that Φ_x is a 3CNF formula with $8 \cdot 2^r$ clauses that possesses the following property:

$$x \in L \Rightarrow \Phi_x \text{ is satisfiable}$$

$$x \notin L \Rightarrow \text{ any assignment to } \Phi_x \text{ violates at least } \frac{1}{16}\text{th of all clauses}$$

Now, suppose there exists an algorithm A running in polynomial time that, given a 3CNF formula ϕ, can satisfy $(\frac{15}{16} + \epsilon) \cdot m$ clauses, where m is the maximum number of satisfiable clauses in ϕ. Let L be any NP-complete language. Using the above procedure, we can construct Φ_x, input this to A, and count the number of satisfied clauses. Consider the result and what it implies:

- If more than $\frac{15}{16}$ clauses are satisfied, then $x \in L$.
- If not more than $\frac{15}{16}$ clauses are satisfied, then $x \notin L$.

We conclude that the existence of algorithm A leads to P=NP. Assuming (as usual) that P$\neq$NP, this result shows that it is impossible to approximate the number of satisfiable clauses by a factor of $(\frac{15}{16} + \epsilon)$.

2 Approximability

2.1 Combinatorial Optimization

Many important problems that arise in combinatorial optimization can be described by a triple

$$\Pi = (\text{sol?}, \text{obj}, \text{opt})$$

where sol?:$\{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}$ takes an instance of the problem and a proposed solution and determines whether the solution is valid, obj:$\{0,1\}^* \times \{0,1\}^* \rightarrow \mathbb{R}^{\geq 0}$ evaluates the objective function given a problem instance and solution, and opt$\in$ {min,max} specifies whether the objective function is to be minimized or maximized. We also require that sol? and obj be computable in polynomial time. As an example of such a problem, consider the traveling salesman problem (TSP). The instances of the problem are weighted graphs. The solutions are subgraphs. We have:

$$\text{sol?}(G, G') = 1 \text{ if } G' \text{ is a cycle spanning all vertices and is contained in } G$$

$$\text{obj}(G, G') = \text{ sum of weights of edges of } G'$$

$$\text{opt} = \min$$

Other examples would be determining the size of the largest clique in a graph or determining the minimum number of colors required to color a graph.

2.2 Heuristics for NP-complete Problems

The difficulty of NP-complete problems has led to the use of many heuristic algorithms. Of course, in order for these algorithms to be efficient, they can only be approximate. Typically, the results of a heuristic algorithm will be like: "Found solution that was 99% close to optimum in 95% of cases." Here, the 99% refers to approximability and the 95% refers to average-case behavior. We will now take a closer look at theses issues.

2.3 Approximability

When considering approximability, problems are of the form (Π, α), where Π is a standard combinatorial optimization problem and $\alpha : \mathbb{Z}^+ \to \mathbb{R}^{\geq 1}$ is an approximation parameter. Our goal is to find an algorithm A that, given x (instance of Π), computes a solution $A(x)$ satisfying:

$$\frac{opt}{\alpha(|x|)} \leq \text{obj}(x, A(x)) \leq opt \cdot \alpha(|x|)$$

where *opt* is the optimum value. Notice that the problem $(\Pi, 1)$ is equivalent to the original problem Π. From the theory of Np-completeness, we know that the three problems (Bin Packing, 1), (MaxSAT, 1), and (Coloring, 1) can each be reduced to one another. However, this does not imply that the three problems (Bin Packing, 2), (MaxSAT, 2), and (Coloring, 2) are equally difficult algorithmically. Here are a few results:

- (MaxSAT, $\frac{16}{15} - \epsilon$) is NP-hard
- (MaxSAT, 2)$\in$P
- (Coloring, $\frac{4}{3} - \epsilon$) is NP-hard
- (Clique, 2) is NP-hard

The first result was proved in section 1.2, and the second follows immediately from the fact that either the assignment of all zeros or the assignment of all ones must satisfy at least half of all clauses. The third result is a direct consequence of the NP-hardness of determining 3-colorability. The last result will be left as an exercise in a future problem set.

3 Average-Case Complexity

Here we consider problems of the form (Π, D), where Π is a standard problem and D is a distribution on instances. For example, D may be the uniform distribution on all n-bit strings. Suppose that we have an algorithm A that satisfies the following:

- on inputs not in S, A runs in time n^2
- on inputs in S, A runs in time 4^n

where $S \subseteq \{0,1\}^n$ and $|S| < 2^{\sqrt{n}}$. We would consider the problem well-solved in practice by A for uniformly distributed instances.

We now give the definition of the complexity class Avg-P, which is the average-case analog of P. We say that $(\Pi, D) \in$Avg-P if there exists algorithm $A(\cdot, \cdot)$ solving Π such that:

$$\forall \delta \ Pr_{x \leftarrow D}[A(x, \delta) \text{ solves } \Pi \text{ in time poly}(|x|, \frac{1}{\delta})] \geq 1 - \delta$$

We will consider average-case complexity and Avg-P in more detail in the next lecture.