Non-approximability results for optimization problems on bounded degree instances

Luca Trevisan*
luca@eecs.berkeley.edu
Computer Science Division, University of California at Berkeley

ABSTRACT

We prove some non-approximability results for restrictions of basic combinatorial optimization problems to instances of bounded "degree" or bounded "width." Specifically:

- We prove that the Max 3SAT problem on instances where each variable occurs in at most B clauses, is hard to approximate to within a factor $7/8+O(1/\sqrt{B})$, unless $RP = NP$. Håstad [18] proved that the problem is approximable to within a factor $7/8+1/64B$ in polynomial time, and that is hard to approximate to within a factor $7/8+1/(\log B)^{\Omega(1)}$. Our result uses a new randomized reduction from general instances of Max 3SAT to bounded-occurrences instances. The randomized reduction applies to other Max SNP problems as well.

- We observe that the Set Cover problem on instances where each set has size at most B is hard to approximate to within a factor $\ln B - O(\ln \ln B)$ unless $P = NP$. The result follows from an appropriate setting of parameters in Feige's reduction [11]. This is essentially tight in light of the existence of $(1 + \ln B)$-approximate algorithms [20, 23, 9]

- We present a new PCP construction, based on applying parallel repetition to the "inner verifier," and we provide a tight analysis for it. Using the new construction, and some modifications to known reductions from PCP to Hitting Set, we prove that Hitting Set with sets of size B is hard to approximate to within a factor $B^{1/19}$. The problem can be approximated to within a factor B [19], and it is the Vertex Cover problem for $B = 2$. The relationship between hardness of approximation and set size seems to have not been explored before.

- We observe that the Independent Set problem on graphs having degree at most B is hard to approximate to within a factor $B/2^{O(\sqrt{\log B})}$, unless $P = NP$. This follows from a combination of results by Clementi and Trevisan [10], Samorodnitsky and Trevisan [28] and Reingold, Vadhan and Wigderson [27]. It had been observed that the problem is hard to approximate to within a factor $B^{\Omega(1)}$ unless $P = NP$ [1]. An algorithm achieving a factor $O(B \log\log B/\log B)$ is also known [21, 2, 30, 16].

*Work supported by a Sloan Research Fellowship and an NSF Career Award.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
STOC'01, July 6-8, 2001, Hersonissos, Crete, Greece.
Copyright 2001 ACM 1-58113-349-9/01/0007 ...$5.00.

1. INTRODUCTION

We consider four fundamental combinatorial optimization problems (Max 3SAT, Set Cover, Hitting Set, Independent Set), and we consider the restriction of the problems to instances where a "degree" or "width" parameter is bounded by a constant. Such restricted problems are known to admit better approximation algorithms than the general versions, the quality of the approximation being related to the value of the degree/width parameter. There is, indeed, an extensive body of algorithmic work devoted to such restricted problems, but the existence of matching non-approximability results seems to have been largely unexplored. In this paper we prove strong (in some cases, almost tight) non-approximability results for such restricted problems. The results for Set Cover and Independent Set are implicit in previous work (but we find it useful to spell them out explicitly for the record). Our main contributions are for Max 3SAT (where we introduce a new randomized reduction from general instances to instances with bounded occurrences of variables) and for Hitting Set (that involves a new PCP construction).

Max 3SAT with bounded occurrences

For the Max 3SAT problem, the parameter of interest is an upper bound on the number of clauses in which a variable can appear. We consider the Max E3SAT problem (i.e. the version of Max 3SAT where each clause contains exactly 3 variables) where no variable occurs in more than B clauses; Håstad [18] proved that this problem can be approximated to within a factor $7/8+O(1/B)$. For constant B this is better than what is possible for the general Max E3SAT problem, for which no approximation algorithm with performance ratio $7/8+\epsilon$, $\epsilon > 0$, can exists, unless $P = NP$ [17]. It was also known that the problem is Max SNP-hard even for small B [25], and Håstad [18] observed that there is a constant $c > 0$ such that it is NP-hard to achieve an approximation ratio of the form $7/8 + 1/(\log B)^c$. In Section 2 we improve this hardness result to $7/8 + O(1/\sqrt{B})$. Our reduction shows in general that an arbitrary instance of Max 3SAT (or, in particular, an arbitrary instance of Max E3SAT) can be converted to an instance where no variable occurs more than B times, in such a way that any ρ-approximate solution for the new instance yields a $(\rho - O(1/\sqrt{B}))$-approximate solution for the original instance. The reduction (that is randomized, and works with constant probability) applies to other constraint satisfaction problems as well.

Set Cover with sets of bounded size

For Set Cover, the natural parameter of interest is the size of the sets. The standard greedy algorithm achieves an approximation ratio of $1 + \ln B$, if B is an upper bound on the size of the sets [20, 23, 9]. If sets can have arbitrary size then the greedy algorithm achieves at least an approximation $1+\ln n$, where n is the size of the instance, and Feige [11] proved that, for any $\epsilon > 0$, it is infeasible to achieve a $(1-\epsilon)\ln n$ approximation, unless NP has quasi-polynomial time algorithms. It was also known that the problem remains Max SNP-hard even if $B = 3$, however it seems that the asymptotic relation between B and approximability has not been explicitly explored. We note that it is implicit in [11] that there is a constant c such that it is NP-hard to achieve an approximation better than $\ln B - c \ln \ln B$.

