Notes on PCP based inapproximability results

Notes de cours motivées par les 15ème JCALM
contact: marin.bougeret@lirmm.fr

Abstract The objective of this document is to present the PCP based proofs of the optimal inapproximability results of several important problems (vertex cover, independent set, max cut, max 3 SAT, ..). The angle we adopt here is to blackbox the very technical proofs of the PCP theorems (like $NP \subseteq PCP(...,...)$), but to make as explicit as possible all the "combinatorial" remaining part. In Section 1 we recall some definitions, Section 2 is devoted to vertex cover and independent set, and Section 3 is devoted to constraint satisfaction problems. This document is still a draft, any comment is welcome!

1 Definitions

1.1 General Definitions

Definition 1 (Promise problem [14]). *A promise problem L is a pair of non-intersecting sets denoted $L = (L_{YES}, L_{NO})$; that is, $L_{YES}, L_{NO} \subseteq \{0,1\}^*$, and $L_{YES} \cap L_{NO} = \emptyset$.*

As explained in [14], we need promise problem for gap problems, and for any other situation where we need that the input verifies a certain given property (which could be hard to check), in particular in Theorem 4. We say that an algorithm A decides L is given $x \in L_{YES} \cup L_{NO}$ (*i.e.* the algorithm is allowed to **suppose** that x is either in L_{YES} or L_{NO}, even if it could be hard to decide it), for any $x \in L_{YES}$ (resp. L_{NO}), $A(x) = 1$ (resp. 0) Then, all the definitions of complexity classes can be adapted in a straightforward manner[1]. For example, P is the class of promise problem $L = (L_{YES}, L_{NO})$ such that there exists a polynomial algorithm A deciding L.

Definition 2 (gap problem [1]). *Let Π be an optimization problem, $\mathcal{I}$ the set of instances of Π, and a, b two functions from $\mathcal{I}$ to $\mathbb{Q}$ with $a > b$. We denote by $gap_{a,b}\Pi$ the promise problem where $L_{YES} = \{I \in \mathcal{I} | OPT(I) \leq b(I)\}$ and $L_{NO} = \{I \in \mathcal{I} | OPT(I) \geq a(I)\}$.*

The motivation behind gap problem is that if $gap_{a,b}\Pi$ is NP-hard, then for any $\epsilon > 0$ there is no $\frac{a}{b} - \epsilon$ polynomial approximation for Π unless P=NP.
Karp reductions between promise problems are also defined in a straightforward way: instances of L_{YES} (resp. L_{NO}) are mapped to instances of L'_{YES} (resp. L'_{NO}). We also say that f is a randomized karp reduction ([5], chapter 7) between L and L' (that are promise problems) (and we denote $L \leq_R L'$) iff for any x, $P(\mathbb{1}^{L'}_{f(x)} = \mathbb{1}^{L}_{x}) \geq \frac{2}{3}$, where $\mathbb{1}^{L}_{x} = 1$ if $x \in L_{YES}$ and 0 if $x \in L_{NO}$. We extend this notation to $\mathcal{L} \subseteq_R \mathcal{L}'$ for two sets of languages $\mathcal{L}$ and $\mathcal{L}'$. Notice that $L \leq_R L'$, f polynomial (meaning polynomial whatever the random choices) and $L' \in$BPP implies $L \in$BPP.

[1] However, some "structural" relations between classes no longer hold for promise problems, as explained for example in [14,21]

1.2 PCP Definitions

The following definition was introduced in [7] and is stated as in [6].

Definition 3 ($PCP(r(n), q(n))$ Verifier). *A $PCP(r(n), q(n))$ verifier V for a langage L is a randomized deterministic turing machine which for any input x of size n, and any proof y of size polynomial in n:*

- *generates a random string τ of $r(n)$ bits (I suppose that each bit is set independently to 0 with probability $\frac{1}{2}$, see Remark 2)*
- *computes in polynomial time (in n) a sequence $S(x, \tau) = (p_1(x, \tau), \ldots, p_k(x, \tau))$ (denoted sometimes S for short) of $k \leq q(n)$ positions (without reading y, we say that the verifier is non adaptative)*
- *reads bits $y_S = (y_{p_1(x,\tau)}, \ldots, y_{p_k(x,\tau)})$ of the proof y (for example if $y = (010010)$, $S = (4, 0, 1)$, $y_S = (1, 0, 1)$)*
- *computes in polynomial time in n a boolean $f_{(V,x,\tau)}(y_S)$, where $f_{(V,x,\tau)}$ is a function from $\{0, 1\}^k$ to $\mathbb{B}$.*

Definition 4 ($\mathcal{F}_V$ and $V(x, y, \tau)$). *Given a $PCP(r(n), q(n))$ verifier V and an input x of size n, we define $\mathcal{F}_{(V,x)} = \{f_{(V,x,\tau)}, \tau \in \{0, 1\}^{r(n)}\}$ the set of functions used by the verifier on x. All these functions have arity at most $q(n)$. Let $\mathcal{F}_V = \cup_x \mathcal{F}_{(V,x)}$ be the set of functions used by the verifier.*
Notice that given a fixed input x, proof y and random string τ, the verifier V always gives the same answer. Thus, we define $V(x, y, \tau) = f_{(V,x,\tau)}(y_{S(x,\tau)})$.

Remark 1. In [6] authors define S as a set, whereas in [7] S is defined as a sequence. We chose here to define S as a sequence as in [6], the elements of S are in fact "explicitly named" (*i.e.* [6] says that "the verifier computes a set $S = \{p_1(x, \tau), \ldots, p_k(x, \tau)\}$, and then reads the bits $y_S = (y_{p_1(x,\tau)}, \ldots, y_{p_k(x,\tau)})$", and thus I think that S behaves like a sequence and not a set.

I chose to introduce the following definition as I need it later (Proposition 3 provides reduction from a commutative verifier to commutative restrictions of CSP problems, and we need these commutative restrictions of CSP problems to have the correspondences in Remark 11).

Definition 5. *A PCP verifier is commutative iff for any $f \in \mathcal{F}_V$, f is commutative.*

For example, as explained in [23], the famous 3 bits PCP verifier for Max 3SAT of [17] "uses it randomness to pick three entries i, j, k in the witness and a bit b, and it accepts iff $i + j + k = b$". Using our formalism, this verifier V has $\mathcal{F}_V = \{f_0, f_1\}$ where $f_i(x, y, z)$ returns $x + y + z = i$. Thus, the random string τ will determine the function to use, and the bits of the proofs to read. Notice that in this case the $f_{(V,x,\tau)}$ do not depend on x, and the verifier is commutative.

