Sliding Scale Conjectures in PCP

Dana Moshkovitz [*]

July 30, 2019

1 Introduction

The PCP (i.e., Probabilistically Checkable Proofs) Theorem [8, 7, 22, 5, 4] states that any mathematical proof can be converted to a format that can be checked by a verifier making only a constant number of queries to the proof. The verifier picks the queries in a randomized way and might err with low probability.

Despite three highly successful decades of research on PCP, we're far from having an "ultimate" PCP theorem in which the randomness complexity of the verifier, the number of queries the verifier makes, the alphabet size of the proof, and the error probability of the verifier are all, simultaneously, as low as they could be. The lack of such a PCP theorem prevents us from obtaining optimal hardness of approximation results for problems like Closest Vector Problem, Directed Sparsest Cut and Constraint Satisfaction Problem over large alphabet.

The "Sliding Scale Conjecture" and related conjectures capture our beliefs about the PCP theorems we should aspire to, and they are the focus of this column. There are tradeoffs between the different parameters, and the conjectures ask for optimal tradeoffs. The name "Sliding Scale Conjecture" was coined by Shafi Goldwasser in talks she gave on her joint work with Bellare, Lund and Russell [11]. Goldwasser focused on the tradeoff between the alphabet size and the error probability – as the alphabet becomes larger (the "window" into the proof slides up), the error probability can be made lower. Personally, I'd like to think of the *error probability* ε as doing the "sliding", and have the rest of the parameters adjusted accordingly. In this column I'll describe the conjectures, their applications and where current research stands with respect to them.

2 The Conjectures

Without further ado, let us state the main conjecture:

Conjecture 2.1 (Sliding Scale Conjecture [11])**.** *There exists $\varepsilon_0 = \varepsilon_0(n) = 1/\operatorname{poly}(n)$, so for any $\varepsilon \geq \varepsilon_0$, there exists Σ of size $\operatorname{poly}(1/\varepsilon)$, such that every NP language L has a PCP verifier that on input x of size n uses $r = O(\log n)$ random bits to make $q = O(1)$ queries to a proof π over alphabet Σ, and:*

- Completeness: *If $x \in L$, then there exists π, such that the verifier always accepts.*

[*] danama@cs.utexas.edu. Department of Computer Science, UT Austin, USA. This material is based upon work supported by the National Science Foundation under grants number 1218547 and 1648712.

- Soundness: *If $x \notin L$, then for any π, the probability that the verifier accepts is at most ε.*

To clarify the parameters: As usual in PCP, the focus is on verifiers that use only $r = O(\log n)$ random bits where n is the size of the input x. The reason is that it implies a standard NP verifier with a poly(n)-sized proof by enumerating all possible settings of the random bits. The lowest the error probability ε can be is exponentially small in the randomness r, which is polynomially small in n. The number of queries q should be a universal constant, independent of the error ε, preferably the minimum 2 queries. It is easy to amplify existing PCP Theorems to achieve error ε using $\Theta(\log(1/\varepsilon))$ queries, but then the number of queries is not a universal constant. the alphabet size must be at least polynomially large in $1/\varepsilon$, since a random assignment satisfies the verifier with probability $1/|\Sigma|^q$.

The paper [11] came out in 1993, shortly after the PCP Theorem was first proved. In the years since, researchers perfected techniques for proving optimal inapproximability results based on the PCP Theorem [10, 24]. PCP of a certain type ("projection PCP") is the basis of those optimal results. Projection PCPs have two queries, and the answer to the first query uniquely determines a satisfying answer to the second query, but not vice versa[1]. The optimization problem associated with such PCPs is called LABEL-COVER:

Definition 1 (Label-Cover). *The input is a bipartite graph $G = (X, Y, E)$, alphabets Σ_X, Σ_Y, functions $\{f_e : \Sigma_X \to \Sigma_Y\}_{e \in E}$ and sets $\{V_x \subseteq \Sigma_X\}_{x \in X}$. We say that assignments $A : X \to \Sigma_X$, $B : Y \to \Sigma_Y$ satisfy an edge $e = (x, y) \in E$ if $A(x) \in V_x$ and $f_e(A(x)) = B(y)$. The goal is to find assignments to the vertices that satisfy as many edges as possible.*

The *Projection Games Conjecture* is that the Sliding Scale Conjecture holds even for projection PCP. In other words, LABEL-COVER is NP-hard to approximate as follows:

Conjecture 2.2 (Projection Games Conjecture, [31]). *There exists $\varepsilon_0 = \varepsilon_0(n) = 1/\operatorname{poly} n$, so for any $\varepsilon \geq \varepsilon_0$, there are Σ_X, Σ_Y of size* poly$(1/\varepsilon)$*, such that it is NP-hard given* LABEL-COVER *on inputs of size n with alphabets Σ_X, Σ_Y, to distinguish whether all edges can be satisfied of at most ε fraction can be satisfied.*

One may also ask for a version of the Projection Games Conjecture where the reduction from SAT is very efficient:

Conjecture 2.3 (Projection Games Conjecture, almost linear size [31]). *For any $\varepsilon = \varepsilon(n) > 0$,* SAT *on input φ of size n can be reduced to* LABEL-COVER *on inputs of size $n^{1+o(1)}$* poly$(1/\varepsilon)$ *with alphabets Σ_X, Σ_Y of size* poly$(1/\varepsilon)$*, such that: If φ is satisfiable, then there exists an assignment that satisfies all edges in* LABEL-COVER*, while if φ is not satisfiable, then at most ε fraction of the edges can be satisfied.*

It is believed that SAT on input of size n requires exponential time $2^{\Omega(n)}$ ("The Exponential Time Hypothesis" [26]). Under this assumption, Conjecture 2.3 shows that approximating LABEL-COVER up to factor ε requires nearly exponential time.

In hardness of approximation reductions, one typically starts with a LABEL-COVER instance and then encodes the alphabet symbols with some code of special structure. To use $\varepsilon = 1/\operatorname{poly}(n)$ in such reductions one might want to use the Hadamard code, which requires the projections to correspond to linear functions. Since there is an efficient algorithm that solves a fully satisfiable system of linear equations, in this case there must be completeness error too:

[1]In *unique* PCP, the answer to each one of the two queries uniquely determines a satisfying answer to the other query. This is a more restricted PCP, whose existence with parameters analogous to what is known for projection PCP is only conjectured [28].

Conjecture 2.4 (Linear Projection Games Conjecture, [31])**.** *There exist $\varepsilon_0 = \varepsilon_0(n) = 1/\operatorname{poly} n$ and $\delta_0 = \delta_0(n) = 1/\operatorname{poly} n$, so for any $\varepsilon \geq \varepsilon_0$ and $\delta \geq \delta_0$, there are Σ_X, Σ_Y that are vector spaces of size that depends on $1/\varepsilon, 1/\delta$, such that it is NP-hard given* Label-Cover *on inputs of size n with alphabets Σ_X, Σ_Y and $\{V_x\}_x$, $\{f_e\}_{e\in E}$ that are linear, to distinguish whether $1-\delta$ fraction of the edges can be satisfied or at most ε fraction can be satisfied.*

For some applications it is useful to have certain structural properties on the graph G underlying Label-Cover. The structural properties include: regularity, low degree and pseudo-random structure:

Conjecture 2.5 (Projection Games Conjecture, structured graph [31])**.** *There exists $\varepsilon_0 = \varepsilon_0(n) = 1/\operatorname{poly} n$, so for any $\varepsilon \geq \varepsilon_0$, there are Σ_X, Σ_Y of size $\operatorname{poly}(1/\varepsilon)$, such that it is NP-hard given* Label-Cover *on inputs of size n with alphabets Σ_X, Σ_Y and a bi-regular bipartite expander graph G whose degrees are at most $\operatorname{poly}(1/\varepsilon)$, to distinguish whether all edges can be satisfied or at most ε fraction can be satisfied.*

We remark that there are transformations that convert a general G to bi-regular with low degree [34], however the transformation that gets the degree of the X vertices $\operatorname{poly}(1/\varepsilon)$ makes the size of Σ_X exponential in $\operatorname{poly}(1/\varepsilon)$ (In contrast, the degree of the Y vertices can be made $\operatorname{poly}(1/\varepsilon)$ without changes to the alphabet).