Hitting Set with sets of bounded size

For Hitting Set, again the natural parameter of interest is set size. (By the way, recall that Hitting Set with sets of size at most B is the same problem as Set Cover with the restriction that no element appears in more than B sets.) If B is an upper bound on the size of the sets, the best known approximation achieves a factor of B [19], however if the size of the sets is unbounded, than it is possible to achieve an approximation $1 + \ln n$, where n is the size of the instance. (The unrestricted version of Hitting Set is the same problem as the unrestricted version of Set Cover.) It is conceivable that an approximation ratio of B is the best possible, however current techniques are still far from being able to prove such a result. In particular, recall that Vertex Cover is the special case of Hitting Set where $B = 2$, and currently the strongest known hardness result for Vertex Cover is a factor of $7/6$. It is possible, from known reductions, to prove a hardness result of the form B^c, for some constant $c > 0$ that would be presumably very small – and hard to figure out. Hardness results for Set Cover or Hitting Set depend, as a starting point, on PCP construction where the verifier makes a small number of queries (ideally, two), receives l-bits long answers (where l is a parameter of the construction); the verifier must have perfect completeness and soundness $2^{-\Omega(l)}$. Håstad [17] gives such a verifier that works for constant (or slowly growing) l (which is good enough for our purposes), makes 3 queries, and has soundness 2^{-l}; unfortunately the verifier does not have perfect completeness and is unsuitable for these reductions. The best available construction is due to Raz [26]; the verifier makes two queries (and decides whether to accept or reject in a particularly simple way, which is useful in the reduction), has perfect completeness, and soundness $2^{-\gamma l}$, where $\gamma > 0$ is some constant. Unfortunately there are no known explicit estimates for γ, and it is likely to be very small.

We construct a new PCP system where the verifier makes 3 queries, has perfect completeness, and soundness $(3/4)^l$. Our system is obtained by running l *parallel repetitions* of a *inner verifier* due to Håstad [17]. Håstad's verifier makes three queries, gets Boolean answers, has perfect completeness, and soundness $3/4$. We show that executing l copies of that verification in parallel[1] gives an inner verifier with soundness $(3/4)^l$. An inner verifier is a testing procedure for a certain coding-theoretic problem. When composed with other verifiers, an inner verifier yields a PCP system with about the same soundness, so, by composition, we get a PCP system of soundness close $(3/4)^l$. We stress that it matters that we do not apply parallel repetition to the PCP system of [17], but rather to the inner verifier. It is probably not true that an l-fold parallel repetition of the PCP system of [17] has soundness close to $(3/4)^l$.

Once we have our PCP system, we want to reduce it to Hitting Set. The Lund-Yannakakis [24] reduction from PCP to Set Cover (and generalizations/improvements/simplifications as in [7, 11, 5]) requires the PCP system to have a particular way of deciding whether to accept or reject after making its queries. Roughly speaking, the value of the answer to one of the queries must "imply" the value of the answers to the other queries. In Section 5 we describe a reduction that, to the best of our knowledge, is the first one to work starting from an arbitrary PCP system. Using our reduction and our PCP system we are able to prove that Hitting Set is hard to approximate to within a factor of about $B^{1/19}$, where B is an upper bound on set size.

Independent Set in bounded-degree graphs

For the Independent Set problem, a natural restriction is to consider graphs of bounded degree. For graphs of maximum degree B the problem is known to be approximable to within a factor $O(B \log\log B / \log B)$, as proved by Vishwanathan [30] (in a personal communication to Haldorsson cited in [16]) using results from [21, 2]. A hardness result of the form B^c for some $c > 0$ was proved in [1]. We observe that combining results of [10], [28] and [27][2] it is possible to prove a hardness of $B/2^{O(\sqrt{\log B})}$. This result is presented in Section 6

Other related work

Finally, we stress that the focus of this paper is exclusively on the asymptotic relation between hardness of approximation and degree/width parameters.

Typically, the focus of previous research on the hardness of bounded degree/width problems has been on the case of very small value of the bound parameter, typically the minimal value that makes the problem NP-hard. For example, considerable effort has gone into figuring out the non-approximability of vertex cover and independent set in graphs of maximum degree 3 or 4, or of Max 3SAT and other constraint satisfaction problem where each variable occurs a small number of times (see [8] and references therein). The

[1] To be precise, our inner verifier differs in a couple of subtle ways from an actual l-fold parallel repetition of Håstad' verifier, and both differences are essential for the analysis.

[2] Specifically, [28] provide the PCP system to start with, [10] provide a variation of the FGLSS reduction [12] that produces bounded-degree graphs, and [27] give an explicit construction of expanders to be used in the reduction, thus making the reduction deterministic rather than probabilistic.

interest in such special cases is that they are useful as intermediate steps in reductions to other important problems. Our reductions do not work if the goal is to obtain instances with very small degree-width parameters, and our results are incomparable with the results of [8].

Even though the case of small parameters is perhaps better motivated, the asymptotic cases deserved some further study. As far as we know, the only previous papers with an asymptotic angle are [1] and [10], that considered the hardness of independent set and vertex cover in bounded-degree graphs, as a function of the degree.

2. MAX 3SAT

Our starting point is the following hardness of approximation result, due to Håstad (recall that Max E3SAT is the version of Max SAT where each clause contains exactly 3 literals).