Remark 2. I didn't find a definition where the "uniformity" of the random string is clearly mentioned (maybe as it may be implicit in the definition of "random coin" in any probabilistic Turing machine) but it seems that this property is required. Indeed we will use the following argument (which becomes clearly false if τ is not chosen uniformly) several times. Suppose a verifier has soundness s and uses r random bits. Then, for any x and proof y, there is at most $s2^r$ random strings that make the verifier accept (*i.e.* $|\{\tau | V(x, y, \tau) = T\}| \leq s2^r$).

Definition 6 ($PCP_{(c,s)}(r(n),q(n))$ Verifier for a langage L). *For any $c,s \in [0,1]$, a $PCP_{(c,s)}(r(n),q(n))$ verifier for a langage L is a $PCP(r(n),q(n))$ verifier V such that there exists a polynomial p such that*

- *for any $x \in L$ (with $|x| = n$), there exists y, $|y| \leq p(n)$ such that $\Pr_\tau[V(x,y,\tau) = T] \geq c$ (c is called the completeness)*
- *for any $x \notin L$ (with $|x| = n$), for y $|y| \leq p(n)$, $\Pr_\tau[V(x,y,\tau) = T] \leq s$ (s is called the soundness)*

Remark 3. If we are given a $PCP_{(c,s)}(r(n),q(n))$ verifier V, we can assume that we explicitly know c, s, r, q. [2]

Definition 7 ($PCP_{(c,s)}(r(n),q(n))$). *We denote by $PCP_{(c,s)}(r(n),q(n))$ the set of languages having a $PCP_{(c,s)}(r(n),q(n))$ verifier.*

Definition 8 (Free bits, Amortized free bits, [10]). *The free bit complexity of V is f iff for any input x, and random string τ, $|\{y|V(x,y,\tau) = T\}| \leq 2^f$. The amortized free bit complexity $\bar{f} = \frac{f}{log(g)}$, where $g = \frac{c}{s}$ is the gap of the verifier. We define also*

- *$FPCP_{(c,s)}(r(n),q(n),f)$ verifier for a langage L as a $PCP_{(c,s)}(r(n),q(n))$ verifier having a free bit complexity of f*
- *$FPCP_{(c,s)}(r(n),f)$ as previously but where $q(n)$ is assumed to be constant*
- *$F\bar{P}CP_{(c,s)}(r(n),q(n),\bar{f})$ verifier for a langage L as a $PCP_{(c,s)}(r(n),q(n))$ verifier having an amortized free bit complexity of $\bar{f}$*
- *$F\bar{P}CP_{(c,s)}(r(n),\bar{f})$ as previously but where $q(n)$ is assumed to be constant*

Remark 4. The free bit complexity simply upper bound the number of accepted proofs. Obviously we have $f < q(n)$. The intuition of the term "free" can be explained as follows (from [16]). Suppose V reads three bits y_1, y_2, y_3, and accepts iff $y_3 = f(y_1, y_2)$ for some function f (+ for example). Thus, sometimes V has no idea of what the value of the bit should be (when reading y_1 and y_2), and sometimes V is in "checking mode" and knows what to expect (when reading y_3). In this situation, there are only 2 free bits (y_1 and y_2), and for any f there is here at most 4 accepted proofs.

Remark 5. Notice that in $F\bar{P}CP$, the soundness and the number of free bits are not specified. Thus, if we are given a $F\bar{P}CP_1(r,\bar{f})$ verifier V, we only know that there exists s such that V is a $FPCP_{1,s}(r,\bar{f}log(s^{-1}))$ verifier.

1.3 UGC Definitions

We follow definitions and remarks from [20]. We only recall here basic definitions related to UGC to make the document self contained.

Definition 9 (Input of the (bipartite weighted) Label Cover Problem). *Input: a tuple $\Phi = (X,Y,R,\Psi,W)$. X (resp. Y) is the set of "left" (reps. "right") vertices, and R is a set of integers called labels. For each $x \in X$, $y \in Y$, Ψ contains one relation $\psi_{xy} \subseteq R \times R$ and W contains its weight $w_{xy} \geq 0$. A labeling is a function L mapping $X \cup Y$ to R. A constraint ψ_{xy} is said to be satisfied by a labeling L iff $(L(x),L(y)) \in \psi_{xy}$. We denote by $w_L(\Phi)$ the sum of the weights of the constraints satisfied by L, and $w(\Phi) = sum_{x\in X, y\in Y} w_{xy}$.*

[2] This hypothesis is required in many proofs, but I didn't find yet a formal definition where it is explicitly mentioned (maybe trivial?).

Theorem 1 ([6,7,22]). *For any $\gamma > 0$, there exists an integer r such that the following is NP-hard. Given a bipartite weighted label cover instance Φ with a label set R, $|R| = r$, and $w(\Phi) = 1$, distinguish between the two following cases:*
- *(YES case) There exists a labeling L such that $w_L(\Phi) = 1$*
- *(NO case) For any labeling L, $w_L(\Phi) \leq \gamma$.*

Definition 10 (Input of the (bipartite weighted) Unique Label Cover problem). *An input of the (bipartite weighted) Unique Label Cover problem is an input Φ of the bipartite weighted label cover problem where for any x, y, ψ_{xy} is a bijection from R to R.*

We can now formulate the Unique Game Conjecture of [18]

Conjecture 1. For any $\zeta, \gamma > 0$, there exists an integer r such that the following is NP-hard. Given a bipartite weighted unique label cover instance Φ with a label set R, $|R| = r$, and $w(\Phi) = 1$, distinguish between the two following cases:
- (YES case) There exists a labeling L such that $w_L(\Phi) = 1 - \zeta$
- (NO case) For any labeling L, $w_L(\Phi) \leq \gamma$.

Before turning to the inapproximability results, notice that the global idea when using PCP (combined or not with UGC) is always the same:
- the objective is to create a gap reduction from a (typically $NP - hard$) decision problem Π to an optimization problem Π'
- to that end, given an instance I of Π and a PCP verifier V of Π, we create an instance $I' = f(I, V)$ of Π'

Thus, "using UGC" in these PCP based reductions simply means that we only know that Π is $NP - hard$ when assuming UGC.

Let us now turn to inapproximability results. Results marked with a $\star$ are optimal in the sense where the inapproximability result matches the ratio of the best known algorithm. Results marked with $\star^{ar}$ (for "approximation resistant", see for example in [2]) correspond to those problems where upper and lower bound match AND the corresponding algorithm is simply the (derandomized) random assignment. For example for Max-E3-SAT, if on assigns independently each variable to "true" with probability $\frac{1}{2}$, each clause is satisfied with probability $\frac{7}{8}$, and by linearity of expectation, the expectation of the solution is $\frac{7}{8}m$ (where m is the number of clauses). As this kind of algorithm can be derandomized, this leads to a $\frac{7}{8}$ approximation.

