Recall the definition of a *probabilistically checkable proof* (or "PCP") from last lecture.

Definition 22.1 *We say a language L is in the complexity class* $\mathbf{PCP}(r(n), q(n))$ *if there is a poly-time randomized verifier V such that for any $x \in \{0,1\}^*$, if we let n denote $|x|$ then*

1. *On input $\langle x, \pi \rangle$, V reads x, tosses $r(n)$ coins, reads $q(n)$ bits of π, and then decides whether to accept or reject.*
2. *Completeness: if $x \in L$, then there exists $\pi \in \{0,1\}^{\mathrm{poly}(n)}$ such that $\alpha \stackrel{\text{def}}{=} \mathbb{P}(V(x,\pi) = 1) = 1$.*
3. *Soundness: if $x \notin L$, then for all $\pi \in \{0,1\}^{\mathrm{poly}(n)}$, we have $\rho \stackrel{\text{def}}{=} \mathbb{P}(V(x,\pi) = 1) \leq 1/2$.*

The string π above is typically called the "proof" (since its purpose is to prove to V that x is in L). In this regard, it is synonymous with what is typically called the "witness" in the definition of **NP**. Now recall the PCP theorem from last lecture.

Theorem 22.2 (PCP Theorem [AS98, ALM+98]) *There is a universal constant $q > 0$ such that* $\mathbf{NP} = \mathbf{PCP}(O(\log n), q)$.

The PCP theorem was originally proven by giving a verifier which reads $q = O(1)$ bits from π without specifying exactly what the constant q is (though it was apparently rumored to be about 10^6 [O'D05]). After the PCP theorem was proven, several works attempted to understand the smallest q achievable. In [BGLR94], it was shown one can achieve $q = 29$, then $q = 22$ in [FK94], then $q = 16$ in [BGS98]. The following was then proven by Håstad.

Theorem 22.3 ([Hås01]) *For any fixed constants $\varepsilon, \delta \in (0,1)$, there exists a PCP for* SAT *with $q = 3$, completeness $\alpha > 1 - \varepsilon$ and soundness $\rho < 1/2 + \delta$. Furthermore, after the verifier V reads three bits $\pi_{i_1}, \pi_{i_2}, \pi_{i_3}$ of the proof, V accepts iff $\pi_{i_1} \oplus \pi_{i_2} \oplus \pi_{i_3} = b$ for some b depending on x and its random coin flips ($\oplus$ denotes XOR).*

22.1 Equivalence of the PCP theorem and hardness of approximation

Notice that Håstad's PCP immediately implies hardness of approximation for the problem MAX3LIN2. In this optimization problem, we are given a collection of m linear equations over n variables. The variables take values in $\{0,1\}$, and we interpret addition as being modulo 2 (which is equivalent to XOR). Furthermore, each equation

involves exactly three variables. Thus, each equation is of the form $x_{i_1} \oplus x_{i_2} \oplus x_{i_3} = b$. (In general, in the MAXkLINp problem we have linear equations each with k variables, with variables taking values in $\{0,\ldots,p-1\}$ with all arithmetic performed modulo p.) The goal of MAX3LIN2 problem is to assign the variables to $\{0,1\}$ so as to maximize the number of linear equations satisfied. Letting OPT denote this maximum number divided by m, i.e. the maximum *fraction* of equations that can be satisfied by an assignment, we first observe that there is a simple *randomized* algorithm which satisfies at least $\mathsf{OPT}/2$ equations in expectation, i.e. achieving a $(1/2)$-approximation. The algorithm is simple: pick an assignment uniformly at random! If we define X_i to be an indicator random variable for the event that equation i is satisfied, then $X = (1/m)\sum_{i=1}^{m} X_i$ is the fraction of satisfied equations. Then

$$\begin{aligned}
\mathbb{E}X &= \frac{1}{m}\sum_{i=1}^{m} \mathbb{E}X_i \text{ (linearity of expectation)} \\
&= \frac{1}{m}\sum_{i=1}^{m} \mathbb{P}(\text{equation } i \text{ is satisfied}) \\
&= \frac{1}{m}\sum_{i=1}^{m} \frac{1}{2} \\
&= \frac{1}{m}\cdot\frac{m}{2} \\
&\geq \frac{1}{m}\cdot\frac{\mathsf{OPT}}{2} \\
&= \frac{1}{2}\cdot\mathsf{OPT}.
\end{aligned}$$