THEOREM 1 ([17]). *For every $\delta > 0$, it is NP-hard to distinguish a satisfiable instance of Max E3SAT from an instance where at most a fraction $7/8 + \delta$ of the clauses can be simultaneously satisfied.*

Let a (sufficiently large) parameter B be fixed for the rest of this section. We will start from an instance φ of Max E3SAT and produce, in several steps (one step will be a randomized procedure), an instance φ_B with the following properties:

- No variable occurs in more than B clauses of φ_B
- If φ is satisfiable, then φ_B is satisfiable
- With probability at least $3/4 - o(1)$ over the random choices made in the construction of φ_B, if there is an assignment that satisfies at least a fraction $7/8+5/\sqrt{B}$ of the clauses of φ_B, then there is an assignment that satisfies at least a fraction $7/8 + 1/\sqrt{B}$ of the clauses of φ.

It follows from the above properties and from Theorem 1 that if there were a $(7/8 + 5/\sqrt{B})$-approximate algorithm for Max E3SAT restricted to instances with at most B occurrences, then $RP = NP$.

The construction of φ_B will proceed in three steps: we first construct a weighted instance φ_w that is completely equivalent to φ; then we probabilistically construct an instance φ_R by sampling clauses from φ_w. We will then show that with high probability, every assignment satisfies more or less the same fraction of clauses in φ_w and φ_R. Also with high probability, in φ_R most variables occur no more than B times, and we create our final instance φ_B by deleting some clauses from φ_R, so as to make sure that no variable occurs more than B times. Even after the deletion, there is still a high probability that every assignment satisfies more or less the same fraction of clauses in φ_w and in φ_B.

So, let φ be an instance of Max 3SAT made of clauses $C_1, \ldots, C_m$ over variables $x_1, \ldots, x_n$. Let us denote by o_i the number of clauses where variable x_i occurs. Let us also introduce o_i new variables y_i^j, with $j \in \{1, \ldots, o_i\}$ for each variable x_i of φ. Let $N = \sum_i o_i = 3m$ be the number of such variables.

The instance φ_w is over variables $\{y_j^i\}$. For each clause C of φ, with variables $\{x_a, x_b, x_c\}$, the formula φ_w contains $o_a o_b o_c$ copies of C, each one having weight $1/o_a o_b o_c$, and each one featuring one of the possible substitution of x_a by a variable $y_a^{j_1}$, of x_b by a variable $y_b^{j_2}$ and of x_c by a variable $y_c^{j_3}$. If φ is satisfiable, then clearly φ_w is satisfiable. Suppose that there is an assignment to the variables $\{y_i^j\}$ that satisfies clauses in φ_w of total weight k. Then consider the random assignment to the variables $\{x_i\}$ where x_i is given value TRUE with probability proportional to the number of variables $\{y_i^j\}$ that have value TRUE in the original assignment. One can verify that k is the average number of clauses of φ that is satisfied by such a random assignment. Hence, if there is an assignment for the variables in φ_w that satisfies clauses of total weight at least k, then there is an assignment for the variables in φ that satisfies at least k clauses.

The random formula φ_R is defined by the following process: independently for $(B/e^2)N$ times, pick at random a clause from φ_w, with probability proportional to its weight.

Consider each variable y_i^j. In φ_w, it occurred in clauses of total weight 1, and the total weight of all clauses was m. It follows that in the distribution that we use to sample the clauses of φ_R, there is a probability $1/m$ that y_j^i be sampled each time. On average, then, each variable y occurs in B/e^2 clauses. Furthermore, the probability that it occurs in $k \geq B$ clauses is at most e^{-k} (we are using the Chernoff bound stated as Lemma 9). Since we want no variable to occur in more than B clauses, if y is such that it occurs in more than B clauses, we will delete some, so that, after the deletion, y occurs in B clauses. This deletion process creates the final instance φ_B. The average number of clauses deleted for y is at most $\sum_{k>B}(k-B)e^{-k} < 1$ for sufficiently large B. Hence, the average number of deleted clauses is at most N. With probability at least $1 - 1/4$, we will not delete more than $4N$ clauses. Just to make calculations easier, we replace each deleted clause with a clause made of fresh variables. The new clauses are always trivially satisfiable, and variables in the new clauses occur exactly once.

Observe that if φ_w is satisfiable, then with probability 1 we have that φ_R is satisfiable and that φ_B is satisfiable (by the same assignment).

Consider now an arbitrary assignment a to the $\{y_j^i\}$ variables, and let ρm be total weight of the clauses satisfied in φ_w by a. Then, when we pick clauses for inclusion in φ_R, each time there is a probability ρ that the picked clause is satisfied by a, and a probability $1-\rho$ that the picked clause is not satisfied by a. Overall, the probability that more than $(\rho + \epsilon)m$ of the clauses of φ_R are satisfied by a is at most $e^{-2\epsilon^2 m} = e^{-2\epsilon^2 BN/e^2}$ (we are using the Chernoff bound of Lemma 10). If $B > e^2/\epsilon^2$, then the above probability is less than 2^{-2N}. So, fixing, say $\epsilon = 3/\sqrt{B}$, and using a union bound, if there is no assignment that satisfies more than ρN clauses in φ_w, then there is a probability at least $1 - 2^{-N}$ that no assignment satisfies more than $(\rho+3/\sqrt{B})m$ clauses of φ_R. With probability at least $3/4$, we deleted (and replaced with new clauses) no more than $4N < m/\sqrt{B}$ clauses (the inequality is true for sufficiently large B). So with probability at least $3/4 - 2^N$ the following is true: if there is no assignment that satisfies more than ρN clauses in φ_w, then there is no assignment that satisfies more than $(\rho+4/\sqrt{B})m$ clauses of φ_B. This completes the proof of the result stated at the beginning of the section.