2 Independent Set and Vertex Cover via the FGLSS reduction

2.1 FGLSS reduction

Let V be a $FPCP_{(c,s)}(r(n), q(n), f)$ verifier for a langage L. Let Π_L be the problem of deciding if an input $x \in L$. Let x be an input Π_L. We construct an instance $G^{IS}(V, x) = (V^{IS}, E^{IS})$ of the MIS problem as follows. For each τ with $|\tau| = r(n)$ and each y with $|y| \leq q(n)$, we run $f_{(V,x,\tau)}(y)$ and create one vertex denoted (x, τ, y) iff $f_{(V,x,\tau)}(y)$ returns true. Notice that by definition of f, for each τ we have at most 2^f such y, and thus the total number of vertices $|V^{IS}|$ is at most $2^{r(n)+f}$ (but the running time to create the graph is still in $\mathcal{O}^*(2^{r(n)+q(n)})$). Given (x, τ, y) and (x, τ', y'), let $S(x, \tau) = (i_1, \ldots, i_k)$ and $S'(x, \tau') = (i'_1, \ldots, i'_{k'})$. We create an edge between (x, τ, y) and (x, τ', y') iff there exists $i = i_j = i'_{j'}$ (with $i_j \in S(x, \tau)$ and $i'_{j'} \in S(x, \tau')$) such that $y_j \neq y'_{j'}$ (see Figure 1). Such an edge means that these two vertices are incoherent as bit i of the underlying proof y is not the same in the two states of the vertifier. Notice that G^{IS} is a "layered graph": for any τ, the set $\{(x, \tau, y),\ y \text{ with } |y| \leq q(n)\}$ is a clique. This ends the description of G^{IS}.

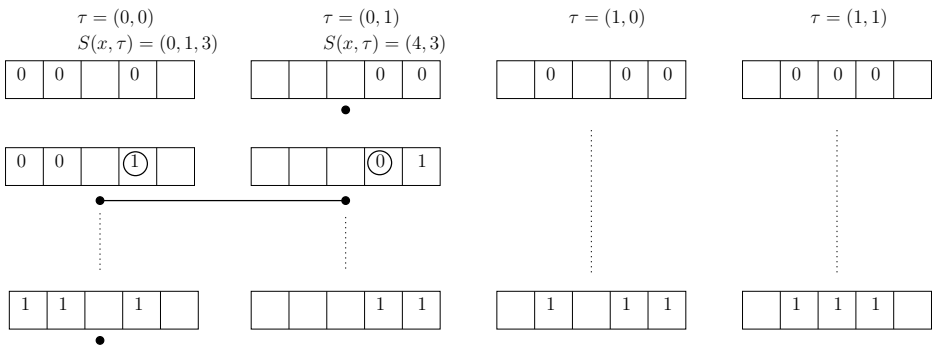

Figure1: Example of the FGLSS reduction with $r = 2$, $q = 3$, $f = 2$. Notice that we only add vertices (represented by black dots) for the (x, τ, y) which are accepted. An edge is depicted between $(x, (0,0), (0,0,1))$ and $(x, (0,1), (0,1))$ as bit 3 (circled) of the proof is different in the two vertices.

2.2 Independent Set

Proposition 1 (gap for IS in the FGLSS reduction). *We can prove (see for example [3], lecture 7) that*

- *for any proof y which is accepted with probability c, there exists an independent set I of G^{IS} of size $|I| = c2^{r(n)}$ (for each $\tau \in R$ we pick the vertex $(x, \tau, y_{S(x,\tau)})$) (recall that for any S, y_S dnotes the restriction of y to bits in S)*
- *conversely, for any independent set I, there is a proof y which is accepted with probability at least $\frac{|I|}{2^{r(n)}}$*

Thus, if $x \in L$, $Opt \geq c2^{r(n)}$, and if $x \notin L$, $Opt \leq s2^{r(n)}$. To conclude we get a gap of $\frac{c}{s}$ in a graph with $2^{r(n)+f}$ vertices.

Theorem 2. *There is no constant approximation for MIS unless P=NP.*

Proof. The original PCP theorem says that $NP = PCP_{1,\frac{1}{2}}(O(log(n)), O(1))$. Using the FGLSS reduction, this provides a reduction from an NP hard problem to the MIS problem running in polynomial time, with a graph of size polynomial in n, and a gap of 2. We can boost this gap by repeating[3] k times the verifier for any constant k. This implies that $NP = PCP_{1,\frac{1}{2^k}}(O(klog(n)), O(k))$. The gap becomes 2^k, and the reduction remains polynomial for fixed k (the graph has $n^{O(k)}$ vertices).

□

Our objective is now to prove the optimal inapproximability of Theorem 5. The next Lemma show that if we can boost the soundness of the verifier to $\frac{1}{2^k}$ using significantly less than rk random bits (but kf free bits is ok), we get an optimal inapproximability result.

Proposition 2 ([10], p11/12). *If for any k and $\bar{f} > 0$ we have a $FPCP_{1,2^{-k}}(r + k, (r + k + 2)\bar{f})$ verifier V (with $r = O(log(n))$), then the FLGSS reduction remains polynomial and provides a gap of $n^{1-\epsilon}$ for any $\epsilon > 0$. Notice that the amortized free bit complexity of such verfier is $\bar{f}(1 + \frac{r+2}{k})$, and thus this hypothesis implies that we can produce verifier with arbitrarily small amortized free bit complexity.*

[3] another simple way to boost the gap using graph vocabulary is to reduce $gap_{c,s}IS$ to $gap_{c^2,s^2}IS$ by taking the graph product

Proof. The gap in the reduction is 2^k, and the number of vertices is $N = 2^{2\bar{f}+(r+k)(1+\bar{f})} = 2^{k(\frac{r(n)(1+\bar{f})+2\bar{f}}{k}+1+\bar{f})}$. Thus, the gap can be written $N^{\frac{1}{\frac{r(n)(1+\bar{f})+2\bar{f}}{k}+1+\bar{f}}}$. As we can have a verifier for any k and $\bar{f}$, and the only constraint is that N remains polynomial in n, we can take $k = \frac{r(n)}{\epsilon'}$ and $\bar{f} = \epsilon''$. Notice that we couldn't take a too big k (like $k = poly(n)$), as the graph would not have a polynomial number of vertices.

