CS880: Approximation Algorithms

Scribe: Tom Watson **Lecturer:** Shuchi Chawla
Topic: Inapproximability **Date:** 5/1/2007

In this lecture we continue our discussion of inapproximability by discussing results based on probabilistically checkable proofs (PCPs). In particular, we show how the PCP Theorem, a deep result in computational complexity, is essentially equivalent to the inapproximability of MAX-3SAT. We also show how a strengthening of the PCP Theorem leads to a tight inapproximability result for MAX-3SAT, and we discuss other issues related to PCPs.

28.1 Probabilistically Checkable Proofs

Recall the template developed in the last lecture for proving inapproximability results. One exhibits a reduction from an NP-complete problem such as SAT to an optimization (say, maximization) problem L in such a way that Yes instances are mapped to instances with optimal objective value at least some α and No instances are mapped to instances with optimal objective value at most some β. The existence of such a reduction proves that L is NP-hard to approximate within any factor better than α/β. In the last lecture we saw examples where standard NP-hardness reductions were used to obtain weak inapproximability results in the above way, and self-reductions were used to magnify the hardness factor achieved. These reductions primarily exploited properties of the problem for which an inapproximability result was desired, i.e. the problem being reduced to. In this lecture, we instead focus on properties of the problem being reduced *from*.

The problem being reduced from is an NP-complete problem, or more generally an arbitrary problem in NP. We now revisit the definition of NP and see how different characterizations of NP can afford additional structure that we can exploit in our gap-introducing reductions. Ideally, we would like the Yes instances to look very different from the No instances in some sense, so that it's easier to introduce a gap in the reduction.

Definition 28.1.1 *A language L is in NP if there exists a polynomial-time verifier V and a constant c such that for an input x of length n, the following two properties are satisfied.*

1) *(Completeness) If $x \in L$ then there exists a witness y of length $O(n^c)$ such that V accepts the pair $\langle x, y\rangle$.*

2) *(Soundness) If $x \notin L$ then for all witnesses y of length $O(n^c)$, V rejects $\langle x, y\rangle$.*

The terminology *completenss* and *soundness* is a carry-over from logic, where completenss refers to the property that all true statements are provable in a given proof system, and soundness refers to the property that no false statements are provable.

Intuitively, there isn't much of a gap between Yes instances and No instances according to this definition, since out of the exponentially many candidate witnesses, a single witness can make or break the membership of x to L. We now alter our notion of proof verification from that of explicitly verifying the correctness of a proof to that of just randomly spot-checking a few locations of the proof. This is the notion of *probabilistically checkable proofs*, or PCPs for short. This notion leads to a surprising and robust characterization of NP, and the restricted structure of this new type of verifier for NP languages can be exploited in our reductions to prove a wide variety of inapproximability results.

Definition 28.1.2 *A language L is in $PCP_{c,s}(r, q)$ if there exists a polynomial-time verifier V that, given an input x of length n and a purported proof y, runs in time $poly(n)$, uses $r(n)$ bits of randomness, makes $q(n)$ queries to y, and satisfies the following two properties.*

1) *(Completeness) If $x \in L$ then there exists a y such that V accepts the pair $\langle x, y\rangle$ with probability at least c.*

2) *(Soundness) If $x \notin L$ then for all y, V accepts $\langle x, y\rangle$ with probability at most s.*

Note that in principle there is no limit on the size of y. Of course, y should be at most exponentially long, since otherwise the verifier would not have time to write down an index into y. However, we will soon see that for NP, y only needs to be polynomially long. Also note that the size of the alphabet that y is presented in is not specified. Frequently, we assume the binary alphabet is used, but sometimes it is more convenient to work with larger alphabets.

Let us see how PCPs relate to the standard definition of NP presented above. Clearly, a standard NP verifier can be thought of as a PCP verifier that uses no randomness, queries polynomially many bits of the proof, and accepts with probability 1 or 0. Thus, the standard definition of NP can be viewed as follows.

Observation 28.1.3 $NP = PCP_{1,0}(0, poly(n))$.

This characterization has excellent completeness and soundness parameters, but it uses many queries. At the other extreme, we have the following characterization.