REMARK 2. *For the sake of simplicity of presentation, the* *analysis of the reduction is not fully optimized, and it is clearly possible to get a constant better than 4 in the above analysis.*

REMARK 3. *We observe that the reduction creates instances where clauses could be repeated. However, since no variable occurs in more than B clauses, no clause can be repeated more than B times. To achieve a stronger result, it is possible to modify the reduction so as to generate instances without repeated clauses (we omit details from this preliminary version).*

3. FEIGE'S REDUCTION AND SET COVER

The goal of this section is to indicate how the results of [11] imply that Set Cover with sets of size B is hard to approximate to within a $\ln B - O(\ln\ln B)$ factor. This section is probably best read having [11] at hand.

Feige [11] already describes his reduction in general terms, setting the parameters only at the end, so we just have to indicate what different setting of parameters is appropriate to achieve hardness for the case of sets of size at most B. We will describe the reduction in terms of a parameter m (which is the same m used in [11], except with a different setting) that will be fixed later (jumping ahead, m will be $B/\text{poly}\log B$).

The PCP construction used as a starting point is analyzed in [11, Lemma 2.3.1]. The construction has parameters l and k. The verifier interacts with k provers, and asks each prover a question that is answered with a string of $2l$ bits. If the statement being proven is correct, and the proof is valid, then for all random strings the verifier receives k answers that are pairwise consistent. If the statement being proven is not correct, then, for any proof, there is a probability at least $k^2 \cdot 2^{-cl}$ over the random choices of the verifier that the k answers received by the verifier are such that any two of them are inconsistent. In the above expression, $c > 0$ is a small constant independent of all other parameters. (A detailed description and analysis of the verifier, and a definition of the notion of consistency, can be found in [11][Section 2.3].) We will use this result with $l = (4/c)\log\ln m$, and $k = \ln m/3\ln\ln m$. Note that the PCP construction requires k and l to be such that there exists an error-correcting code with k codewords of length l, such that every codeword has weight $l/2$ and the minimum distance is $l/3$. This condition is satisfied by our choice of l and k, since c is quite small.

The reduction from PCP to Set Cover uses the following gadget: a family $\{C_{i,j}\}$ of subsets of a universe U (where $|U| = m$), where $i = 1,\ldots,2^l$ and $j = 1,\ldots,k$. The family has the following properties:

- For each fixed i, the sets $C_{i,1},\ldots,C_{i,k}$ form a partition of U.
- If we take $\le (k-2)\ln m$ sets $C_{i,j}$, no two of them with the same i, then their union does not cover $[m]$.

The existence of such a family (for sufficiently large m), is stated in [11, Lemma 3.2] (one has to make the substitutions $f(k) = 2/k$) and $L = 2^l$). Notice that, for constant B, all the parameters m, l, k are constants, and so the family has constant size and can be found deterministically by brute-force search.

Let R be the number of random strings used by the verifier. The reduction uses R disjoint copies of the gadget; for a random string r, we denote by U^r the corresponding copy of the universe, and by $C^r_{i,j}$ the corresponding copy of the set $C_{i,j}$.

The instance of Set Cover created by the reduction (as described in [11, Section 4] is as follows. The universe for the instance is the union of the sets U^r. The set system contains a set $S_{i,q,a}$ for each $i = 1,\ldots,k$ (identifying one of the k provers), each query q that can be made to prover i, and each possible answer a. The set $S_{i,q,a}$ is the union, over all r such that the verifier, on random string r makes query q to prover i, of the set C^r_{i,a_r} where a_r is a value that depends on a and r. It can be verifier that for each i and q there are at most $3^{l/2} = 2^{\Theta(l)}$ random strings r such that the verifier makes query q to prover i. Furthermore, each set $C^r_{i,a,r}$ has size at most m. Therefore, in the instance produced by the reduction, all sets have size at most $m2^{\Theta(l)} = m2^{\Theta(\log\log m)}$. We fix m so that no sets has size more than B.

The hardness of approximation proved by this reduction is of the form $(1 - 4/k)\ln m$, provided that $k^2 2^{-cl} < 4/k^3(\ln m)^2$. That is, we need $l > \frac{1}{c}(5\log k + 2\log\ln m)$, and recalling that $k = \ln m/3\ln\ln m < \ln m/3$, it's enough if $l = (4/c)\log\ln m$, as assumed at the beginning of the section.

Thus the hardness of approximation is of the form $\ln m - 12\ln\ln m$, and recalling that $B = m \cdot 2^{O(\log\log m)}$, it can be written as $\ln B - O(\ln\ln B)$.