□

Our goal is now to achieve a $n^{1-\epsilon}$ gap, but only under NP != BPP (the derandomization has only been proved later in [25]). The good news is that (according to free bits, pcp , p11) [16] provided verifier with arbitrary small amortized free bit complexity. The following theorem is usually referred as the article proving that independent set is hard to approximate within $n^{1-\epsilon}$.

Theorem 3. *[16] For any $\bar{f}$, $NP \subseteq F\bar{P}CP_1(O(log(n)), \bar{f})$.*

Remember that as stated in Remark 5, when having a verifier in $F\bar{P}CP_1(r, \bar{f})$, we only know that there exists s such that V is a $FPCP_{1,s}(r, \bar{f}log(s^{-1}))$ verifier, and thus the soundness and free bits do not necessarily respect the hypothesis we need in Proposition 2. Thus, it remains now to show how from a $F\bar{P}CP_1(log(n), \bar{f})$ verifier we can get an $FPCP$ verifier as required in Proposition 2. This is where we loose determinism (implying only the NP != BPP), as the following reduction is randomized.

Theorem 4 ([10], corollary 11.3). *For any $\bar{f} > 0$ and integer k, $F\bar{P}CP_1(O(log(n)), \bar{f}) \subseteq_R FPCP_{1,2^{-k}}(r+k, (r+k+2)\bar{f})$ (with $r = O(log(n))$). This result can be seen as, given any $F\bar{P}CP_1(r, \bar{f})$ verifier V (for a given $\bar{f}$), we can chose (according to k) how we boost the gap of V.*

Proof. This proof follows [12] (lemma 5.4). Before proving this theorem, we need the following lemmas and definitions.

Lemma 1. *$\forall \bar{f}, s$ and integer r, R, $FPCP_{1,s}(r, f) \subseteq_R FPCP_{1,2^{r-R}}(R, df)$, where $d = \frac{R+2}{log(s^{-1})}$ (here we can chose R to adjust the gap)*

Proof. The idea to allow k repetitions without having kr random bits is explained in [12] as follows: "Zuckerman [23] used a probabilistic construction to in some sense recycle randomness. His construction uses R random bits to simulate a number of runs each using r random bits. In his paper, the parameters s and f were constants and r logarithmic, but we make no such assumptions in this section. Instead we establish that Zuckerman's construction works even in the general case, and we prove a general theorem where the parameters involved appear. The idea is to pick a random bipartite graph with 2^R vertices on the left side, 2^r vertices on the right side, and where the degree of the vertices on the left side is d. From this graph construct a new probabilistic machine, which uses the R random bits to select a vertex v on the left side, and then runs the verifier in the proof system d times, letting the verfier's random bits be determined by the vertices adjacent to v. If we obtain only accepting runs, we accept our input, otherwise we reject. Obviously, it may happen that we incorrectly accept an input which is not in the language. Since our original proof system for NP has soundness s, an s fraction of all r-bit random strings may cause the verifier to accept. Thus, we must bound the probability that a large subset of the vertices on the left side has only such neighbors."

Definition 11 (as stated in [12]). *A bipartite graph $H = (E, V_1 \cup V_2)$ is a (d, n_1, n_2)-disperser if all vertices in V_1 have degree d, all vertices in V_2 have expected degree $\frac{d|V_1|}{|V_2|}$, and no sets $S_1 \subseteq V_1$ and $S_2 \subseteq V_2$ of cardinality n_1 and n_2, respectively, have the property that every vertex in S_1 is connected only with vertices in S_2.*

Remark 6. It seems that the property "expected degree" is not used here, and moreover this property is not mentioned in other definitions of disperser.

Lemma 2 (as stated in [12]). *Let $H = A(R, r, s)$ be a random bipartite graph $H = (E, V_1^H \cup V_2^H)$ where $|V_1^H| = 2^R$, $|V_2^H| = 2^r < 2^R$, and the edge set $E \subseteq V_1^H \times V_2^H$ is chosen as follows: for each $v \in V_1^H$, pick d neighbors independently and uniformly among the vertices in V_2^H. We allow multiple edges in H. If $d = (R+2)/logs^{-1}$ for some $s < 1$, the graph H is a $(d, 2^r, s2^r)$-disperser with probability at least $1 - 2^{-2^r}$.*

Definition 12. *Let $L^1 \in FPCP_{1,s}(r, f)$ (with V^1 the corresponding verifier). Let $R > r$, and suppose H is a $(d, 2^r, s2^r)$ disperser of Lemma 2. Let us define a verifier V^2_{H,V^1} as follows. Given an input x, $V^2_{H,V^1}(x)$ behaves as follows. It first picks uniformly a random $v \in V_1^H$ (using R random bits). Let $u_1, \ldots, u_d$ be the neighbors of v in V_2^H. Given a proof y, run $f_{(V^1,x,u_i)}(y_{S_i})$ for any $i \in [d]$ (where $S_i = S(x, u_i)$ is the sequence of positions computed when using random string u_i in V_1), and accept iff the d runs accept.*

Lemma 3. *Let $R > r$, and suppose H is a $(d, 2^r, s2^r)$ disperser, and that V_1 is a $FPCP_{1,s}(r, f)$ verifier. Then V^2_{H,V^1} is a $FPCP_{1,2^{r-R}}(R, df)$ verifier.*

Proof (of Lemma 3). If $x \in L^1$, then there exists a proof y such that all $f_{(V^1,x,\tau)}(y_{S_{(x,\tau)}})$ accept, and thus V^2_{H,V^1} accepts. Let us suppose now that $x \notin L^1$. Let y be a proof. Let $S_2 \subseteq V_2$ (resp. $S_1 \subseteq V_1$) be the set of random strings that make V^1 (resp. V^2) accept. Since V^1 has soundness s we have $|S_2| \leq s2^r$, and by definition of a disperser we have $|S_1| < 2^r$. Thus, the soundness of V^2_{H,V^1} is 2^{r-R} as there is at most 2^r random strings that make it accept among the 2^R possible, the number of random bit used in R, and the number of free bits is at most df.

□

Remark 7. Notice that V^2_{H,V^1} follows the definition of a non adaptative PCP verifer: we can first compute for all i the sequences $\{y_{S_i}\}$ of positions to read in the proof, and then run the $f_{(V^1,x,u_i)}(y_{S_i})$.