3 Known PCP Theorems

The known PCP constructions are detailed in Table 1 below with citations. To summarize the table, the lowest error known for Label-Cover is $2^{-\Omega(\sqrt{\log N})}$ where N is the size of the input to Label-Cover, but this requires a *quasi-polynomial reduction* from Sat (This follows from the second row of the table with $\varepsilon = 1/n$ and $N = n^{\log n}$). Under the Exponential Time Hypothesis, this only gives a $2^{2^{\Omega(\sqrt{\log N})}}$ time lower bound for Label-Cover. The lowest error known with near-exponential time lower bound $2^{N^{1-o(1)}}$ for Label-Cover is poly-logarithmically small in N (for some poly-logarithm). For more than two queries, researchers achieved lower error with near-exponential time lower bounds. The larger the number of queries is – the lower the error researchers achieved. With $\operatorname{poly}\log\log N$ queries researchers achieved polynomially small error $1/N^{\Omega(1)}$.

In Table 1, *size* is the size of the instance when reducing from Sat on inputs of size n. *proj* means projection. In the projection games constructions in the table the underlying graph can be made an expander (this is implicit in the constructions). For constructions where only a small error probability is mentioned (e.g., $2^{-(\log n)^{\beta}}$), one can obtain constructions for every ε larger than that error by a family of techniques called "composition". Those techniques may increase the number of queries and the size, and hence we only state the result for the smallest error probability (see Subsection 6.2 for more details about composition).

4 Approximation Algorithms

What evidence do we have that polynomially small error for PCP is achievable? The dual question is whether there are approximation algorithms for Constraint Satisfaction Problem or (the easier) Label Cover that approximate to within factors better than polynomial.

Label Cover is *provably* hard up to polynomial factors for semidefinite programming based algorithms, both basic ones and strong ones based on a hierarchy of semidefinite programs [14].

Queries	**Alphabet**	**Error**	**Size**	**Comments**	**Ref**
2	$\exp(\mathrm{poly}(\frac{1}{\varepsilon}))$	ε	$n^{1+o(1)}\,\mathrm{poly}(\frac{1}{\varepsilon})$	proj, deg $\mathrm{poly}(\frac{1}{\varepsilon})$	[34]
2	$\mathrm{poly}(1/\varepsilon)$	ε	$n^{O(\log(1/\varepsilon))}$	proj, deg $\mathrm{poly}(\frac{1}{\varepsilon})$	[4] + [37]
2	$\exp(1/\varepsilon)$	ε	$\mathrm{poly}(n, 1/\varepsilon)$	proj, deg $\mathrm{poly}(\frac{1}{\varepsilon})$	[34] + [21]
3	$\mathrm{poly}(n)$	$\exists\beta > 0,\ 2^{-(\log n)^{\beta}}$	$\mathrm{poly}(n)$		[6, 38]
7	$n^{o(1)}$	$\exists\beta > 0,\ 2^{-(\log n)^{\beta}}$	$n^{1+o(1)}$		[33]
$\mathrm{poly}(1/\beta)$	$O(1/\varepsilon)$	$\varepsilon \geq 2^{-(\log n)^{1-\beta}}$	$\mathrm{poly}(n)$		[17]
$\mathrm{poly}\log\log n$	$n^{o(1)}$	$1/n$	$\mathrm{poly}(n)$		[19]

Table 1: Known PCP constructions.

So, one could say that the Projection Games Conjecture is true for semidefinite programming algorithms. What about other algorithms?

The best known efficient algorithms for Label Cover are combinatorial. The best known worst-case algorithm gives an approximation ratio roughly $1/N^{0.233}$ [14]. When the Label Cover graph is picked at random, but the constraints are worst-case, there is an efficient algorithm that approximates Label Cover to within $1/N^{3-2\sqrt{2}} \approx 1/N^{0.17}$ [14]. This is the best an algorithmic technique called *the log density method* could give, and it might be best possible for Label-Cover (See discussion in [14]).

5 Implications

Next we list some of the implications of the conjectures:

1. Constraint Satisfaction Problem (over large alphabet): The input consists of m tests over n variables, where each test depends on q variables, and the variables assume values from an alphabet Σ. The task is to assign values from Σ to the variables in a way that satisfies as many tests as possible. The Sliding Scale Conjecture implies hardness up to polynomial factors for polynomial sized Σ.

2. 3Lin: The input consists of m linear equations over n Boolean variables, where each equation depends on three variables. The task is to assign the variables so as many equations as possible are satisfied. A random assignment satisfies half of the equations in expectation. The linear Projection Games Conjecture implies hardness of satisfying $1/2 + 1/\mathrm{poly}\, n$ fraction of the equations [31].