4. OUR NEW PCP CONSTRUCTION

Let us first briefly review the terminology in PCP construction. A verifier is confronted with an instance of an NP-complete language – for concreteness, let us say an instance φ of 3SAT – and is given oracle access to an alleged proof P that φ is satisfiable. After looking at φ, running in polynomial time (in the size of φ) and tossing a logarithmic (in the size of φ) number of random bits, the verifier decides to make a series $q_1,\ldots,q_k$ of queries into the oracle proof (k is called the *query complexity* of the verifier), thus receiving answers $P(q_1),\ldots,P(q_k)$. Each answer is a string in $\{0,1\}^l$, and l is called the *answer size* of the proof system. After examining the answers, the verifier decides whether to accept or reject. If there is a constant c such that for every satisfiable φ there is a P such that the verifier accepts with probability $\ge c$, then the verifier is said to have *completeness* at least c; ideally $c = 1$, in which case the verifier is said to have *perfect completeness*. If there is a constant s such that for every unsatisfiable φ and for any proof P the verifier accepts with probability at most s, then the verifier is said to have *soundness* at most s. We need a proof system with small query complexity, perfect completeness, and a strong relation between answer size and soundness.

Håstad [17] shows, for every $\epsilon > 0$, the existence of a PCP characterization of NP where the verifier makes three queries, it has perfect completeness, the answer size is 1, and the soundness is $3/4 + \epsilon$.

In Section 4.1 we sketch a proof of the following result.

THEOREM 4. *For any $\epsilon > 0$ and positive integer l, there is a PCP characterization of NP where the verifier makes 3 queries, has answer size l, perfect completeness, and soundness $(3/4)^l + \epsilon$.*

A naive approach would be to define a new proof system where each entry in a proof for the new system contains the answer to l queries in the original system. The new verifier runs in parallel l computations of the original verifier, and get l triples of questions $(a_1, b_1, c_1), \ldots, (a_l, b_l, c_l)$, and it then reads the three entries indexed by $(a_1, \ldots, a_l)$, $(b_1, \ldots, b_l)$, $(c_1, \ldots, c_l)$ in the new proof. It then accepts if the answers are acceptable for all the l computations. We do not know how to analyze this approach. In general, analyzing parallel repetition of proof system is an exceedingly difficult task, and typically the soundness of a proof system obtained by l parallel repetition can be higher than the original soundness raised to the l-th power.

The PCP construction of [17] is obtained by composing two proof systems, an "outer" proof system due to Raz [26] and an "inner" proof system described and analyzed in [17]. We apply parallel repetition to the inner verifier, and we show that we can drive the soundness down at an optimal rate within the inner verifier. Composition gives a PCP construction with the right soundness. For readers who are familiar with the constructions and terminology of [6, 17], we remark that our analysis takes advantage of the way in which the "long code" over a big alphabet can be "folded."

4.1 Proof or Theorem 4

4.1.1 Background

Let us start with an abstract view of the proof system of Raz[26]. The verifier is given a 3SAT instance φ and has oracle access to two proofs, say P and Q, where P's entries are over some alphabet $[u]$ and Q's entries are over some alphabet $[w]$. Based on its coin tosses, the verifier picks a pair of queries (p, q), and also a function $\pi_{p,q} : [w] \to [u]$. The verifier reads $a = P(p)$ and $b = Q(q)$ and accepts if and only if $a = \pi(b)$. If formula φ is satisfiable, there is a proof pair (P, Q) that is accepted with probability one, otherwise no proof pair is accepted with probability higher than $1/u^\gamma$ where $\gamma > 0$ is some constant independent of the other parameters. For any t, there is such a construction with $u = 2^t$ and $w = 7^t$.[3]

One gets efficient PCP systems by taking a proof in the format above, and further encoding the entries of P and Q using the *long code*. The long code of an element of $a \in [u]$ is the function $A : ([u] \to \{-1, 1\}) \to \{-1, 1\}$ that on input a function $f : [u] \to \{-1, 1\}$ returns the value $A(f) = f(a)$.

In an efficient PCP system, the verifier has access to proofs LP and LQ, where, for every p, $LP(p)$ is supposed to contain the long code of $P(p)$, where P is an acceptable proof for Raz's verifier. Similarly, LQ is supposed to be an entry-wise encoding of a proof Q.

The efficient verifier picks entries p, q and function π with the same distribution as Raz's verifier, and then considers the functions $A = LP(p)$ and $B = LQ(q)$: the verifier's goal is to determine whether A and B are, at least "approximately," the encoding of some elements a and b such that $\pi(b) = \pi(a)$. The notion of being approximately correct is formalized as follows. There is a probabilistic procedure $\text{DECODE}()$ that given a function $A : ([u] \to \{-1, 1\}) \to \{-1, 1\}$ outputs an element distributed over $[u]$. The testing procedure tries to determine if, at least, there is a noticeable probability that $\text{DECODE}(A) = \pi(\text{DECODE}(B))$, where the two applications of $\text{DECODE}()$ are done with independent coin tosses.

Before getting to Håstad's verifier we have to introduce another technicality. If $A : ([u] \to \{-1, 1\}) \to \{-1, 1\}$, we define the *folding* of A, denoted A', as the function such that $A'(f) = A(f)$ if $f(1) = 1$, and $A'(f) = -A(-f)$ if $f(1) = -1$. First of all, notice that if A is a long code then it is always true that $A(f) = -A(-f)$, therefore $A' = A$. Furthermore, for every A, we have $A'(f) = -A'(-f)$. In particular, for a random f, $A'(f)$ is equally likely to be 1 or -1. Finally, notice that it is possible to simulate oracle access to A' if we have oracle access to A.

We can now give a high-level description of the verifier of [17]. The construction uses a parameter $\delta > 0$.