Håstad's PCP above implies that approximating MAX3LIN2 with approximation factor $1/2+\gamma$ for *any* constant $0 < \gamma \leq 1/2$ is **NP**-hard! We will show that if we *could* perform such an approximation in polynomial time, then we could solve SAT in polynomial time. Consider a SAT instance φ. We will use Håstad's PCP with his verifier V, which takes in proofs π of size at most Cn^c for some constants $C, c > 0$. We set $\varepsilon = \gamma/2, \delta = (3/4)\gamma - \gamma^2 > 0$. We now consider the following MAX3LIN2 instance with $N = Cn^c$ variables $x_1,\ldots,x_N$. These variables are supposed to represent the bits of a proof π. Recall the verifier V tosses $R = O(\log n)$ bits. For each $r \in \{0,1\}^R$, we include a linear equation in our MAX3LIN2 instance $x_{i_1} \oplus x_{i_2} \oplus x_{i_3} = b$, where i_1, i_2, i_3, b are as V would have chosen if its coin tosses had yielded r. This completes the description of our reduction from φ to a MAX3LIN2 instance. Note the reduction runs in polynomial time since V runs in polynomial time (so we can feed it φ, r to determine i_1, i_2, i_3, b quickly), and the number of bitstrings r we have to loop over to generate all the equations is $2^R \leq poly(n)$. We then ask our algorithm approximating MAX3LIN2 to approximate OPT: if it says $\mathsf{OPT} \geq 1/2 + (3/4)\gamma - \gamma^2$, we output that φ is satisfiable. Otherwise we say that it is unsatisfiable.

Now we argue about correctness of the reduction. If φ is satisfiable, then there exists a proof π so that V accepts with probability $\alpha > 1 - \gamma/2$ for a random $r \in \{0,1\}^R$. In other words, if we set $x = \pi$, then at least an α-fraction

of the linear equations are satisfied by x. Thus our approximation algorithm with approximation factor $1/2+\gamma$ will satisfy at least $(1-\gamma/2)(1/2+\gamma) = 1/2+(3/4)\gamma-\gamma^2$ equations, which is strictly larger than ρ by our choice of δ. If φ is unsatisfiable, then soundness guarantees that *no* proof π causes V to accept with probability $1/2+\delta$ or larger. Since x can always be set to equal any proof, this shows that no setting of x satisfies a $(1/2+\delta)$-fraction of equations or more. Thus, even if our approximation algorithm managed to achieve OPT in this case, we would satisfy strictly less than a $1/2+(3/4)\gamma-\gamma^2$ fraction of the equations, and thus correctly declare that φ is unsatisfiable.

It is worth noting that deciding whether *all* equations can be simultaneously satisfied is easy to do in polynomial time! For example, one could use gaussian elimination.

As it turns out, the fact that Håstad's PCP implied hardness of approximation for some optimization problem is not an accident. As we will see now, the PCP theorem *itself* is equivalent to a statement on hardness of approximation. We here follow the treatment of this topic in the computational complexity book of Arora and Barak [AB09].

Definition 22.4 *In $q\mathsf{CSP}_W$ (CSP stands for "Constraint Satisfaction Problem"), the input is a tuple of constraints $\varphi = (\varphi_1, \ldots, \varphi_m)$. Each constraint φ_i is allowed to be an arbitrary function mapping $[W]^n$ to $\{0,1\}$, where $[W] \stackrel{\text{def}}{=} \{0,\ldots,W-1\}$, such that for each i $\varphi_i(x)$ depends only on q entries of the vector x. Each φ_i is specified by its truth table, which is of size W^q. We use $q\mathsf{CSP}$ to denote $q\mathsf{CSP}_2$. The goal in this optimization problem is to find a vector $x \in [W]^q$ so as to maximize the fraction of constraints satisfied. We let $\mathsf{OPT}(\varphi)$ denote the maximum possible fraction of constraints that can be simultaneously satisfied in φ.*

Note that $\mathsf{MAX3SAT}$ is a special case of $3\mathsf{CSP}_2$. In $\mathsf{MAX3SAT}$, we would like to satisfy the maximum number (or fraction) of clauses, where each clause is the disjunction of three literals. Thus for each i, the ith clause gives rise to a function $\varphi_i : \{0,1\}^n \to \{0,1\}$ that only depends on 3 entries of an assignment x (namely the three entries corresponding to the variables that appear in clause i). $\varphi_i(x) = 1$ iff x satisfies clause i.