It remains now to prove Lemma 1 ($FPCP_{1,s}(r, f) \subseteq_R FPCP_{1,2^{r-R}}(R, df)$). In Lemma 3 we supposed that H was a disperser, but we are only able to create a disperser with good probability. Notice that we cannot add the "disperser creation" to the role of the verifier, as otherwise the amounts of random bits used to create the disperser would be too large (and we only want to use R random bits).
To clarify the distinction between the two sources of randomness (one to create the disperser, and the other in the pcp verifier), let us rewrite lemma 3 in terms of randomized karp reduction between L^1 and a langage L^2, where L^2 is a promise problem.[4] L_2 is simply L^1 with an additional input which is supposed to be a disperser, and thus we define $L^2_{YES} = \{(x, H)| \in L^1_{YES}$ and H is a $(d, 2^r, s2^r)$ disperser$\}$ and $L^2_{NO} = \{(x, H)| \in$

[4] I have never seen the result explicitly written as a randomized karp reduction, but it seems reasonable that we need a promise problem, as [14] confirms p16 that "the randomness in a PCP can be reduced [..], but the actual result is a randomized karp reduction of any problem having a PCP to a *promise* problem having a PCP with [..].

L^1_{NO} and H is a $(d, 2^r, s2^r)$ disperser$\}$. Then, we see that $f(x) = (x, A(R, r, s))$ is a randomized karp reduction as we have $P(\mathbb{1}^{L^2}_{f(x)} = \mathbb{1}^{L^1}_x) \geq 1 - 2^{-2^r} \geq \frac{2}{3}$. Moreover, the previous verifier V^2_{H,V^1} is a verifier for L^2, as given an instance (x, H), we can suppose by definition of the promise problem that H is a disperser. *end of proof of Lemma 1*

□

We can now prove theorem 4. Let L in $F\bar{P}CP_1(O(log(n)), \bar{f})$. By definition, there exists s such $L \in FPCP_{1,s}(O(log(n)), \bar{f}log(s^{-1}))$. Applying the previous result, we get a $FPCP_{1,2^{r-R}}(R, (R+2)\bar{f})$ for any R. It suffices now to chose $R = r + k$.

□

Remark 8. The parameters of the verified obtained in Theorem 4 do not exactly match the result stated in [10] in corollary 11.3, where they prove that for any $\bar{f}$, for every $\epsilon > 0$, there exists a constant c such that $F\bar{P}CP_1(O(log(n), \bar{f}) \leq_R FPCP_{1,2^{-t}}((1+\epsilon)t, \bar{f}t)$ (where $t(n) = clog(n)$). Indeed, if we replace k by $\frac{r(n)}{\epsilon} = \frac{O(log(n))}{\epsilon}$ in Theorem 4, we get $F\bar{P}CP_1(O(log(n), \bar{f}) \leq_R FPCP_{1,2^{-k}}((1+\epsilon)k, \bar{f}(k+\epsilon+2)$. However, this leads to the same innaproximability result for independent set, and the motivation was to prove Theorem 4 following [12] (lemma 5.4), where the role of the disperser is clearly mentioned.

Using Theorem 4 and 3, we get the following result (the derandomization leading to the conditional result under $NP \neq P$ has only been proved in [25]).

Theorem 5 ($\star$). *For any $\epsilon > 0$, there is no $n^{1-\epsilon}$ approximation algorithm for MIS unless $NP \neq BPP$.*

2.3 Vertex Cover

Let us now consider the gap created for VC in the FLGSS reduction. According to Proposition 1, if $x \in L$, then $OPT_{VC} \leq 2^{r(n)+f} - c2^{r(n)}$, and otherwise $OPT_{VC} \geq 2^{r(n)+f} - s2^{r(n)}$. Thus, we get a gap of $\frac{2^f - s}{2^f - c}$, and thus we see the importance of the parameters of the verifier on the approximability of VC.
Thus, for a long time the best inapproximability result (under $P \neq NP$) was the following.

Theorem 6. *For any $\epsilon > 0$ there is no $\frac{7}{6} - \epsilon$ polynomial approximation algorithm for VC unless $P = NP$.*

Proof. In [17] (Theorem 8.1), authors provide a PCP verifier for $Max - E3 - lin2$ (an NP-hard problem) with 2 free bits, completeness $1 - \epsilon$, and soundness $\frac{1}{2} + \epsilon$, implying the desired result.

□

The previous result was improved to 1.36 (as explained in [20]), but this is only recently that, thanks to UGC, this result has been improved.

Theorem 7 ([20] $\star$). *Under UGC, for any $\epsilon > 0$ there is no $2 - \epsilon$ polynomial approximation algorithm for VC.*

Proof. As explained in [9], assuming UGC, authors of [20] provide for any ϵ, δ a $FPCP_{\frac{1}{2}-\epsilon,\delta}(log, 0)$ verifier for an NP hard langage[5], implying the desired result. Another way to prove

[5] I think that "assuming UGC" means that they provide a $FPCP_{\frac{1}{2}-\epsilon,\delta}(log, 0)$ verifier for the unique label cover problem, which is only known to be NP hard when assuming UGC

(where the role of UGC is more clear) is the result of [9]. In this paper (thm 1.3), authors provide for any ϵ, δ a $FPCP_{1-\epsilon,\delta}(log, 1)$ verifier for the unique game problem (whose NP-hardness is assumed by the definition of UGC!), leading also to a $2-\epsilon$ inapproximability result.

□

3 CSP problems

[17] provides inapproximatiliby for several problems of the flavor of max-3-sat, motivating thus the definition of the general wMax-CSP-$\mathcal{F}$ problem, inspired from [5], definition 18.11, or [2].

Definition 13. *Let k be an integer and $\mathcal{F}$ (also called a "constraint language") be a set of function $F : \mathbb{B}^k \to \mathbb{B}$, with $\mathcal{F} = \{f_1, \dots, f_{|\mathcal{F}|}\}$, such that for any i, j, $f_i \neq f_j$. The problem wMax-CSP-$\mathcal{F}$ of arity k is defined as follows:*

- *input: a set of variables $X = \{x_j, j \in [n]\}$, a set of constraints $\{\mathcal{C}_i, i \in [m]\}$, where $\mathcal{C}_i = (C_i, u_i, w_i)$, were $C_i = (l^i_1, \dots, l^i_k)$ is a k-tuples of literals, (l^i_j is of the form x_t or $\bar{x}_t$), $u_i \in [|\mathcal{F}|]$ is the index of a function in $\mathcal{F}$, and $w_i > 0$.*
- *output: a boolean assignment a of variables of X (and we denote by $a(C_i) \in \mathbb{B}^k$ k-tuples when substituting according to a)*
- *objective function: maximizes $\frac{1}{\sum_i w_i} \sum_{i \in [m] | f_{u_i}(a(C_i))=T} w_i$ the weighted fraction of constraints that are satisfied by a*

Moreover, we suppose that no pair of constraints are identical, where $\mathcal{C}_i = \mathcal{C}_j$ iff $u_i = u_j$ and $C_i = C_j$ (and $C_i = C_j$ iff for any t, $l^i_t = l^j_t$, where two litterals are equal iff they correspond to the same variable in the same form).