After choosing queries (p, q) and function π as Raz's verifier, let us call A the restriction of the proof to $LP(p)$ and B the restriction of the proof to $LQ(q)$. The verifier picks uniformly at random functions $f : [u] \to \{-1, 1\}$ and $g : [w] \to \{-1, 1\}$, it then picks a function $h : [w] \to \{-1, 1\}$ according to a complicated distribution that depends on δ. Finally, the verifier accepts if the following condition is satisfied

$$A'(f) = -1 \text{ or } B'(g) = B'(g \cdot ((f \circ \pi) \wedge h))$$

We can check that the probability that the verifier accepts is equal to

$$\frac{3}{4} + \frac{1}{4} \mathop{\mathbf{E}}_{A,B,\pi,f,g,h}[B'(g) = B'(g \cdot ((f \circ \pi) \wedge h))] +$$

$$\frac{1}{4} \mathop{\mathbf{E}}_{A,B,\pi,f,g,h}[B'(g) = B'(g \cdot ((f \circ \pi) \wedge h))]$$

The analysis is completed by the following two results.

LEMMA 5 ([17]). *Let $B([w] \to \{-1, 1\}) \to \{-1, 1\}$, let be π be a random variable ranging over functions $\pi : [w] \to [u]$, and let f, g, h be random functions distributed as in Håstad's verifier with parameter δ, and suppose that π is distributed in such a way that there are constants $c_1, c_2 > 0$ such that for every set $\beta \subseteq [w]$*

$$\Pr_\pi[|\{\pi(b) : b \in \beta\}| \geq c_1 \log \beta] \geq 1 - 1/|\beta|^c$$

Then

$$\mathop{\mathbf{E}}_{B,\pi,f,g,h}[B'(g) B'(g \cdot ((f \circ \pi) \wedge h))] \leq 3\delta$$

The above statement is actually a slight generalization of a result of [17], and it appears in the above version in [14, Lemma 3.4]

The "hashing condition" about π, that it maps with high probability big sets into relatively big sets, is satisfied by the distribution of the function π in Raz's verifier (this is proved, for our presentation of Raz's verifier, in [14] – a similar result for the standard presentation is in the appendix of the full version of [17]).

Note that no assumption is made on B in Lemma 5. The use of folding is necessary in the statement. If B is identically equal to 1, then the expectation of $B(\cdot)B(\cdot)$ is always 1.

[3] This is the style of presentation of [15, 29, 28]. The standard presentation has $w = 8^t$, and the verifier accepts iff $a = \pi(b)$ and b satisfies some additional property. Having the additional property creates minor complications in the analysis of the verifier, but in the standard presentation the function π has a simpler structure than in this presentation.

LEMMA 6 ([17]). *There exists a δ', that depends only on δ, such that if LP and LQ are such that*

$$\mathop{\mathbf{E}}_{A,B,\pi,f,g,h}[A'(f)B'(g)B'(g\cdot((f\circ\pi)\wedge h))] > \delta$$

then

$$\mathop{\mathbf{Pr}}_{A,B,\pi}[\textsc{Decode}(A) = \pi(\textsc{Decode}(B))] > \delta'$$

where f,g,h are as in Håstad's verifier with parameter δ, and $A = LP(p)$ and $B = LQ(q)$, where (p,q,π) is the distribution of queries and function generated by Raz's verifier.

Again, the above lemma is implicit in [17], and the proof of this particular version can be derived from [14, Lemma 3.5].

Now, if the verifier accepts with probability more than $3/4+\delta$, the condition of Lemma 6 is satisfied. It can then be showed that from LP and LQ one can "decode" proofs P and Q for Raz's verifier that are accepted with probability at least δ'. If we chose u so that $1/u^\gamma < \delta'$, it then must be the case that the formula being proved is satisfiable. So, if we have an unsatisfiable formula, no proof can be accepted with probability greater than $3/4+\delta$.

4.1.2 Our Construction

Let us now get to our construction. Let us fix an integer parameter l and a constant $\delta > 0$. We also fix a sufficiently large u.

For an element $a \in \{0,1\}^u$, the long l-code of a is a function $A : ([u] \to \{-1,1\}^l) \to \{-1,1\}^l$, such that for every $\bar f : [u] \to \{-1,1\}^k$ we have $A(\bar f) = \bar f(a)$. We denote by $A_i(\bar f)$ the i-th bit of the vector $A(\bar f)$.

We now define folding for the long l-code. As before, we want to start from an arbitrary function $A : ([u] \to \{-1,1\}^l) \to \{-1,1\}^l$ and define a new function $A' : ([u] \to \{-1,1\}^l) \to \{-1,1\}^l$ with the following properties:

- $A = A'$ if A is a long l-code;
- an oracle access into A' can be simulated with an oracle access into A;
- for a random f, $A'(f)$ is uniformly distributed in $\{-1,1\}^l$.

This can be achieved by setting, for example, $A'(f) = f(1)\cdot A(f(1)\cdot f)$.

The verifier expects proofs LlP and LlQ, where for each p $LlP(p)$ is supposed to be the long l-code of $P(p)$, and $LlQ(q)$ is supposed to be the long l-code of $Q(q)$, where P and Q are an acceptable pair of proof for Raz's verifier.

