CSE 533: The PCP Theorem and Hardness of Approximation **(Autumn 2005)**

Lecture 1: Introduction

Sep. 28, 2005

Lecturer: Ryan O'Donnell *Scribe: Ryan O'Donnell*

1 Proof, and The PCP Theorem

The PCP Theorem is concerned with the notion of "proofs" for mathematical statements. We begin with a somewhat informal definition:

Definition 1.1. *A* traditional proof system *works as follows:*

- *A statement is given. (e.g., "this graph $G = \cdots$ is 3-colorable" or "this CNF formula $F = (x_1 \vee \overline{x}_2 \vee \overline{x}_{10}) \wedge \cdots$ is satisfiable").*
- *A* prover *writes down a proof, in some agreed-upon format.*
- *A* verifier *checks the statement and the proof, and accepts or rejects.*

From the perspective of theory of computer science, we usually fix a constant size alphabet and assume the statement and proof are strings over this alphabet; we also usually write n for the length of the statement and measure lengths of strings and running times in terms of n. The familiar complexity class NP can be cast in this setup:

Remark 1.2. *Language L is in NP iff there is a* polynomial time *deterministic verifier V (a Turing Machine, say) and an arbitrarily powerful prover P, with the following properties:*

- *"Completeness": For every $x \in L$, P can write a proof of length $\mathrm{poly}(|x|)$ that V accepts.*
- *"Soundness": For every $x \notin L$, no matter what $\mathrm{poly}(|x|)$-length proof P writes, V rejects.*

To equate the notion of the verifier being efficient with it being a deterministic polynomial time algorithm is nowadays a bit quaint; ever since the late '70s we have been quite happy to consider randomized polynomial time algorithms to be efficient. As it turns out, when proof systems are allowed to have randomized verifiers, some very surprising things can happen. This line of research was begun in the early-to-mid '80s by Goldwasser, Micali, and Rackoff [9] and also independently by Babai [3, 4]. See the accompanying notes on the history of the PCP theorem. One pinnacle of research in this area is *The PCP Theorem*:

Theorem 1.3. *(due to Arora-Safra (AS) [2] and Arora-Lund-Motwani-Sudan-Szegedy (ALMSS) [1])* "The PCP (Probabilistically Checkable Proof) Theorem"*:*

All languages $L \subseteq NP$ have a P.C.P. system wherein on input $x \in \{0,1\}^n$:

- *Prover P writes down a $\mathrm{poly}(n)$-bit-length proof.*
- *Verifier V looks at x and does polynomial-time deterministic computation. Then V uses $O(\log n)$ bits of randomness to choose C random locations in the proof. Here C is a absolute universal constant; say, 100. V also uses these random bits to produce a deterministic test (predicate) ϕ on C bits.*
- *V reads the bits in the C randomly chosen locations from the proof and does the test ϕ on them, accepting or rejecting.*
- *Completeness: If $x \in L$ then P can write a proof that V accepts with probability 1.*
- *Soundness: For every $x \notin L$, no matter what proof P writes, V accepts with probability at most $1/2$.*

Remark 1.4. *This P.C.P. system has "one-sided error": true statements are always accepted, but there is a chance a verifier might accept a bogus proof. Note that this chance can be made an arbitrarily small constant by naive repetition; for example, V can repeat its same spot-check 100 times independently, thus reading $100C$ bits and accepting false proofs with probability at most 2^{-100}.*

The first time one sees this theorem, it seems a little hard to conceive how it can be true. It's even more striking when one learns that essentially C may be taken to be 3. (See Theorem 1.12 below.) How could you possibly be convinced a proof is true just by spot-checking it in 3 bits?

Remark 1.5. *By the classical theory NP-completeness, it suffices to prove the PCP Theorem for one particular NP-complete language — say, 3-COLORING or 3-SAT — since poly-time reductions can be built into the verifier's initial step (and into the prover's plans).*

Remark 1.6. *The PCP that the prover needs to write down can be obtained in deterministic polynomial time from the "standard" proofs for $x \in L$ (i.e., the coloring for 3-COLORING, the assignment for 3-SAT).*