Definition 14. *We define the problem Max-CSP-$\mathcal{F}$ as wMax-CSP-$\mathcal{F}$ when $w = 1$.*

Definition 15. *Following the notation of [8], we denote by wMax-CSP$^+$-$\mathcal{F}$ the restriction of wMax-CSF-$\mathcal{F}$ where all C_i are k-tuples of variables (i.e. the l^i_j is of the form x_t only).*

I chose to introduce the following definition (see Remark 9).

Definition 16. *We denote by (w)Max-CSP$_c$($^+$)-$\mathcal{F}$ the restriction of (w)Max-CSP($^+$)-$\mathcal{F}$ where all functions in $\mathcal{F}$ are commutative. In this case, the C_i become sets $C_i = \{l^i_1, \dots, l^i_k\}$. We still impose that no pair of constraints are identical but the definition of identical remains $\mathcal{C}_i = \mathcal{C}_j$ iff $u_i = u_j$ and $C_i = C_j$.*

Remark 9. Up to now I have never seen the definition 16, but I think we need it to have the correspondences mentioned in Remark 11 (which are mentioned everywhere). For example, Max-CSP$^+$-$\mathcal{F}$ with $\mathcal{F} = \{f\}$, $f(x, y) \mapsto x \neq y$ does not correspond to Max-CUT (whereas Max-CSP^+_c-$\mathcal{F}$ do), but only to $w_{poly}MAXCUT$, as we could have $\mathcal{C}_1 = ((x, y), f, 1)$ and $\mathcal{C}_2 = ((y, x), f, 1)$, leading to an edge of weight 2.

Remark 10. The constraints that all the functions in $\mathcal{F}$ are distincts and that no pair of constraints are identical are generally not mentioned (for example in [11]) , but I think we need them as otherwise, Theorem 8 would become straightforward. Indeed, the first and easy part of the proof of [11] shows that we can always restrict ourselves to w_{poly}Max-CSP$^+$-$\mathcal{F}$ where weights are polynomially bounded($w \leq p(n + m)$), while the hard part of the proof is to show that w_{poly}Max-CSP$^+$-$\mathcal{F} \leq^{\alpha}_{AP}$ Max-CSP$^+$-$\mathcal{F}$. However,

if we allow to have equivalent constraints (or identical functions), we could simply duplicate the constraints to get the unweighted version (this duplication being polynomial as weights are polynomially bounded). Thus, I assume (and it seems) that Theorem 8 still works for the version of Max-CSP where identical constraints are forbidden AND for both versions (*i.e.* commutative or not).

We will use several time the following theorem. Indeed, the techniques showed here generally provide negative results for the weighted versions only. As the following theorem tells us that unweighted versions are as hard to approximate as their weighted counterpart, the inapproximability results transfer to unweighted versions.

Theorem 8 ([11]). *For any $\mathcal{F}$ that does not contains a unary constraint, for any $\alpha > 1$,*
- *wMax-CSP$^+$-$\mathcal{F}$ $\leq^{\alpha}_{AP}$ Max-CSP$^+$-$\mathcal{F}$*
- *wMax-CSP^+_c-$\mathcal{F}$ $\leq^{\alpha}_{AP}$ Max-CSP^+_c-$\mathcal{F}$*

Remark 11. As expected, this generalizes many well known problems (recall that I assume in the definition of SAT problems (or graphs problems) that we cannot repeat clauses (or edges))
- (w)Max-CSP$_c$-$\mathcal{F}$ with $\mathcal{F} = \{f\}$, $f(x, y, z) \mapsto x \vee y \vee z$ is (w)Max-3-SAT
- (w)Max-CSP^+_c-$\mathcal{F}$ with $\mathcal{F} = \{f\}$, $f(x, y) \mapsto x \neq y$ is (w)Max-CUT (each C_i corresponds to an edge)
- (w)Max-CSP^+_c-$\mathcal{F}$ with $\mathcal{F} = \{f\}$, $f(x, y) \mapsto x = y$ is (w)Max-UNCUT (each C_i corresponds to an edge)
- (w)Max-CSP^+_c-$\mathcal{F}$ with $\mathcal{F} = \{f_0, f_1\}$, $f_i(x, y, z) \mapsto x + y + z = i$ is (w)Max-E3-LIN

The following proposition is the reformulation of [4], lemma 2.2, but using our CSP vocabulary.

Proposition 3. *If V if a $PCP_{(c,s)}(r(n), q(n))$ verifier for a language L for $r(n) = O(log(n))$ and $q(n) = O(1)$, then there is a polynomial reduction from L to $gap_{c,s}$ wMax-CSP$^+$-$\mathcal{F}_V$. It the verifier is commutative, the reduction is even to $gap_{c,s}$ wMax-CSP^+_c-$\mathcal{F}_V$.*

Proof. Given an input x to decide, let us create an instance of wMax-CSP$_c$-$\mathcal{F}_V$. For each $j \in p(n)$ (recall that $p(n)$ is the upper bound on the size of the proof submitted to the verifier), we create a variable y_j that represents the j^{th} bit of the proof. For each $\tau \in \{0, 1\}^{r(n)}$, we set $C_{(x,\tau)} = y_{S(x,\tau)}$, and $f_{(x,\tau)} = f_{(V,x,\tau)}$ (computing this remains polynomial as $r(n) = O(log(n))$).
Thus, if $x \in L$, and according to Remark 2, then there exists an assignment of the y_j variables such that at least $c2^{r(n)}$ of random strings lead to an accepting state, and thus there are also at least $c2^{r(n)}$ satisfied constraints. Otherwise, for any assignment, using the same argument, at most $s2^{r(n)}$ constraints are satisfied.
Finally, we first rewrite the instance by removing idential functions (but brute force testing if for any τ, τ' if $f_{(x,\tau)} = f_{(x,\tau')}$, which is polynomial as $q(n) = O(1)$), and then removing identical constraints and introducing weights accordingly, getting thus an equivalent instance of wMax-CSP$^+$-$\mathcal{F}_V$, or wMax-CSP^+_c-$\mathcal{F}_V$ if V is commutative.
□

Remark 12. As soon as $\mathcal{F}_V$ does not contains unary constraint, Proposition 3 and Theorem 8 lead a reduction from L to $gap_{c,s}$ Max-CSP$^+$-$\mathcal{F}_V$ (or Max-CSP^+_c-$\mathcal{F}_V$ if V is commutative).