Our verifier starts by picking (p,q,π) as Raz's verifier, and it (almost but not quite) executes "in parallel" l instances of the "inner" test described above. Parallel repetition would demand that we select $3l$ functions $f_1,\ldots,f_l : [u] \to \{-1,1\}$, $g_1,\ldots,g_l : [w] \to \{-1,1\}$, $h_1,\ldots,h_l : [w] \to \{-1,1\}$ where each f_i and each g_i is uniform, and h_i are distributed as in Håstad's verifier. Instead, we select functions $\bar f : [k]\times[u] \to \{-1,1\}$, $\bar g : [k]\times[w] \to \{-1,1\}$ and $\bar h : [k]\times[w] \to \{-1,1\}$ with the distribution that Håstad's verifier would have used in selecting its queries if it were expecting to check long codes of elements of $[ku]$ and $[kw]$ (we are identifying $[k]\times[u]$ with $[ku]$ and $[k]\times[w]$ with $[kw]$). Then we define the functions $f_1,\ldots,f_l$ as $f_i(x) = \bar f(i,x)$, and similarly for g_i and h_i. This is the same as picking f_i and g_i independently and uniformly at random (that is, doing independently for each i what Håstad's verifier would have done expecting long codes of elements of $[u]$ and $[w]$), but it is slightly different than picking h_i independently as in Håstad's verifier.

This is one difference with respect to plain parallel repetition. The other difference is in the access mechanism to the proof implied by folding an l-long code.

We can also define a projection function $\bar\pi : [k]\times[w] \to [k]\times[u]$ such that $\bar\pi(i,b) = (i,\pi(a))$. It is possible to verify that if π satisfies the hashing condition, then also $\bar\pi$ satisfies the hashing condition.

Our test will accept if and only if

$$\forall i \in \{1,\ldots,l\}.(A'_i(\bar f) = -1 \vee B'(\bar g) = B'(\bar g(\bar f\circ\bar\pi\wedge h))$$

We then have that the acceptance probability of the test is

$$\mathop{\mathbf{E}}_{\bar A,\bar B,\bar\pi,\bar f,\bar g,\bar h}\prod_{i=1}^{l}\left(\frac{3}{4}+\frac{1}{4}\bar B'_i(\bar g)\bar B'_i(\bar g(\bar f\circ\bar\pi\wedge h))+\right.$$

$$\left.\frac{1}{4}\bar A'_i(\bar f)\bar B'_i(\bar g)\bar B'_i(\bar g(\bar f\circ\bar\pi\wedge h))\right)$$

which can be expanded as

$$\sum_{\alpha,\beta\subset[l],\alpha\cap\beta=\emptyset}\left(\frac{3}{4}\right)^{l-|\alpha|-|\beta|}\left(\frac{1}{4}\right)^{|\alpha|}\left(\frac{1}{4}\right)^{|\beta|}$$

$$\mathop{\mathbf{E}}_{\bar A,\bar B,\bar\pi,\bar f,\bar g,\bar h}\left[\prod_{i\in\alpha}\bar A'_i(\bar f)\prod_{j\in\alpha\cup\beta}\bar B'_j(\bar g)\bar B'_j(\bar g(\bar f\circ\bar\pi\wedge h))\right]$$

Suppose that our proofs LlP and LlQ are such that the test accepts with probability more than $(3/4)^l + 2^{2l}\delta$, then there must be some pair of sets α,β, at least one of them non-empty, such that

$$\left(\frac{1}{4}\right)^{|\alpha|+|\beta|}\mathop{\mathbf{E}}_{\bar A,\bar B,\bar\pi,\bar f,\bar g,\bar h}\left[\prod_{i\in\alpha}\bar A_i(\bar f)\right.$$

$$\left.\prod_{j\in\alpha\cup\beta}\bar B_j(\bar g)\bar B_j(\bar g(\bar f\circ\bar\pi\wedge h))\right] > \delta$$

Now, let us call A_α the random function that is obtained by sampling $\bar A$ and then defining $A_\alpha() = \prod_{i\in\alpha}A'_i()$ and let us call B_β the random function that is obtained by sampling $\bar B$ and then defining $B_\beta() = \prod_{i\in\alpha\cup\beta}B'_i$. If α is empty, then A_α is just identically 1, but otherwise we can verify that it is a folded function; likewise B_β is always a folded function. We can then rule out the case that α is empty, because we are in a position to invoke Lemma 5 and say

$$\mathop{\mathbf{E}}_{\bar B,\bar\pi,\bar f,\bar g,\bar h}\left[\prod_{j\in\beta}B_\beta(\bar g)B_\beta(\bar g(\bar f\circ\bar\pi\wedge h))\right] < 3\delta$$

We can then use Lemma 6 and see that there is a δ' such that

$$\mathbf{Pr}[\textsc{Decode}(A_\alpha) = \pi(\textsc{Decode}(B_\beta))] \geq \delta'$$

Now we can define a new procedure l-Decode that on input a function $A : ([l] \times [u] \to \{-1,1\}) \to \{-1,1\}$ first picks random subset $\alpha \subseteq [l]$, and then applies Decode to A_α. We then have

$$\mathbf{Pr}[l\text{-Decode}(\bar{A}) = \pi(l\text{-Decode}(\bar{B}))] \geq 2^{-2l}\delta'$$

This procedure can be used to reconstruct proofs P and Q (from LlP and LlQ) that Raz's verifier would accept with probability at least $2^{-2l}\delta'$. We omit a complete analysis of the composition step, but in a way similar to the analyses of [17, 15, 29, 14] one can prove that the soundness of our PCP system can be made arbitrarily close to $(3/4)^l$ by properly setting the parameters δ and u.