Observation 28.1.4 $NP = PCP_{1/2^{O(n)},0}(poly(n), 0)$.

Proof: An NP statement can be verified by randomly guessing a witness and checking that it causes the standard verifier to accept. Conversely, a $PCP_{1/2^{O(n)},0}(poly(n), 0)$ language can be solved in NP by interpreting the witness as a random string that causes the PCP verifier to accept. ■

This characterization is not terribly useful either since the completeness parameter is terrible. Is there some way to interpolate between these two characterizations? As a step in this direction, we can obtain the following characterization, which exploits the NP-completeness property of 3SAT.

Observation 28.1.5 $NP = PCP_{1,1-1/n^{O(1)}}(O(\log n), 3)$.

Proof: The inclusion from right to left follows from Lemma 28.1.7, which we argue below. For the inclusion from left to right, it suffices to demonstrate a $PCP_{1,1-1/m}(O(\log m), 3)$ verifier for 3SAT, where m is the number of clauses in the given formula φ. The verifier can interpret the proof as an assignment the variables of φ. It can select one clause uniformly at random, query the three bits of the proof containing the values of the variables in that clause, and accept if and only if the clause is satisfied by the assignment. If φ is satisfiable then a satisfying assignment causes the verifier to accept with probability 1. If φ is not satisfiable then all purported proofs encode a non-satisfying assignment, and so with probability at least $1/m$, the three queried bits will not satisfy that clause. ■

The 3SAT verifier described above has very nice r and q parameters, but its soundness is rather high. This is because an unsatisfiable formula may be very close to being satisfiable, in the sense that some assignment satisfies all but one of its clauses. Indeed, recalling the standard mapping reduction from an arbitrary NP language to 3SAT, one can see that the formulas produced are encodings of nondeterministic computations, and there is a single clause expressing that the computation must be *accepting*. It is always possible to satisfy all clauses except this last one simply by encoding a valid nonaccepting computation.

Driving the soundness down to a constant value without spoiling the r and q parameters requires working with encodings that are robust in the sense that any error in the proof must be "spread out" so that it is caught with constant probability. In fact, this can be achieved; this is the content of the famous PCP Theorem, due to [1] and [2]. The proof of this celebrated result is very involved, so we omit it.

Theorem 28.1.6 (The PCP Theorem) $NP = PCP_{1,1/2}(O(\log n), O(1))$.

The difficult direction of the PCP Theorem is the inclusion from left right. The other inclusion follows from the following lemma. This lemma also finishes the proof of Observation 28.1.5. Finally, this lemma justifies the claim we made earlier that restricting our attention to polynomial-size proofs for NP is no loss of generality.

Lemma 28.1.7 $PCP_{c,s}(r(n), q(n)) \subseteq NTIME(2^{O(r(n)+q(n))} n^{O(1)})$.

Proof: Consider a PCP where the verifier uses $r(n)$ random bits and makes $q(n)$ queries. For each random string, the number of locations of the proof that could ever get queried is at most $2^{q(n)}$ (due to the possibly adaptive nature of the verifier). Thus over all proofs and all random strings, there are at most $2^{r(n)+q(n)}$ locations that ever get queried (for a given input). A nondeterministic machine can guess the addresses and answers to these queries in time $2^{r(n)+q(n)} n^{O(1)}$. It can then run over all random strings, simulate the verifier for each one (looking up the answers to queries in the guessed table), and explicitly compute the probability of acceptance of the verifier for the guessed proof. This takes time $2^{r(n)} n^{O(1)}$. If the probability of acceptance is at least c then the machine should accept, and otherwise it should reject. ■

28.2 Inapproximability of MAX-3SAT

While the PCP Theorem is quite interesting and profound in its own right, some of its major customers are inapproximability results. We now show that the PCP Theorem is actually equivalent in some sense to the inapproximability of MAX-3SAT.

Theorem 28.2.1 *The following are equivalent.*

1) $NP \subseteq PCP_{1,1/2}(O(\log n), O(1))$.