Remark 1.7. *Sometimes enthusiastic descriptions of the PCP Theorem make it seem like it greatly reduces the* time *a verifier needs to spend to check a proof. This is not accurate since the verifier still dies polynomial-time deterministic pre-computations; these may already take more time then it would have taken to simply check a classical proof. What the PCP Theorem saves is* proof accesses*. There is other work on developing proof systems that let the verifier save on time or space (see the accompanying notes on the history of the PCP Theorem); however it seems to have had fewer interesting applications.*

Our first task in this course will be to prove Theorem 1.3 completely. The fact that this will be possible is only due to a very recent development. The original proof of the PCP Theorem was very intricate and difficult; it might have been up to 100 pages, with subsequent simplifications bringing it down to a very densely packed 30 pages or so. However in April 2005, Irit Dinur gave a new proof [5] which is elegant and clear and only a dozen pages or so long. This is the proof we

will see in the course.

Subsequent to the PCP Theorem were many more "PCP Theorems" that strengthened certain parameters or extended the result in different directions. What follows are a few of these:

Theorem 1.8. *(Feige-Kilian [8], Raz [17]) (Raz's strong version of this result is sometimes called "the Raz Verifier" or "hardness of Label Cover"): For every constant $\epsilon > 0$, there is a poly-size PCP for NP that reads* two *random proof entries written with $O(1)$-size alphabet and has completeness 1, soundness ϵ.*

Remark 1.9. *Note that in this theorem, the poly-size of the PCP and the alphabet size both depend on ϵ; with Raz's version, the proof has length $n^{O(\log 1/\epsilon)}$ and the alphabet has size $\mathrm{poly}(1/\epsilon)$.*

Remark 1.10. *The result proven is actually stronger in a technically subtle but important way: One can additionally have the verifier use a predicate $\phi(x, y)$ with the "projection property", namely, that for every choice of x there is exactly one choice of y that makes $\phi(x, y)$ true.*

Remark 1.11. *Comparing this result to the basic PCP Theorem, we see that it uses a constant-size alphabet and two queries to get arbitrarily small constant soundness, whereas the basic PCP Theorem uses constantly many queries to a size-two alphabet. It might not be immediately clear which is better, but it is indeed the former. There are several ways to look at this: For example, with fewer queries you have fewer opportunities to "cross-check"; as an extreme, it's clear that a verifier that made only one query (to a constant size alphabet) could always be fooled. Or suppose that you tried to encode every triple of bits in a proof with a single character from an alphabet of size 8 — although you could now read three bits with just one query, the prover can cheat you by encoding a single bit in different ways in different triples.*

We hope to prove Theorem 1.8 in this course — at least, the Feige-Kilian version without the projection property.

The following result essentially shows that we can take $C = 3$ in the original PCP Theorem:

Theorem 1.12. *(Håstad [12]) "3-LIN hardness": For every constant $\epsilon > 0$, there is a poly-size PCP for NP that reads just* three *random* bits *and tests their XOR. Its completeness is $1 - \epsilon$ and its soundness is $1/2 + \epsilon$.*

Remark 1.13. *This result has "imperfect completeness". However, if one is willing to allow an* adaptive *three-bit-querying verifier (i.e., the verifier does not have to pick the three bits in advance but can base what bit it reads next on what it's seen so far) then one can get completeness 1. This is due to Guruswami, Lewin, Sudan, and Trevisan [10].*

This result, which we will prove in the course, requires Theorem 1.8.

Finally, here is one more PCP Theorem which we *won't* prove:

Theorem 1.14. *(due to Dinur [5], based heavily on a result of Ben-Sasson and Sudan [**?**]): In the basic PCP Theorem, the proof length can be made $n \cdot \mathrm{polylog}(n)$ rather than $\mathrm{poly}(n)$.*

1.1 Hardness of approximation

Perhaps the most important consequence of the PCP theorems and the most active area of research in the area are results about "hardness of approximation". These will be the major focus of the second half of this course. To be able to state hardness of approximation results, we need to understand the notion of *(NP) combinatorial optimization problems*. Instead of making a formal definition we will just give some examples. Briefly, these are "find the best solution" versions of classic NP-complete problems.