Definition 22.5 *ρ-gap-$q\mathsf{CSP}_W$ is a promise decision problem where we are given an input φ with the promise that* either

1. $\mathsf{OPT}(\varphi) = 1$, or
2. $\mathsf{OPT}(\varphi) < \rho$

and we must decide which. If φ does not fall in either case (i.e. if $\rho \leq \mathsf{OPT}(\varphi) < 1$), our algorithm is allowed to output anything.

Theorem 22.6 *The PCP theorem is equivalent to the statement that* ρ*-gap-*$q\mathsf{CSP}$ *is* **NP***-hard for some constants* ρ, q*.*

Proof: PCP theorem $\Longrightarrow$ NP-hardness of $(1/2)$-gap-$q\mathsf{CSP}$: We reduce SAT to ρ-gap-$q\mathsf{CSP}$. By the PCP theorem, there is a PCP system for SAT with some verifier V who reads input φ, flips $R = O(\log n)$ coins, and does poly$(|\varphi|)$ computation before making queries to a proof $\pi \in \{0,1\}^N$ for some $N \leq poly(n)$. To decide whether $\varphi \in \mathsf{SAT}$, it suffices to discover whether there is a proof that causes V to accept with probability larger than $1/2$, over the randomness of choosing r. For each $r \in \{0,1\}^R$ we create a function $\varphi'_{x,r} : \{0,1\}^N \to \{0,1\}$. $\varphi'_{x,r}(\pi) = 1$ iff V would accept π after flipping random string r; note that $\varphi'_{x,r}$ only depends on at most q bits of π for any r. Thus our $q\mathsf{CSP}$ instance is the collection $\varphi'_x = (\varphi'_{x,r})_{r\in\{0,1\}^R}$, and this reduction takes polynomial time since V runs in polynomial time (to allow us to quickly determine the truth tables for the $\varphi'_{x,r}$), and the number of r we have to consider is $m = 2^R \leq poly(n)$. Correctness of the reduction follows by completeness and soundness of the PCP for SAT: if $x \notin L$ then there is no way to fill in the bits of the proof π to satisfy half of the constraints (i.e. at least half of the random strings r cause V to reject), and if $x \in L$ then there is a proof which causes V to accept for any r.

NP-hardness of ρ-gap-$q\mathsf{CSP}$ $\Longrightarrow$ PCP theorem: Let L be a language in **NP** so there is some poly-time reduction f from L to ρ-gap-$q\mathsf{CSP}$ for some $\rho \in (0,1)$. This reduction gives us some $q\mathsf{CSP}$ instance $\varphi = (\varphi_1, \ldots, \varphi_m)$. Our PCP for L is then as follows. The verifier first runs this reduction f on the input x to obtain φ. It expects to be given a proof π which is simply an assignment to the variables in φ. It then flips $\lceil \log_2 m \rceil = O(\lg(|x|))$ coins ($m \leq poly(n)$ since f is a poly-time reduction) to pick a random constraint i, then checks that the assignment in π satisfies φ_i. By correctness of the reduction f, this yields perfect completeness and soundness ρ. For $\rho > 1/2$, the soundness can be reduced to $1/2$ as in the PCP theorem stated above by the verifier picking $k = \lceil 1/\log(1/\rho) \rceil + 1$ constraints independently at random and checking that all are satisfied. This increases the number of queries by V from q to $qk = O(q)$. In a no instance, the probability that π satisfies all k constraints is at most $\rho^k < 1/2$. ∎

We can also define a gap version of $3\mathsf{SAT}$, and in fact it is possible to show that the PCP theorem is equivalent to the statement that there exists some constant $\rho \in (0,1)$ such that ρ-gap-$3\mathsf{SAT}$ is **NP**-hard. We show one direction below.

Definition 22.7 ρ*-gap-*$3\mathsf{SAT}$ *is a promise decision problem where we are given a 3-CNF formula* φ *with the promise that* either

1. $\mathsf{OPT}(\varphi) = 1$, or
2. $\mathsf{OPT}(\varphi) < \rho$

and we must decide which. If φ does not fall in either case (i.e. if $\rho \leq \mathsf{OPT}(\varphi) < 1$), our algorithm is allowed to output anything.