2) There is a mapping reduction from 3SAT to 3SAT that maps satisfiable formulas to satisfiable *formulas and maps unsatisfiable formulas to formulas for which no assignment satisfies more than a $1-\epsilon$ fraction of the clauses, for some constant $\epsilon > 0$.*

Proof: We first prove that statement 2 implies statement 1. Suppose the reduction specified in statement 2 exists. We can design a $PCP_{1,1/2}(O(\log n), O(1))$ verifier for 3SAT by having the verifier perform the reduction on its input formula φ to obtain a formula ψ and assume the proof is an assignment to ψ. The verifier can randomly pick a clause of ψ and query the 3 bits of the proof specifying the values of the variables in that clause, as in Observation 28.1.5.

If φ is satisfiable then so is ψ and thus the verifier's test passes with probability 1 in this case. If φ is not satisfiable then every assignment to the variables of ψ, and in particular the assignment given by any proof, violates at least an ϵ fraction of the clauses. Thus the verifier's test passes with probability at most $1-\epsilon$. The soundness is not yet $1/2$, but we can pick another clause uniformly at random, independently of the first clause, and it will be satisfied by the proof's assignment with probability at most $1-\epsilon$, and so if the verifier only accepts if both tests pass, then the soundness is driven down to at most $(1-\epsilon)^2$. Repeating this a constant number of times drives the soundness down to $1/2$. The reason this soundness reduction strategy doesn't work in Observation 28.1.5 is that there, the soundness is $1-o(1)$ to begin with and so a super-constant number of repetitions would be needed to drive it down to $1/2$, and this would result in a super-logarithmic amount of randomness and a super-constant number of queries to the proof.

We now prove that statement 1 implies statement 2. Suppose that $NP \subseteq PCP_{1,1/2}(O(\log n), O(1))$; then in particular, there is a $PCP_{1,1/2}(r,q)$ verifier V for 3SAT where $r = O(\log n)$ and q is a constant. We assume that this verifier chooses its query locations nonadaptively; this is no loss of generality since an adaptive verifier that queries q bits of the proof can query at most $2^q - 1$ different locations for a given choice of random string, over all possible answers to those queries, and can thus be simulated by a nonadaptive verifier that makes at most $2^q - 1$ queries, which is still a constant number.

We design a mapping reduction from 3SAT to 3SAT satisfying the property of statement 2. Given a formula φ, we generate a formula ψ with a variable for each of the polynomially many bits of the proof V could ever query on input φ. We run over all random strings R of length r (of which there are only polynomially many), determine which q bits of the proof (variables of ψ) V queries for that choice of R, and determine which of the 2^q settings of those variables cause V to accept. This function on q variables can be expressed in CNF using at most 2^q clauses of size q. The conjunction of these clauses over all choices of R is a CNF formula with at most $2^q 2^r$ clauses. Now a clause $(\ell_1 \vee \cdots \vee \ell_q)$ where the ℓ_i's are literals can be expressed as

$$(\ell_1 \vee \ell_2 \vee z_2) \wedge (\overline{z_2} \vee \ell_3 \vee z_3) \wedge (\overline{z_3} \vee \ell_4 \vee z_4) \wedge \cdots \wedge (\overline{z_{q-3}} \vee \ell_{q-2} \vee z_{q-2}) \wedge (\overline{z_{q-2}} \vee \ell_{q-1} \vee \ell_q),$$

where $z_2, \ldots, z_{q-2}$ are new variables unique to that clause. It can be easily verified that a satisfying assignment to this conjunction yields a satisfying assignment to $(\ell_1 \vee \cdots \vee \ell_q)$ and vice versa. Expanding each clause in our CNF formula in this manner yields a 3CNF formula with at most $q2^q2^r = poly(n)$ clauses. If φ is satisfiable then ψ is satisfiable by setting the variables according to a proof that makes V accept with probability 1. If φ is not satisfiable, then each assignment to the variables of ψ yields a proof that causes V to accept for at most half of the random strings

R. Since the block of clauses corresponding to R can only satisfied by the assignment if V accepts this proof for that choice of R, it follows that at least half of these blocks must have at least one unsatisfied clause under this arbitrary assignment. Therefore the fraction of unsatisfied clauses is always at least $1/2q2^q$. We can take ϵ to be this value, and the theorem is proved. ■