3. Directed Multi Cut: The input is an n-vertex directed graph along with source-sink pairs, and the goal is to find the minimum cardinality subset of edges whose removal separates all source-sink pairs. The Sliding Scale Conjecture implies hardness up to polynomial factors [15].

4. Directed Sparsest Cut: The input is an n-vertex directed graph along with source-sink pairs, and the goal is to find a subset of edges to delete so as to minimize the ratio of the number of deleted edges to the number of source-sink pairs that are separated by this deletion. The Sliding Scale Conjecture implies hardness up to polynomial factors [15].

5. Closest Vector Problem: The input is a basis $b_1, \ldots, b_n \in \mathbb{R}^n$ and a point $x \in \mathbb{R}^n$, and the goal is to find a point in the lattice $\{\sum_{i=1}^{n} \alpha_i b_i \mid \alpha_1, \ldots, \alpha_n \in \mathbb{Z}\}$ that is close

to x in ℓ_2 norm. The Projection Games Conjecture implies hardness up to polynomial factors [3, 27].

6. Nearest Codeword Problem: The input is an $n \times k$ matrix A over a finite field $\mathbb{F}$ and a word $w \in \mathbb{F}^n$. The goal is to find the codeword Av for $v \in \mathbb{F}^k$ that is closest in Hamming distance to w. The Projection Games Conjecture implies hardness up to polynomial factors [35].

7. Shortest Vector Problem: The input is a basis $b_1, \ldots, b_n \in \mathbb{R}^n$, and the goal is to find a point in the lattice $\{\sum_{i=1}^n \alpha_i b_i \mid \alpha_1, \ldots, \alpha_n \in \mathbb{Z}\}$ that is as small as possible in ℓ_2 norm. The Projection Games Conjecture implies hardness up to polynomial factors [35].

8. Learning Halfspace Problem: The input is a set of linear inequalities in n variables, and the goal is to find an assignment to the variables that satisfies as many inequalities as possible. The Projection Games Conjecture implies hardness up to polynomial factors [35].

6 Approaches

In this section we survey natural approaches to proving or disproving Sliding Scale Conjectures.

6.1 Error Reduction

One could prove the Sliding Scale Conjecture by decreasing the error of existing PCP verifiers. *Parallel repetition* is a natural method for decreasing the error without increasing the number of queries. Suppose that we start with a PCP in projection form[2]. The proof in the repeated PCP consists of all k-tuples of symbols of the original proof. The repeated verifier picks independently k randomness strings and performs the k tests by querying two k-tuples. There is vast literature ultimately showing that parallel repetition decreases the error from ε to roughly $\varepsilon^{\Omega(k)}$ [23, 37, 25, 36, 21, 30, 13]. However, the size is raised to a power k, and hence one typically uses parallel repetition only for constant k. If one could save in randomness and perform error reduction to polynomially small error while keeping the size polynomial, it would have proved the Projection Games Conjecture. Unfortunately, there are impossibility results for randomness-efficient parallel repetition: when the initial PCP has low degree [23] and for black box analyses of parallel repetition [32]. It is possible that error can be decreased in a randomness-efficient way for specially structured verifiers. An approach along these lines was suggested in [29].

6.2 Alphabet reduction

If we could reduce the alphabet size of existing PCPs (e.g., [34]) without increasing other parameters (size and number of queries) substantially, we could prove the Sliding Scale Conjecture. In error correcting codes, one can decrease alphabet by an idea called *concatenation*: replace each alphabet symbol with an encoding of it over a smaller alphabet. This so-called "inner" encoding can be less efficient than the original, so-called "outer" code, since it is applied on shorter strings. While being much more involved, there is an analogous operations on PCPs, called *composition* [5], which is key to almost all PCP constructions. Unfortunately, existing

[2]Parallel repetition of PCPs in projection form is well understood and useful for hardness of approximation since parallel repetition preserves projection form. Giving similarly strong analyses of parallel repetition of verifiers with more than two queries is an interesting open problem.

composition methods either increase the number of queries [5] or work in the regime of error close to 1 [12, 20] or use alphabet that – while independent of n – is exponentially large in $1/\varepsilon$ [34, 18].