Theorem 22.8 *There exists $\rho \in (0,1)$ such that ρ-gap-3SAT is* **NP***-hard.*

Proof: We give a polynomial time reduction from ρ-gap-qCSP, which is **NP**-hard for some constants q, ρ by the PCP theorem, to ρ'-gap-3SAT. In this reduction, if $\rho = 1-\varepsilon$, then $\rho' = 1 - \varepsilon/(q2^q)$.

Given an input $\varphi = (\varphi_1, \ldots, \varphi_m)$ for ρ-gap-qCSP, for each i we know that $\bar{\varphi}_i$ depends on at most q variables (where $\bar{f}$ denotes the negation of function f, i.e. $1-f$). Thus we can write $\bar{\varphi}_i$ in disjunctive normal form (DNF), i.e. as an OR of clauses where each clause is an AND of literals, and furthermore each clause contains exactly q literals. The number of clauses is at most 2^q. Thus by De Morgan's laws, we can write $\varphi_i = \bar{\bar{\varphi}}_i$ as a q-CNF (an AND of ORs) with at most 2^q clauses. We can then transform the q-CNF to a 3-CNF by increasing the number of clauses by a factor of at most q. For example, a clause that was formerly

$$(a \vee b \vee c \vee d \vee e)$$

can be replaced by

$$(a \vee b \vee x) \wedge (\bar{x} \vee c \vee y) \wedge (\bar{y} \vee d \vee e).$$

We leave it as an exercise to the reader to verify that after performing the above transformation to all clauses in a q-CNF, the original q-CNF is satisfiable iff the transformed version is also satisfiable. The above gives us a 3-CNF φ'_i that is satisfiable iff φ_i is. We then define $\varphi' = \bigwedge_{i=1}^{m} \varphi'_i$.

Now, if φ is a YES instance, then all the φ'_i are simultaneously satisfiable, so φ' is a satisfiable 3SAT instance. If φ is a NO instance, then no assignment to the variables can satisfy more than an ε-fraction of the constraints φ_i (where $\varepsilon = 1-\rho$). In other words, at least one clause in φ'_i must not be satisfied for an ε-fraction of the formulae φ'_i. Since each φ'_i has at most $q2^q$ clauses, this implies that no more than an $\varepsilon/(q2^q)$-fraction of clauses in φ' can be satisfied by any assignment, giving the theorem statement for $\rho' = 1 - \varepsilon/(q2^q)$ as desired. ■

22.2 Label cover, parallel repetition, and 2-prover 1-round games

Although the PCP theorem provides us with **NP**-hardness of various gap decision problems (or also, hardness of approximation for various optimization problems), the constant factors gaps ρ that come out of the PCP theorem tend to be quite weak. In this section we describe a theorem of Raz, the "parallel repetition theorem" [Raz98], which shows how to get better gaps for many problems out of the PCP theorem.

Before doing so, we define the "label cover" problem introduced in [AL97].

Definition 22.9 *In the* $\mathsf{LabelCover}(\Sigma)$ *problem, we are given as input a bipartite graph* $G = (V,E)$ *with bipartition* (L,R) *(the "left" and "right" vertex sets). The graph is left-regular; that is, all vertices in L have the same degree. Furthermore, in the input for each edge* $e = (u,v)$ *we are given a constraint* $\Pi_e : \Sigma \to \Sigma$*. For any assignment* $\sigma : V \to \Sigma$ *of elements of the alphabet* Σ *to the vertices, we say a constraint* Π_e *for* $e = (u,v)$ *is satisfied if* $\sigma(v) = \Pi_e(\sigma(u))$*. For an input* φ*, we let* $\mathsf{OPT}(\varphi)$ *denote the maximum fraction of edges that can be satisfied with an assignment. Then in the* ρ*-gap-*$\mathsf{LabelCover}(\Sigma)$ *problem, we are given as input some* φ *with the promise that* either

1. $\mathsf{OPT}(\varphi) = 1$, or
2. $\mathsf{OPT}(\varphi) < \rho$

and we must decide which.