Definition 1.15. *MAX-E3SAT: Given an E3CNF formula — i.e., a conjunction of "clauses" over boolean variables $x_1, \ldots, x_n$, where a clause is an OR of exactly 3 literals, x_i or $\overline{x_i}$ — find an assignment to the variables satisfying as many clauses as possible.*

Definition 1.16. *SET-COVER: Given a bunch of sets $S_1, \ldots, S_m \subseteq \{1, \ldots, n\}$, find the fewest number of them whose union covers all of $\{1, \ldots, n\}$. (We assume that every ground element i is in at least one set S_j.)*

Definition 1.17. *MAX-CLIQUE: Given an undirected graph, find the largest clique in it, where a clique is a subset of vertices which contain all possible edges.*

Definition 1.18. *KNAPSACK: Given are "weights" $w_1, \ldots, w_n \geq 0$ of n items and also "values" $v_1, \ldots, v_n \geq 0$. Also given is a "capacity" C. Find a set of items S such that $\sum_{i \in S} w_i \leq C$ while maximizing $\sum_{i \in S} v_i$.*

Remark 1.19. *Each of these is associated to a classic NP-complete decision problem; e.g., "CLIQUE: Given G and k, does G have a clique of size at least k?" Notice that frequently the NP decision problem is a contrived version of the more natural optimization problem.*

Remark 1.20. *Combinatorial optimization problems can be divided into two categories: Maximization problems (like MAX-3SAT, MAX-CLIQUE, KNAPSACK) and minimization problems (like SET-COVER).*

It is well-known that these problems are all NP-hard. However, suppose that for, say, MAX-E3SAT, there was a polynomial time algorithm with the following guarantee: Whenever the input instance has optimum OPT — i.e., there is an assignment satisfying OPT many clauses — the algorithm returns a solution satisfying $99.9\% \times OPT$ many clauses. Such an algorithm would be highly useful, and would tend to refute the classical notion that the NP-hardness of MAX-E3SAT means there is no good algorithm for it.

Indeed, such results are known for the KNAPSACK problem. As early as 1975, Ibarra and Kim [14] showed that for every $\epsilon > 0$ there is an algorithm for KNAPSACK that runs in time $\mathrm{poly}(n/\epsilon)$ and always returns a solution which is within a $(1-\epsilon)$ factor of the optimal solution. So, although KNAPSACK is NP-complete, in some sense it's very easy. Let us make some definitions to capture these notions:

Definition 1.21. *Given a combinatorial optimization* maximization *problem, we say algorithm A is an α-*approximation algorithm *(for $0 < \alpha \leq 1$) if whenever the optimal solution to an instance*

has value OPT*,* A *is guaranteed to return a solution with value at* least $\alpha \cdot \mathrm{OPT}$*. We make the analogous definition for* minimization *problems, with* $\alpha \geq 1$ *and* A *returning a solution with value at* most $\alpha \cdot \mathrm{OPT}$*. Unless otherwise specified, we will also insist that* A *runs in polynomial time.*

Remark 1.22. *Our definition for maximization problems is sometimes considered unconventional; some like to always have* $\alpha \geq 1$*, in which case their notion is that the algorithm* A *returns a solution with value at least* OPT/α*.*

Definition 1.23. *A maximization (resp., minimization) combinatorial optimization problem is said to have a* PTAS *(Polynomial Time Approximation Scheme) if it has a* $(1-\epsilon)$*-approximation algorithm (resp.,* $(1+\epsilon)$*-approximation algorithm) for every constant* $\epsilon > 0$*.*

As mentioned, the KNAPSACK problem has a PTAS, and this is true of certain other combinatorial optimization problems, mostly related to scheduling and packing. But what about, say, MAX-E3SAT? It is a remarkable consequence of the PCP Theorem that MAX-E3SAT has no PTAS unless $\mathrm{P} = \mathrm{NP}$. In fact, the two statements are basically equivalent!

Theorem 1.24. *(credited to an unpublished 1992 result of Arora-Motwani-Safra-Sudan-Szegedy): Speaking roughly, the PCP Theorem is equivalent to the statement, "MAX-E3SAT has no PTAS assuming* $\mathrm{P} \neq \mathrm{NP}$*".*

We will precisely formulate and prove this theorem in the next lecture.

Indeed, most work in the PCP area these days is centered on proving "hardness of approximation" results like Theorem 1.24. We will state here a few striking "optimal hardness-of-approximation results" that have followed from work on PCP theorems.