6.3 Sub-sampling

Sub-sampling is the idea of decreasing the randomness by sampling a small subset of the proof symbols and only performing the tests that query them. This idea works for dense PCPs (where verifier tests involve a large fraction of all q-tuples of proof symbols), but dense PCPs have efficient algorithms [2, 9, 1].

6.4 Linear size PCP

One can attempt to prove a weaker version of the Projection Games Conjecture, which shows a mildly sub-exponential reduction from Sat to Label Cover, rather than polynomial time reduction (this in turn rules out polynomial time algorithms for Label Cover under the assumption that Sat requires exponential time). The weaker version would follow from PCP verifiers that achieve constant error using only $r = \log n + O(1)$ random bits, a notorious open problem in PCP [16].

6.5 Locally Decode/Reject Codes

It suffices to construct codes called *locally decode or reject codes* [34] with polynomially small error. Given a list of k-tuples of indices in $\{1, \ldots, n\}$, such codes encode words $w \in \{0,1\}^n$ in a way that enables the decoding of k-tuples of bits from the original word. The decoder is allowed to make $O(1)$ queries to the purported codeword, and should either decode random k bits from the list or reject. The decoder always returns correct k bits for legal codewords. There is probability at most ε (over the choice of k bits and over the randomness of the decoder) that the decoder does not reject yet returns k bits that do not correspond to a list decoding. To prove the Sliding Scale Conjecture, one needs to accommodate $k = \log n$.

6.6 Approximation Algorithms

If one wishes to disprove the Sliding Scale Conjecture, perhaps a good starting point would be to disprove the more ambitious Projection Games Conjecture, or – yet easier – approximate Densest k-Subgraph to within $n^{o(1)}$. In Densest k-Subgraph the input is an undirected graph of size n and the goal is to find a dense sub-graph of size k. The connection between Densest k-Subgraph and Label Cover is as follows: Consider the *label extended graph* of a Label Cover instance, that is, the graph where each vertex $x \in X$ is replaced with vertices (x, a) for $a \in \Sigma_X$ corresponding to the different assignments to x, and each vertex $y \in Y$ is replaced with vertices (y, b) for $b \in \Sigma_Y$ corresponding to different assignments to y. Connect (x, a) and (y, b) if $e = (x, y) \in E$ and $f_e(a) = b$. Set $k = |X \cup Y|$. Label Cover focuses on the density of sets that take at most one vertex $(v, \cdot)$ for every $v \in X \cup Y$.

7 More Open Problems

We mentioned a large number of open problems throughout the column. Several additional ones are:

- Prove more implications of the conjectures, e.g., to Clique and to understanding approximation near the threshold, generalizing the result for 3Lin in Section 5 to other problems.
- Which hardness of approximation results are equivalent to (rather than just implied by) the conjectures? (Among the implications listed in Section 5 we only know equivalence for Constraint Satisfaction Problem with polynomial-sized alphabet).
- Design approximation algorithms for linear projection games.

References

[1] S. Aaronson, R. Impagliazzo, and D. Moshkovitz. AM with multiple merlins. In *Computational Complexity Conference*, 2014.

[2] N. Alon, W. F. de la Vega, R. Kannan, and M. Karpinski. Random sampling and approximation of MAX-CSPs. *J. Comput. Sys. Sci.*, 2(67):212–243, 2003.

[3] S. Arora, L. Babai, J. Stern, and Z. Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. *Journal of Computer and System Sciences*, 54(2):317–331, 1997.

[4] 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.

[5] S. Arora and S. Safra. Probabilistic checking of proofs: a new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998.

[6] S. Arora and M. Sudan. Improved low-degree testing and its applications. *Combinatorica*, 23(3):365–426, 2003.

[7] L. Babai, L. Fortnow, L. A. Levin, and M. Szegedy. Checking computations in polylogarithmic time. In *Proc. 23rd ACM Symp. on Theory of Computing*, pages 21–32, 1991.

[8] L. Babai, L. Fortnow, and C. Lund. Nondeterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991.

[9] B. Barak, M. Hardt, T. Holenstein, and D. Steurer. Subsampling mathematical programs and average-case complexity. In *In Proc. ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 512–531, 2011.

[10] M. Bellare, O. Goldreich, and M. Sudan. Free bits, PCPs, and nonapproximability—towards tight results. *SIAM Journal on Computing*, 27(3):804–915, 1998.