Corollary 28.2.2 *There is no PTAS for MAX-3SAT unless $P = NP$.*

Proof: Statement 1 in Theorem 28.2.1 holds by the PCP Theorem and thus the mapping reduction and constant $\epsilon > 0$ given by statement 2 exists. A $(1 - \epsilon')$-approximation algorithm for MAX-3SAT for any $\epsilon' < \epsilon$ could be combined with the reduction in the standard way to arrive at a polynomial-time (exact) algorithm for 3SAT. Given a satisfiable formula, the reduction would produce a satisfiable formula, and then the MAX-3SAT algorithm would find an assignment that satisfies at least a $1-\epsilon' > 1-\epsilon$ fraction of the clauses. Given an unsatisfiable formula, the reduction would produce a formula for which no assignment satisfies more than a $1-\epsilon$ fraction of the clauses. Thus testing whether the assignment produced by the MAX-3SAT algorithm satisfies more than a $1-\epsilon$ fraction of the clauses would allow us to distinguish between the Yes and No instances of 3SAT. ■

28.3 Other PCP Theorems

The inapproximability factor $1 - \epsilon$ given in Theorem 28.2.1 is very close to 1 and is thus not very strong. In this section we consider other results related to PCPs that can be used to prove stronger inapproximability results.

28.3.1 Repetition of PCPs

We begin by considering the soundness parameter. Typically, the lower the soundness and the lower the number of queries, the stronger the inapproximability results one can prove. Having the prover provide k separate copies of the proof and running the verifier independently on each proof drives the soundness down to s^k, since if the input is not in the language, then each of the k sections of the proof defines some proof that makes the original verifier accept with probability at most s. This is called *sequential repetition.* While it has the desired effect on the soundness parameter, it increases the number of queries by a factor of k.

Another approach for reducing the soundness is called *parallel repetition.* In this approach the new verifier runs k copies (repetitions) of the old verifier simultaneously, using k independent random strings, but it waits for each repetition to formulate its first query and then sends all the queries to the prover at once. The prover responds to these k queries at once (and thus requires a larger alphabet), and the simulation of the k repetitions continues until they form their second queries, all of which are sent to the prover at once, and so on. If the original verifier makes q queries and the proof is written in the alphabet R, then the verifier for k parallel repetitions makes q queries and the proof is writen in the alphabet $R^{(k)} = R \times \cdots \times R$. Often, the increase in alphabet size is an acceptable tradeoff for keeping the number of queries at its original value.

But what happens to the soundness parameter? Unfortunately, it does not decrease to s^k in general. The basic reason for this can be intuited by considering the case $k = 2$. The first repetition assumes it is being run with some proof y_1 written in alphabet R and the second repetition assumes it is being run with some proof y_2 written in alphabet R. The actual proof y has of an entry for each pair consisting of an entry in y_1 and an entry in y_2. That is, if the ith entry of y_1 is supposed to be some symbol $a \in R$ and the jth entry of y_2 is supposed to be $b \in R$, then the (i, j) entry of y is supposed to be the symbol $(a, b) \in R \times R$. However, the entries of y might not be consistent with each other, in the sense that the (i, j) entry of y may contain (a, b) but the (i, j') entry for some $j' \neq j$ may contain (c, d) where $c \neq a$. Thus no unique proof y_1 can be extracted from the proof y in this case; the proof as seen by the first repetition can "adapt" depending on what the second repetition is doing. The proof can exploit this to get an overall probability of acceptance greater than s^2. We give a concrete example of this in the next subsection, in a slightly different PCP setting.

28.3.2 2-Prover 1-Round Games

In the 2-Prover 1-Round Game model, the verifier interacts with two provers who may agree on any strategy before the protocol begins but may not communicate during the protocol. The verifier flips some coins, asks one question (that depends on the outcome of the coin flips) to each prover, and decides whether to accept based on their responses. Each prover's response is a function only of its query and the common input, and not the other prover's query. Equivalently, we can think of this model as a PCP where the proof is divided into two parts — the first part is written in some alphabet R_1 and corresponds to the first prover's responses to the different queries the verifier could ask it, and the second part is written in some alphabet R_2 and corresponds to the second prover's responses to the different queries the verifier could ask it.