Lemma 22.10 *There is a constant* $\rho \in (0,1)$ *such that* ρ*-gap-*$\mathsf{LabelCover}(\Sigma)$ *is* **NP***-hard.*

Proof: We reduce from ρ'-gap-3SAT for a constant ρ' such that ρ'-gap-3SAT is **NP**-hard. For a 3SAT instance φ with clauses $C_1,\ldots,C_m$ and variables $x_1,\ldots,x_n$, we create a bipartite graph with m vertices on the left and n vertices on the right, corresponding to the clauses and vertices, respectively. An edge exists from C_1 to x_j iff variable x_j participates in clause C_i. The alphabet Σ is $\{0,1,\ldots,6\}$. An assignment $\sigma : V \to \Sigma$ should be interpreted as follows. Each clause in a 3SAT instance can be satisfied in one of 7 ways, and the label $\sigma(v)$ of a vertex $v \in L$ tells us *which* of these 7 ways it would like to be satisfied. That is, if a clause C looks like, say, $x_1 \vee \bar{x_4} \vee x_7$, then there are $2^3 = 8$ ways to assign values to x_1,x_4,x_7, and all but one of these ways satisfies C (namely the assignment $(x_1,x_4,x_7) = (0,1,0)$ fails to satisfy C). Labels $\sigma(v)$ for variable vertices $v \in \mathbf{R}$ should be 0 or 1, corresponding to whether that variable should be set to 0 or 1. Then for an edge e from clause C_i to variable x_j, the function Π_e is simply a consistency check; for each of the 7 possible labels ℓ for C_i, $\Pi_e(\ell) \in \{0,1\}$ is the value for variable x_j which is consistent with the satisfying assignment for C_i that corresponds to label ℓ.

Note that in a YES instance for φ, there is a labeling to the vertex set such that all Π_e are satisfied. In a NO instance, no assignment can satisfy more than an ε'-fraction of clauses in φ for $\varepsilon' = 1 - \rho'$. Thus for any unsatisfied clause C, at least one of the three edges leaving C must fail our consistency test, implying at most an $\varepsilon'/3$-fraction of edges in the LabelCover instance can be satisfied. Thus we conclude ρ-gap-$\mathsf{LabelCover}(\Sigma)$ is **NP**-hard for $\rho = 1 - \varepsilon/3$. ∎

Although Lemma 22.10 tells us that ρ-gap-LabelCover(Σ) is **NP**-hard, the ρ that comes out of the proof is extremely close to 1. This is where parallel repetition [Raz98] comes in. Parallel repetition tells us that for any constant $\varepsilon > 0$, there exists a Σ such that ρ-gap-LabelCover(Σ) is **NP**-hard (in fact $|\Sigma|$ will be at most $poly(1/\varepsilon)$).

Before stating the parallel repetition theorem, we first introduce 2**-prover** 1**-round games**.

Definition 22.11 *In a* 2*-prover* 1*-round game $\mathcal{G}$, there are two provers P_1 and P_2, as well as:*

- *An answer set A.*
- *A set of questions X that can be asked to P_1 (and Y to P_2).*
- *A probability distribution μ on $X \times Y$.*
- *A predicate $V : X \times Y \times A \times A \to \{0,1\}$.*
- *Strategies $f_1 : X \to A$ and $f_2 : Y \to A$.*

In this game a verifier picks $(x,y) \in X \times Y$ distributed according to μ then sends x to P_1 and y to P_2. The verifier then accepts iff $V(x,y,f_1(x),f_2(y)) = 1$, in which case we say the provers win. If $V(x,y,f_1(x),f_2(y)) = 0$, then the provers lose. The value $\mathsf{val}(\mathcal{G})$ denotes the maximum possible probability of winning (under distribution μ) if P_1, P_2 pick their strategies f_1, f_2 optimally.

Note that an instance G of LabelCover(Σ) can be viewed as a 2-prover 1-round game. Specifically $A = \Sigma, X = L, Y = R$. The distribution μ is the distribution on $L \times R$ induced by picking a random edge $e \in E$. The strategies f_1, f_2 are simply restrictions of $\sigma : V \to \Sigma$ to L and R, respectively. The predicate $V(u,v,a,b)$ is 1 iff $b = \Pi_{(u,v)}(a)$. For this game $\mathcal{G}$, we then have $\mathsf{val}(\mathcal{G}) = \mathsf{OPT}(G)$. Thus for ρ-gap-LabelCover(Σ), either $\mathsf{val}(\mathcal{G}) = 1$ (if the instance is satisfiable), or it is at most ρ. In the latter case, essentially the provers are trying to pick assignments, or "strategies f_1, f_2", so as to maximize the probability that they trick the verifier into thinking that the instance is satisfiable.