[11] M. Bellare, S. Goldwasser, C. Lund, and A. Russell. Efficient probabilistically checkable proofs and applications to approximations. In *Proc. 25th ACM Symp. on Theory of Computing*, pages 294–304, 1993.

[12] E. Ben-Sasson, O. Goldreich, P. Harsha, M. Sudan, and S. Vadhan. Robust PCPs of proximity, shorter PCPs, and applications to coding. *SIAM Journal on Computing*, 36(4):889–974, 2006.

[13] M. Braverman and A. Garg. Small value parallel repetition for general games. In *Proc. 47th ACM Symp. on Theory of Computing*, 2015.

[14] E. Chlamtac, P. Manurangsi, D. Moshkovitz, and A. Vijayaraghavan. Approximation algorithms for label cover and the log-density threshold. In *Proc. of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2017.

[15] J. Chuzhoy and S. Khanna. Polynomial flow-cut gaps and hardness of directed cut problems. *Journal of the ACM*, 56(2), 2009.

[16] I. Dinur. Mildly exponential reduction from gap-3SAT to polynomial-gap label-cover. Technical report, ECCC TR16-128, 2016.

[17] I. Dinur, E. Fischer, G. Kindler, R. Raz, and S. Safra. PCP characterizations of NP: Toward a polynomially-small error-probability. *Computational Complexity*, 20(3):413–504, 2011.

[18] I. Dinur and P. Harsha. Composition of low-error 2-query PCPs using decodable PCPs. In *Proc. 50th IEEE Symp. on Foundations of Computer Science*, pages 472–481, 2009.

[19] I. Dinur, P. Harsha, and G. Kindler. Polynomially low error PCPs with polyloglog n queries via modular composition. In *Proc. 47th ACM Symp. on Theory of Computing*, pages 267–276, 2015.

[20] I. Dinur and O. Reingold. Assignment testers: Towards a combinatorial proof of the PCP theorem. *SIAM Journal on Computing*, 36(4):975–1024, 2006.

[21] I. Dinur and D. Steurer. Analytical approach to parallel repetition. In *Proc. 46th ACM Symp. on Theory of Computing*, 2014.

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

[23] U. Feige and J. Kilian. Impossibility results for recycling random bits in two-prover proof systems. In *Proc. 27th ACM Symp. on Theory of Computing*, pages 457–468, 1995.

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

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

[26] R. Impagliazzo and M. Paturi. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512 – 530, 2001.

[27] S. Khot. Inapproximability results for computational problems on lattices. In *The LLL Algorithm*, pages 453–473. 2010.

[28] S. Khot. On the unique games conjecture (invited survey). In *IEEE Conference on Computational Complexity*, pages 99–121, 2010.

[29] D. Moshkovitz. An approach to the sliding scale conjecture via parallel repetition for low degree testing. Technical Report 30, ECCC, 2014.

[30] D. Moshkovitz. Parallel repetition from fortification. In *Proc. 55th IEEE Symp. on Foundations of Computer Science*, 2014.

[31] D. Moshkovitz. The projection games conjecture and the NP-hardness of $\ln n$-approximating set-cover. *Theory of Computing*, 11(7):221–235, 2015.

[32] D. Moshkovitz, G. Ramnarayan, and H. Yuen. A no-go theorem for derandomized parallel repetition: Beyond feige-kilian. In *RANDOM*, 2016.

[33] D. Moshkovitz and R. Raz. Sub-constant error probabilistically checkable proof of almost-linear size. *Computational Complexity*, 19(3):367–422, 2010.

[34] D. Moshkovitz and R. Raz. Two query PCP with sub-constant error. *Journal of the ACM*, 57(5), 2010.

[35] P. Mukhopadhyay. The projection games conjecture and the hardness of approximation of SSAT and related problems. Technical report, arXiv:1907.05548, 2019.

[36] A. Rao. Parallel repetition in projection games and a concentration bound. *SIAM Journal on Computing*, 40(6):1871–1891, 2011.

[37] R. Raz. A parallel repetition theorem. In *SIAM Journal on Computing*, volume 27, pages 763–803, 1998.

[38] R. Raz and S. Safra. A sub-constant error-probability low-degree test and a sub-constant error-probability PCP characterization of NP. In *Proc. 29th ACM Symp. on Theory of Computing*, pages 475–484, 1997.