Definition 28.3.1 *A language L is in $2P1R_{c,s}(r)$ if there exists a polynomial-time verifier V that, given an input x of length n and access to two noncommunicating provers, runs in time poly(n), uses $r(n)$ bits of randomness, makes one query to each prover, and satisfies the following two properties.*

1) *(Completeness) If $x \in L$ then there exist provers such that V accepts x with probability at least c.*

2) *(Soundness) If $x \notin L$ then for all provers, V accepts x with probability at most s.*

Although this model may seem quite restricted since the verifier may only ask one question of each prover, it turns out that the verifier's ability to cross-check the noncommunicating provers' answers restricts their ability to cheat and allows this model to capture all of NP. This is illustrated in the following result.

Theorem 28.3.2 *$NP = 2P1R_{1,1-\epsilon'}(O(\log n))$ for some constant $\epsilon' > 0$.*

Proof: The inclusion from right to left is similar to Lemma 28.1.7 and is left as an exercise. For the other inclusion, by NP-completeness it suffices to show that 3SAT $\in 2P1R_{1,1-\epsilon'}(O(\log n))$. Let the input formula be φ. By the PCP Theorem and Theorem 28.2.1, there is reduction that maps φ to a 3CNF formula ψ such that if φ is satisfiable then so is ψ, and if φ is unsatisfiable then no assignment satisfies more than a $1 - \epsilon$ fraction of the clauses of ψ, for some constant $\epsilon > 0$. The verifier expects prover 1 to have an assignment to the variables of ψ, and we expect prover 2 to have for each clause a satisfying assignment to the variables in that clause. Thus if ψ has ℓ variables and m clauses then the first proof is written in the binary alphabet and is of length ℓ, and the second proof is written in an alphabet of size 7 and is of length m. The verifier selects a clause of ψ and one of the variables in that clause uniformly at random, queries that variable and that clause to provers 1 and 2 respectively, and accepts if and only if the two responses agree on the value of the chosen variable.

Now if φ is satisfiable then so is ψ, and the provers can give answers consistent with some satisfying assignment to ψ, and the verifier will accept with probability 1. If φ is not satisfiable then with probability at least ϵ, the verifier will choose a clause of ψ that is not satisfied by the assignment that prover 1 has written down. Since prover 2's response for this clause satisfies it, there must be at least one variable on which prover 2's response disagrees with prover 1's assignment. Conditioned on picking a clause that is not satisfied by prover 1's assignment, the verifier catches the inconsistency with probability at least 1/3. It follows that the soundness of this $2P1R$ system is at most $1 - \epsilon/3$. ■

The characterization given by Theorem 28.3.2 is frequently used in inapproximability results.

We now return to the question of parallel repetition. In the 2P1R setting, we can perform k parallel repetitions in the same way as with PCPs. This increases prover 1's alphabet from R_1 to $R_1^{(k)}$ and prover 2's alphabet from R_2 to $R_2^{(k)}$ but does not increase the number of queries. In this setting, we can give a simple concrete example to show that k parallel repetitions does not drive the soundness down to s^k in general.

Consider the (silly) protocol where the verifier selects two bits r_1, r_2 uniformly at random and sends r_1 to prover 1 and r_2 to prover 2. Each prover is supposed to respond with a pair (i, r) indicating that prover i was sent bit r. The verifier accepts if and only if the two responses agree and are both correct. Then for any outcome that causes the verifier to accept, say both provers respond with $(1, r_1)$, since flipping r_1 cannot change prover 2's response, the latter outcome leads to rejectance. Thus at most half of the outcomes lead to acceptance and the soundness of this protocol is at most 1/2. (It is also straightforward to see that the soundness is at least 1/2.)

However, when we take $k = 2$ parallel repetitions, the soundness stays at 1/2. In this case, the verifier picks four bits r_1, r_2, s_1, s_2 uniformly at random and sends (r_1, s_1) to prover 1 and (r_2, s_2) to prover 2. Each prover responds with a tuple (i_1, r, i_2, s) and the verifier accepts if and only if the two responses agree and $r_{i_1} = r$ and $s_{i_2} = s$. However, the provers can get the verifier to accept with probability 1/2 in the following way. Prover 1 responds with $(1, r_1, 2, r_1)$ and prover 2 responds with $(1, s_2, 2, s_2)$. They agree and are both correct if and only if $r_1 = s_2$, which happens with probability 1/2. In this case we can see that prover 1 is using information about repetition 1 in its response to repetition 2, and prover 2 is using information about repetition 2 in its response to prover 1. In this way, they are able to cheat the verifier.