As we saw with LabelCover, the gap for which hardness follows from the PCP theorem is quite small (i.e. ρ is very close to 1). Fortnow, Rompel, and Sipser then proposed *parallel repetition* of a game as a way to increase the gap, i.e. to decrease ρ [FRS88]. In the k-fold parallel repetition $\mathcal{G}^{\otimes k}$ of a game $\mathcal{G}$, the answer set becomes A^k, and the set of questions becomes X^k and Y^k for P_1, P_2, respectively. The probability distribution is the k-fold product distribution μ^k (i.e. the verifier pickers $(x_1,y_1),\ldots,(x_k,y_k)$ each independently from μ). The predicate V^k is then defined by

$$V^k((x_1,\ldots,x_k),(y_1,\ldots,y_k),(a_1,\ldots,a_k),(b_1,\ldots,b_k)) = \bigwedge_{i=1}^{k} V(x_i,y_i,a_i,b_i).$$

For label cover, this corresponds to converting an instance G into a new instance $G^{\otimes k}$ with left vertex set L^k, right vertex set R^k, alphabet Σ^k, and where $e = ((u_1,\ldots,u_k),(v_1,\ldots,v_k))$ is in the edge set of $G^{\otimes k}$ iff $(u_i,v_i) \in E$ for each $i \in \{1,\ldots,k\}$. Furthermore, we would define $\Pi_e^{\otimes k} : \Sigma^k \to \Sigma^k$ by $\Pi_e^{\otimes k}(\sigma_1,\ldots,\sigma_k) = (\Pi_{(u_1,v_1)}(\sigma_1),\ldots,\Pi_{(u_k,v_k)}(\sigma_k))$.

The intuition for why parallel repetition might decrease the value of the game is as follows. If success of the two provers across the k repetitions had been independent, then we would have exponential decay in the value: $\mathsf{val}(\mathcal{G}^{\otimes k}) = \mathsf{val}(\mathcal{G})^k$. Unfortunately there is no reason for them to be independent. Even though the (x_i,y_i) are independent, the function $f_1^{\otimes k}$ in the repeated game is allowed to be a function of the entire tuple $(x_1,\ldots,x_k)$ (and similarly for f_2). Thus each prover can choose a strategy that bases its answer on the entire set of questions, rather than answering each question individually. In this way, they can hope to correlate their failure to answer questions well across the k repetitions. And in fact, as we will now see via an example of Feige, it is possible for them to do so to their advantage!

Consider the following coin game $\mathcal{C}$. The answer set is $A = \{P_1,P_2\} \times \{0,1\}$. The question sets are $X = Y = \{0,1\}$. The distribution μ is uniform on $X \times Y$. The predicate V is such that $V(x,y,(P,a),(P',b)) = 1$ iff either ($P = P' = P_1$ and $a = b = x$) or ($P = P' = P_2$ and $a = b = y$). In other words, the verifier flips two fair coins c_1,c_2 and sends c_i to prover P_i. The provers then have to, without communicating with each other, agree on the same prover and correctly guess the coin flip of that prover.

Lemma 22.12 $\mathsf{val}(\mathcal{C}) = 1/2$

Proof: Achieving value $1/2$ is easy: the two provers simply always output $(P_1,0)$. In this way the two provers always agree on the same prover, namely P_1, and P_1's coin is 0 with probability $1/2$.

Note also that the value cannot possibly be larger than $1/2$, since one of the two provers must guess the coin value of the other prover to succeed with absolutely no information about their coin, so they cannot do better than random guessing. ■

If parallel repetition had behaved perfectly the way we wanted, we would have $\mathsf{val}(\mathcal{C}^{\otimes 2}) = 1/4$. However, this is not the case.

Lemma 22.13 $\mathsf{val}(\mathcal{C}^{\otimes 2}) = 1/2$

Proof: For a similar reason as above, $\mathsf{val}(\mathcal{C}^{\otimes 2})$ cannot be larger than $1/2$.