Theorem 1.25. *(follows from Håstad's Theorem 1.12): MAX-E3SAT has no* $(7/8+\epsilon)$*-approximation algorithm for any constant* $\epsilon > 0$ *unless* $\mathrm{P} = \mathrm{NP}$*.*

We will see the proof of this in the course.

Remark 1.26. *There is a very easy* $7/8$*-approximation algorithm for MAX-E3SAT: Just picking a random assignment gives a* $7/8$*-approximation in expectation, and this algorithm is easily derandomized.*

Regarding SET-COVER:

Theorem 1.27. *(Feige [6]): SET-COVER has no* $(1-\epsilon)\ln n$ *approximation algorithm for any constant* $\epsilon > 0$ *unless* $\mathrm{NP} \subseteq \mathrm{DTIME}(n^{\log\log n})$*.*

Remark 1.28. *The greedy algorithm achieves a* $(\ln n + 1)$*-approximation algorithm. (Johnson [15])*

Remark 1.29. *The fact that we have the conclusion "unless* $\mathrm{NP} \subseteq \mathrm{DTIME}(n^{\log\log n})$*" is due to technical difficulties; however since the conclusion is almost as unlikely as* $\mathrm{NP} = \mathrm{P}$*, we don't really mind much.*

We won't prove Feige's theorem about SET-COVER in this course but we will prove a due to Lund and Yannakakis [16], that shows hardness of giving a $\Omega(\log n)$-approximation.

The situation for MAX-CLIQUE is direst of all:

Theorem 1.30. *(due to Håstad [11], with a significant simplification by Samorodnitsky-Trevisan [18], another simplification by Håstad and Wigderson [13], and a slight improvement by Zuckerman in September 2005 [19]): MAX-CLIQUE has no $(1/n^{1-\epsilon})$-approximation for any constant $\epsilon > 0$ unless $\mathrm{P} = \mathrm{NP}$.*

Remark 1.31. *There is a trivial $1/n$-approximation: Output a single vertex.*

We won't prove this theorem in the course, but the weaker result that $(1/n^{\Omega(1)})$-approximating is hard follows relatively easily from the main PCP Theorem, via a reduction given by Feige-Goldwasser-Lovasz-Sudan-Szegedy [7]. We will give this reduction later in the course.

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998.

[2] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. *J. ACM*, 45(1):70–122, 1998.

[3] L. Babai. Trading group theory for randomness. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, STOC'85 (Providence, RI, May 6-8, 1985)*, pages 421–429, New York, 1985. ACM, ACM Press.

[4] L. Babai and S. Moran. Arthur-Merlin games: A randomized proof system, and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36(2):254–276, Apr. 1988.

[5] I. Dinur. The pcp theorem by gap amplification. ECCC, TR05-046, 2005.

[6] U. Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45(4):634–652, 1998.

[7] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Interactive proofs and the hardness of approximating cliques. *J. ACM*, 43(2):268–292, 1996.

[8] U. Feige and J. Kilian. Two-prover protocols — low error at affordable rates. *SIAM J. Comput.*, 30(1):324–346, 2000.

[9] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput*, 18(1):186–208, 1989.

[10] V. Guruswami, D. Lewin, M. Sudan, and L. Trevisan. A tight characterization of NP with 3 query PCPs. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science(FOCS-98)*, pages 8–17, Los Alamitos, CA, Nov.8–11 1998. IEEE Computer Society.

[11] J. Håstad. Clique is hard to approximate to within $n^{1-\epsilon}$. *Acta Mathematica*, 182:105–142, 1999.

[12] J. Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001.

[13] J. Håstad and A. Wigderson. Simple analysis of graph tests for linearity and PCP. *Random Struct. Algorithms*, 22(2):139–160, 2003.

[14] O. H. Ibarra and C. E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of the ACM*, 22(4):463–468, Oct. 1975.

[15] D. S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. Syst. Sci*, 9(3):256–278, 1974.

[16] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *J. ACM*, 41(5):960–981, 1994.

[17] R. Raz. A parallel repetition theorem. *SIAM J. Comput*, 27(3):763–803, 1998.

[18] A. Samorodnitsky and L. Trevisan. A PCP characterization of NP with optimal amortized query complexity. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing, STOC'2000 (Portland, Oregon, May 21-23, 2000)*, pages 191–199, New York, 2000. ACM Press.

[19] D. Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. ECCC, TR05-100, 2005.