Despite this difficulty, [4] showed that the soundness does go down exponentially with k. The proof of this result is involved, so we omit it.

Theorem 28.3.3 *For every 2P1R system with soundness s, k parallel repetitions have soundness at most $(s')^k$ where s' is a constant depending only on s and the sizes of the alphabets used by the provers.*

28.3.3 Hastad's 3-Bit PCP

We now move from the issue of reducing the soundness parameter of a PCP to that of reducing the number of queries it makes. The following result is due to [3]. Again, we omit the proof.

Theorem 28.3.4 (Hastad's 3-Bit PCP) *$NP = PCP_{1-\epsilon,1/2+\epsilon}(O(\log n), 3)$ for all constants $\epsilon > 0$. Moreover, the PCP verifier for NP operates by choosing three positions i_1, i_2, i_3 of the proof and a bit b according to some distribution, reading the three bit values $y_{i_1}, y_{i_2}, y_{i_3}$, and accepting if and only if $y_{i_1} \oplus y_{i_2} \oplus y_{i_3} = b$.*

Recall that we obtained a 3-bit PCP characterization of NP in Observation 28.1.5. However, the present result has a soundness parameter that is close to 1/2, which is much better than the $1 - 1/n^{O(1)}$ soundness parameter in that simple protocol.

The linear nature of the tests run by Hastad's 3-bit PCP intimately connect it to the following constraint satisfaction problem.

Definition 28.3.5 (MAX-3LIN) *Given a system of linear equality constraints over $GF(2)$ where each constraint involves exactly three variables, find a solution that maximizes the number of satisfied constraints.*

Theorem 28.3.6 *The following are equivalent.*

1) *Theorem 28.3.4.*

2) *For all constants $\epsilon > 0$ there is a reduction from 3SAT (or any NP-complete problem) to MAX-3LIN that maps satisfiable formulas to MAX-3LIN instances where a $1 - \epsilon$ fraction of the constraints can be satisfied simultaneously, and maps unsatisfiable formulas to MAX-3LIN instances where no assignment satisfies more than a $1/2 + \epsilon$ fraction of the constraints.*

Proof: The proof is similar to Theorem 28.2.1. We first prove that statement 2 implies statement 1. Fix $\epsilon > 0$, and suppose the reduction specified in statement 2 exists. We can design a $PCP_{1-\epsilon,1/2+\epsilon}(O(\log n), 3)$ verifier for 3SAT by having the verifier perform the reduction on its input formula φ to obtain a system of equations over $GF(2)$. The proof is assumed to contain an assignment to the variables of this system. The verifier picks one of the constraints uniformly at random, queries the 3 bits of the proof specifying the values of the variables in that constraint, and accepts if and only if the constraint is satisfied.

If φ is satisfiable then there is an assignment for the system of equations that satisfies at least a $1 - \epsilon$ fraction of them. The prover can provide this assignment, and the verifier will accept with probability at least $1 - \epsilon$. If φ is not satisfiable then every assignment to the variables of the system, and in particular the assignment given by any proof, violates at least a $1/2 - \epsilon$ fraction of the constraints. Thus the verifier's test passes with probability at most $1/2 + \epsilon$.

We now prove that statement 1 implies statement 2. Fix $\epsilon > 0$, and suppose that the characterization of NP given in Theorem 28.3.4 holds. Then in particular, there is a $PCP_{1-\epsilon,1/2+\epsilon}(O(\log n), 3)$ verifier V for 3SAT. This verifier chooses its three query locations nonadaptively.

We design a reduction from 3SAT to MAX-3LIN satisfying the property of statement 2. Given a formula φ, we generate a system of linear equations over $GF(2)$ with a variable for each of the polynomially many bits of the proof that V could ever query on input φ. We run over all random strings R of length $O(\log n)$, determine which three bits of the proof V queries for that choice of R, and determine which of the eight settings of those variables cause V to accept. Since V's test is linear, this function can be expressed as a MAX-3LIN constraint. The collection of these constraints over all random strings R forms our MAX-3LIN instance.