Now we show that $1/2$ is achievable. The two provers P_1,P_2 play using the following strategy. In the first round, they both guess the coin value for P_1's first coin. In the second round, they both guess the coin value for P_2's

second coin. Furthermore, in these guesses, they both assume that a probabilistic event $\mathcal{E}$ occurred, defined as the event that P_1's first had the same outcome as P_2's second coin. Note that they can carry out this strategy: for their first answer, P_1 knows his own coin and returns that, and P_2 knows his second coin so he guesses that for P_1's first coin (and similarly for the second round of the game). This pair of strategy causes the verifier to accept exactly when event $\mathcal{E}$ occurs, but $\mathbb{P}(\mathcal{E}) = 1/2$, so $\mathsf{val}(\mathcal{C}^{\otimes 2}) = 1/2$. ■

From the above example we see that parallel repetition does not perfectly reduce the value of the game, but nevertheless, Raz showed that it *almost* does in [Raz98] (the proof was later simplified by Holenstein [Hol09]).

Theorem 22.14 (Parallel repetition theorem [Raz98]) *For any* 2*-prover* 1*-round game* $\mathcal{G}$ *with* $\mathsf{val}(\mathcal{G}) < 1-\alpha$*, there exists a constant* $c > 0$ *depending only on the game (specifically, depending only on* $|A|$ *and* α*) such that for all integers* $k > 0$

$$\mathsf{val}(\mathcal{G}^{\otimes k}) \leq (1-\alpha^3)^{ck}.$$

For example, for the coin game above even though we saw that it is *not* true that $\mathsf{val}(\mathcal{C}^{\otimes k}) \leq \mathsf{val}(\mathcal{C})^k$, it was shown by Feige that $\mathsf{val}(\mathcal{C}^{\otimes k}) \leq (1/\sqrt{2})^k$ (and furthermore, $\mathsf{val}(\mathcal{C}^{\otimes k})$ exactly equals $(1/\sqrt{2})^k$ when k is even) [Fei91]. Raz's theorem is more general, however, and applies to *any* 2-prover 1-round game (or, as relevant for us, any instance of $\mathsf{LabelCover}(\Sigma)$).

The desired parallel repetition corollary for us is the following.

Corollary 22.15 *For any* $\varepsilon > 0$ *there exists* Σ*,* $|\Sigma| \leq poly(1/\varepsilon)$*, such that* ε*-gap-*$\mathsf{LabelCover}(\Sigma)$ *is* **NP***-hard.*

It is beyond the scope of this course, but the above corollary can be used, for example, to show that Set Cover has no C-approximation for any constant C unless $\mathbf{P} = \mathbf{NP}$. It can also be used to show that, for some fixed constant $\beta > 0$, Set Cover has no $\beta \log n$-approximation unless $\mathsf{SAT} \in \mathrm{DTIME}(n^{O(\log\log n)})$. See for example the Lecture 14 and 15 notes at [Gur05].

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[AL97] Sanjeev Arora and Carsten Lund. Hardness of approximations. In Dorit Hochbaum (Ed.), editor, *Approximation Algorithms for NP-Hard Problems*, pages 399–446. PWS Publishing, Boston, MA, 1997.

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998.

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *J. ACM*, 45(1):70–122, 1998.

[BGLR94] Mihir Bellare, Shafi Goldwasser, Carsten Lund, and Alexander Russell. Efficient probabilistic checkable proofs and applications to approximation. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing (STOC)*, page 820, 1994.

[BGS98] Mihir Bellare, Oded Goldreich, and Madhu Sudan. Free bits, PCPs, and nonapproximability-towards tight results. *SIAM J. Comput.*, 27(3):804–915, 1998.

[Fei91] Uriel Feige. On the success probability of the two provers in one-round proof systems. In *Structures (CCC)*, 1991.

[FK94] Uriel Feige and Joe Kilian. Two prover protocols: low error at affordable rates. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing (STOC)*, pages 172–183, 1994.

[FRS88] Lance Fortnow, John Rompel, and Michael Sipser. On the power of multi-prover interactive protocols. In *Structures (CCC)*, 1988.

[Gur05] Venkatesan Guruswami. The PCP theorem and hardness of approximation, 2005. `http://courses.cs.washington.edu/courses/cse533/05au/`.

[Hås01] Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001.

[Hol09] Thomas Holenstein. Parallel repetition: Simplification and the no-signaling case. *Theory of Computing*, 5(1):141–172, 2009.

[O'D05] Ryan O'Donnell. A history of the PCP theorem, 2005. `http://courses.cs.washington.edu/courses/cse533/05au/pcp-history.pdf`.

[Raz98] Ran Raz. A parallel repetition theorem. *SIAM J. Comput.*, 27(3):763–803, 1998.