Let us now state the main theorem on PCP verifiers.

Theorem 9 (PCP Theorem, [6]). $NP = PCP_{(1,\frac{1}{2})}(O(log(n)), O(1))$

Using Proposition 3, we immediately get

Corollary 1. *$gap_{1,\frac{1}{2}}$ Max-CSP$^+$ $\mathcal{F}_V$ is NP-hard.*

Let us now review some inapproximability result for specific Max-CSP problems.

3.1 3 variables CSP

Theorem 10. *There exists a constant $\epsilon > 0$ such that $gap_{1,1-\epsilon}$ Max E3SAT is NP-hard.*

Proof. The proof is just a gap preserving reduction from $gap_{1,\frac{1}{2}}$ Max-CSP$^+$ $\mathcal{F}_V$. Without loss of generality, we can suppose that for any random string τ, the verifier makes exactly $k = q(n)$ queries. We can traduce each $f \in F_v$ (of arity $q(n)$) into $2^{q(n)}$ disjunctive clauses. Then, given an assignment a, if a given $f_i(a(C_i))$ is satisfied, then the $2^{q(n)}$ corresponding clauses are satisfied. Otherwise, at most $2^{q(n)} - 1$ clauses are satisfied. Thus, if $x \in L$, then all the clauses are satisfied, then the $2^{r(n)}2^{q(n)}$ clauses are satisfied. If $x \notin L$, then any assignment satisfies at most $2^{r(n)-1}$ of the f_i, and thus at most $2^{r(n)-1}(2^{q(n)} - 1) + 2^{r(n)-1}2^{q(n)}$ clauses are satisfied. Thus, we get that $gap_{1,1-\epsilon'}MAXEq(n)SAT$ is NP-hard, where $1 - \epsilon' = \frac{2^{q(n)+1}-1}{2^{q(n)+1}}$ is constant as $q(n) = O(1)$.
Then, we can also write a gap preserving reduction from $gap_{1,1-\epsilon'}MAXEq(n)SAT$ to $gap_{1,1-\epsilon}MAXE3SAT$ by using standard technique to traduce each q SAT clause into a set of 3 SAT clauses.

□

Let us now improve the gap for MAX E3 SAT by directly using Proposition 3 on a verifier that makes very few queries.

Theorem 11 (from [17], but stated as in [23]). *For every $\epsilon > 0$, $NP = PCP_{1-\epsilon,\frac{1}{2}+\epsilon}(O(log(n)), 3)$. Furthermore, the verifier behaves as follows: it uses its randomness to pick three entries i,j,k in the witness proof and a bit b, and accepts iff $y_i + y_j + y_k = b$ (i.e. $\mathcal{F}_V = \{f_0, f_1\}$ where $f_i(x, y, z) \mapsto x + y + z = i$).*

Corollary 2 ($\star^{ar}$). *For any ϵ, $gap_{1-\epsilon,\frac{1}{2}+\epsilon}MAXE3LIN$ is NP-hard*

Proof. Theorem 11 gives us a verifier with $\mathcal{F}_V = \{f_0, f_1\}$, $f_i(x, y, z) \mapsto x + y + z = i$. Thus, as V is commutative, according to proposition 3 and remark 12, $gap_{1-\epsilon,\frac{1}{2}+\epsilon}$ Max-CSP^+_c-$\mathcal{F}_V$ = $gap_{1-\epsilon,\frac{1}{2}+\epsilon}$ MAX E3 LIN is NP-hard.

□

Corollary 3 ($\star^{ar}$). *For any ϵ, $gap_{1,\frac{7}{8}-\epsilon}MAXE3SAT$ is NP-hard*

Proof. This is a direct consequence of corollary 2. Indeed, let us write a reduction from $gap_{1-\epsilon,\frac{1}{2}+\epsilon}MAXE3LIN$ to $gap_{1,\frac{7}{8}-\epsilon'}MAXE3SAT$. To any predicate on the form $x + y + z = 0$ we associate the three clauses $(x \vee y \vee \bar{z})$, $(x \vee \bar{y} \vee z)$, $(\bar{x} \vee y \vee z)$, and $(\bar{x} \vee \bar{y} \vee \bar{z})$ (and to any predicate on the form $x + y + z = 1$ we associate the same clauses by replacing each literal by its opposite). If a given equation is satisfied then the 4 corresponding clauses are also satisfied, and otherwise at most 3 clauses are satisfied. Thus, if at least x equations are satisfiable in MAX E3 LIN, then $4x$ clauses are satisfiable. Otherwise, if at most $\frac{x}{2-\epsilon}$ equations are satisfiable in MAX E3 LIN, then at most $3(\frac{x}{2-\epsilon}) + 4(x - \frac{x}{2-\epsilon})$ clauses are satisfiable, leading to the desired gap.

□

3.2 2 variables CSP

A first way to derive inapproximability results for 2 variables CSP is to construct gap preserving reduction from the previous 3 variables CSP to 2 variables CSP. For example, we can reduce from MAX E3 SAT to MAX 2 SAT by replacing a clause $C_k = x_1 \vee x_2 \vee x_3$ by the following set of ten clauses (called "gadget") on variables x_1, x_2, x_3 and a new variable y_k: $x_1, x_2, x_3, \overline{(x_1)} \vee \overline{(x_2)}, \overline{(x_2)} \vee \overline{(x_3)}, \overline{(x_1)} \vee \overline{(x_3)}, y_k, x_1 \vee \overline{(y_k)}, x_2 \vee \overline{(y_k)}, x_3 \vee \overline{(y_k)}$. The property satisfied by this gadget is that is C_k is satisfied, then 7 of the 10 clauses can be satisfied, and otherwise only 6 are satisfiable. Thus, we would get a gap preserving reduction from $gap_{\frac{8}{7}-\epsilon}MAXE3SAT$ to $gap_{\frac{56}{55}-\epsilon'}$. An important work on the construction of "optimal gadget" has been done in [24] where authors present a computed assisted method (based on linear programming) to find gadgets used in a large classe of reductions.
Another way to derive inapproximability results for 2 variables CSP is to directly look for PCP reading only 2 bits of proofs. Again, depending on the form of the test that the verifier makes, we will get hardness result for the corresponding constraint language $\mathcal{F}$.

Theorem 12 ([18], theorem 3). *For any $t, \frac{1}{2} < t < 1$, for any sufficiently small constant $\epsilon > 0$, there exists a $PCP_{1-\epsilon,1-\epsilon^t}(O(log(n)), 2)$ verifier V for the unique game, where V makes a linear test on the two bits, and more precisely*[6] *$\mathcal{F}_V = \{f\}$ where $f_i(x, y) \mapsto x = y$.*

As Max-CSP^+_c-$\mathcal{F}_V$ for the previous verifier V is a special case of $MAXE2LIN$, the reduction of Proposition 3 immediately provide the following corollary.