If φ is satisfiable then there is a proof such that at least a $1-\epsilon$ fraction of the choices of R lead to acceptance. Thus setting the variables of our system according to this proof satisfies at least a $1-\epsilon$ fraction of the constraints. If φ is not satisfiable, then each assignment to the variables of the system yields a proof that causes V to accept for at most a $1/2+\epsilon$ fraction of the random strings R. Thus no assignment to the variables of the system satisfies more than a $1/2+\epsilon$ fraction of the constraints. ■

Theorem 28.3.6 reveals why Hastad's 3-bit PCP does not have a completeness parameter of 1 — if it did, then we could reduce all of NP to the problem of determining whether a system of linear equations over $GF(2)$ is consistent. This can be solved in polynomial time by Gaussian elimination, so we would have $P = NP$.

With Hastad's 3-bit PCP and Theorem 28.3.6 in hand, we can now obtain a strong inapproximability result for MAX-3SAT. Since there exists a 7/8-approximation algorithm for MAX-3SAT, the following result essentially closes the case on the approximability of MAX-3SAT.

Corollary 28.3.7 *For all constants $\epsilon' > 0$, MAX-3SAT is NP-hard to approximate within a factor of $7/8+\epsilon'$.*

Proof: By Hastad's 3-bit PCP and Theorem 28.3.6, a reduction of the type given by statement 2 exists for all $\epsilon > 0$. We give a gap-preserving reduction from MAX-3LIN to MAX-3SAT. A linear constraint on three variables over $GF(2)$ is a function that happens to evaluate to 1 for exactly four of the eight settings of its variables, so the CNF representation of this function is a collection of four clauses with three variables each. The conjunction of all these clauses over all the constraints of the given MAX-3LIN instance forms our MAX-3SAT instance. Note that the two instances have the same set of variables.

Assignments that satisfy at least a $1-\epsilon$ fraction of the linear constraints satisfy at least a $1-\epsilon$ fraction of the clauses of our 3CNF formula, since they satisfy all the clauses in blocks corresponding to satisfied constraints. Assignments that violate at least a $1/2-\epsilon$ fraction of the linear constraints violate at least a $(1/2-\epsilon)/4 = 1/8-\epsilon/4$ fraction of the clauses of our 3CNF formula, since they violate at least one of the four clauses in each of the blocks corresponding to violated constraints. Given a 3CNF formula in which at least a $1-\epsilon$ fraction of the clauses can be satisfied, a $(7/8+\epsilon')$-approximation algorithm for MAX-3SAT would find an assignment satisfying at least a $(1-\epsilon)(7/8+\epsilon') = 7/8+(\epsilon'-\epsilon(7/8+\epsilon'))$ fraction of the clauses. This fraction is greater than $1-(1/8-\epsilon/4)$ provided we choose ϵ small enough. Thus a $(7/8+\epsilon')$-approximation algorithm for MAX-3SAT would allow us to distinguish between instances of MAX-3LIN where a $1-\epsilon$ fraction of the constraints can be satisfied and those where at most a $1/2+\epsilon$ fraction can be satisfied, for some constant $\epsilon > 0$. ■

In the next lecture, we will see how to use Hastad's 3-bit PCP to show that it is NP-hard to approximate Vertex Cover within any constant factor less than 7/6. We will also see that Set

Cover cannot be approximated within a factor of $b \cdot \log n$ for some constant b, under an assumption slightly stronger than $P \neq NP$.

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, M. Szegedy. Proof Verification and the Hardness of Approximation Problems. In *Journal of the ACM*, 45(2), pp. 501-555, 1998.

[2] A. Arora, S. Safra. Probabilistic Checking of Proofs: A New Characterization of NP. In *Journal of the ACM*, 45(1), pp. 70-122, 1998.

[3] J. Hastad. Some Optimal Inapproximability Results. In *Journal of the ACM*, 48(4), pp. 798-859, 2001.

[4] R. Raz. A Parallel Repetition Theorem. In *SIAM Journal on Computing*, 27(3), pp. 763-803, 1998.