Corollary 4. *Under UGC, for any $t, \frac{1}{2} < t < 1$, for any sufficiently small constant $\epsilon > 0$, $gap_{1-\epsilon,1-\epsilon^t}MAXE2LIN$ is NP-hard.*

Let us now turn to max cut, which is the restriction of $MAXE2LIN$ where all the equations have the form $x + y = 1$.

Theorem 13 ([19], section 8.4, or theorem 1). *Assuming that $gap_{1-\eta}\gamma UGC$ is NP hard for any sufficiently small constant η and γ, for any constant $-1 < \rho < 0$ and $\epsilon > 0$, there is a $PCP_{(\frac{1}{2}-\frac{1}{2}\rho),\frac{arccos\rho}{\pi}+\epsilon}(O(log(n)), 2)$ verifier V for the unique game where $\mathcal{F}_V = \{f\}$ where $f(x, y) \mapsto x \neq y$.*

As Max-CSP^+_c-$\mathcal{F}_V$ for the previous verifier V corresponds exactly to $MAXCUT$, the reduction of Proposition 3 immediately provide the following corollary.

Corollary 5 ($\star$). *Under UGC, for any $-1 < \rho < 0$ and $\epsilon > 0$, $gap_{(\frac{1}{2}-\frac{1}{2}\rho),\frac{arccos\rho}{\pi}+\epsilon}MAXCUT$ is NP-hard, and by choosing ρ appropriately, we get that the $\alpha_{GW} - \epsilon$ inapproximability result.*

Remark 13. The reduction of Proposition 3 for the case where $\mathcal{F}_V = \{f\}$ where $f(x, y) \mapsto x \neq y$ is sometimes directly written as producing a graph: for every position $i \leq p(n)$ in the proof we create a vertex v_i, and the weight of each edge $\{v_i, v_j\}$ is equal to $|\{\tau \in \{0, 1\}^{r(n)} | S(x, \tau) = \{v_i, v_j\}\}|$.

The stame story holds for Max 2 SAT: A first way to derive inapproximability result for Max 2 SAT is via gap preserving reduction and gadgets (for example, replacing $x+y = 0$ by $x \vee \bar{y}$, $\bar{y} \vee x$ as mentioned in [18]), but we can also directly the appropriate shaped verifier of [19] (p20)that have $\mathcal{F}_V = \{f_1, f_2\}$ where $f_1(x, y) \mapsto x \vee y$ and $f_2(x, y) \mapsto \bar{x} \vee \bar{y}$.

[6] The fact that we even have $\mathcal{F}_V = \{f\}$ where $f_i(x, y) \mapsto x = y$ (and not only $F_V = \{f_0, f_1\}$ where $f_i(x, y) \mapsto x + y = i$, which is the direct traduction of making a linear test on two bits) is not explicitly mentioned is [18], but when looking at the verifier p5, it seems that they are only tests of the form "x=y"

Acknoledgment

Merci aux Jcalm 2015!

References

1. http://www-cc.cs.uni-saarland.de/media/oldmaterial/advcct.pdf.
2. http://www.cs.cmu.edu/ venkatg/talks/csp-approx-tutorial.pdf.
3. www.cs.nyu.edu/ khot/pcp-course.html.
4. Sanjeev Arora. Probabilistic checking of proofs and hardness of approximation problems. PhD thesis, Princeton University, 1994.
5. Sanjeev Arora and Boaz Barak. Computational complexity: a modern approach. Cambridge University Press, 2009.
6. Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. Journal of the ACM (JACM), 45(3):501–555, 1998.
7. Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of np. Journal of the ACM (JACM), 45(1):70–122, 1998.
8. Per Austrin. Towards sharp inapproximability for any 2-csp. SIAM Journal on Computing, 39(6):2430–2463, 2010.
9. Nikhil Bansal and Subhash Khot. Optimal long code test with one free bit. In Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on, pages 453–462. IEEE, 2009.
10. Mihir Bellare, Oded Goldreich, and Madhu Sudan. Free bits, pcps, and nonapproximability—towards tight results. SIAM Journal on Computing, 27(3):804–915, 1998.
11. Pierluigi Crescenzi, Riccardo Silvestri, and Luca Trevisan. On weighted vs unweighted versions of combinatorial optimization problems. Information and Computation, 167(1):10–26, 2001.
12. Lars Engebretsen and Jonas Holmerin. Towards optimal lower bounds for clique and chromatic number. Theoretical Computer Science, 299(1):537–584, 2003.
13. Uriel Feige, Shafi Goldwasser, Laszlo Lovász, Shmuel Safra, and Mario Szegedy. Interactive proofs and the hardness of approximating cliques. Journal of the ACM (JACM), 43(2):268–292, 1996.
14. Oded Goldreich. On promise problems. memory of Shimon Even (1935–2004). ECCC, TR05-018 (January 2005), 2005.
15. Oded Goldreich. Using the fglss-reduction to prove inapproximability results for minimum vertex cover in hypergraphs. In Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation, pages 88–97. Springer, 2011.
16. Johan Hrastad. Clique is hard to approximate within n 1-&epsiv. In Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on, pages 627–636. IEEE, 1996.
17. Johan Hrastad. Some optimal inapproximability results. Journal of the ACM (JACM), 48(4):798–859, 2001.
18. Subhash Khot. On the power of unique 2-prover 1-round games. In Proceedings of the thiry-fourth annual ACM symposium on Theory of computing, pages 767–775. ACM, 2002.
19. Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for max-cut and other 2-variable csps? SIAM Journal on Computing, 37(1):319–357, 2007.
20. Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within 2- ε. Journal of Computer and System Sciences, 74(3):335–349, 2008.
21. Mohammad Mahmoody and David Xiao. On the power of randomized reductions and the checkability of sat. In Computational Complexity (CCC), 2010 IEEE 25th Annual Conference on, pages 64–75. IEEE, 2010.
22. Ran Raz. A parallel repetition theorem. SIAM Journal on Computing, 27(3):763–803, 1998.
23. Luca Trevisan. Inapproximability of combinatorial optimization problems. The Computing Research Repository, 2004.
24. Luca Trevisan, Gregory B Sorkin, Madhu Sudan, and David P Williamson. Gadgets, approximation, and linear programming. SIAM Journal on Computing, 29(6):2074–2097, 2000.
25. David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. In Proceedings of the thirty-eighth annual ACM symposium on Theory of computing, pages 681–690. ACM, 2006.