5. FROM PCP TO HITTING SET

As a first step, we modify our PCP system so that, when it makes its queries, each of the queries is uniformly distributed. We can use an approach described in [4]. First, we can make sure that each entry in the proof has the same probability of being queried by the verifier. This is achieved by defining a new proof format, where each entry of the proof in the original system is replicated a number of times proportional to the probability that it be queried. The new verifier simulates the old verifier and then, for each query, it chooses at random one of the places storing the answer to the query. Then, we modify the verifier so that it randomly permutes the order of the queries (this is possible since the verifier is non-adaptive). After this transformation, each of the three queries is uniformly distributed within the proof. [4]

We use the verifier of Theorem 4 with answer size l, and we assume that it has soundness at most $2 \cdot (3/4)^l$

Note that, for each fixed random string, our proof system has precisely 6^l accepting configurations.

The main gadget in the reduction is a set-system made of $m = 6^l$ triples of subsets $(A_1, B_1, C_1), \ldots, (A_m, B_m, C_m)$ of some fixed universe U, with the following properties:

- For each i, $A_i \cup B_i \cup C_i = U$;
- If we pick any sub-collection of these sets, but no three of them from the same triple, then their union is strictly smaller than U.

Such a gadget exists with $U = 2^{O(m)}$ (pick m times at random a random partition of U into three sets; if U is sufficiently large there is a positive probability that none of the 3^m possible ways of picking two sets from each triple is enough to cover U). For constant m, it can also be constructed in constant time.

Starting from the description of the computation of the PCP verifier of Theorem 4 on input a formula φ, we define an instance of Set Cover. If φ is satisfiable, then there will be a cover of a certain size; if the instance is unsatisfiable every cover will have size at least $\Omega((4/3)^{l/3})$ larger. For every random string r for the verifier, we define a copy U^r of the universe, and copies $(A_1^r, B_1^r, C_1^r), \ldots, (A_m^r, B_m^r, C_m^r)$ of the set system. The universe of the set cover instance that we produce is the union of all the sets U^r. The set system in the set cover instance has a set $S_{i,a,v}$ for each index $i \in \{1,2,3\}$ denoting one of the queries, for each entry x in the proof, and for each $v \in \{0,1\}^l$. The sets are defined as

$$S_{1,a,v} = \bigcup_{\substack{r,i\ :\ \text{on random string } r \text{ verifier}\\ \text{makes } a \text{ as first query, and answer}\\ v \text{ is consistent with } i\text{-th accepting}\\ \text{configuration}}} A_i^r$$

$$S_{2,b,v} = \bigcup_{\substack{r,i\ :\ \text{on random string } r \text{ verifier}\\ \text{makes } b \text{ as second query, and value}\\ v \text{ is consistent with } i\text{-th accepting}\\ \text{configuration}}} B_i^r$$

$$S_{3,c,v} = \bigcup_{\substack{r,i\ :\ \text{on random string } r \text{ verifier}\\ \text{makes } c \text{ as third query, and value}\\ v \text{ is consistent with } i\text{-th accepting}\\ \text{configuration}}} C_i^r$$

Note that each set A_i^r or B_i^r or C_i^r is included into exactly one set $S_{j,x,v}$. Each element of U^r belongs to exactly m sets in the gadget construction, so it follows that the reduction produces an instance where each element of the universe occurs into m sets.

It is standard, in reductions from PCP to Set Cover, to have a set for each position in the proof and for each value that that position can assume. It is also standard to have a gadget made of a collection of partitions of an universe U, with the property that one cannot cover U if not using all the elements of one of the partitions (in other reductions, that try to keep the size of U small, this condition is only satisfied by "small" subsets of the set system of the gadget). And it is also standard that one makes a copy of the gadget for each random string, and then creates a Set Cover instance where each set is the union of copies of sets of the gadget. The novelty in our reduction is that the partitions in the set system are associated with *satisfying assignments* for the acceptance predicate.

Let R be the total number of random strings, and M the length of the proof. If there is a proof P that is accepted with probability one, then one can select $3M$ sets and cover the entire universe. For every index j, and for every entry x in the proof, we select the set $S_{j,x,P(x)}$. This covers the entire universe, that is, it covers U^r for every r. In fact, consider an arbitrary random string r, and let (a,b,c) be the query made by the verifier. The answers $(P(a), P(b), P(c))$ are accepted by the verifier, and constitute, say, its i-th accepting configuration for string r. This means that A_i^r is part of $S_{1,a,P(a)}$, and, as such, part of our solution, and likewise B_i^r and C_i^r are part of our solution, and so U^r is covered.

Now, suppose there is a set cover that uses at most αM sets. Consider the following experiment: pick at random a query triple (a,b,c) as done by the verifier. Then count the number of sets of the form $S_{1,a,v}$, $S_{2,b,v'}$, and $S_{3,c,v''}$ in the solution, and consider the average of this total. Since a, b and c are uniformly distributed in the proof, this average has to be at most 3α. Then, for at least half of the random strings, the total is at most 6α. We call such random choices the "good" ones.

Consider the following randomized proof construction. For each query x, pick at random $j \in \{1,2,3\}$, and pick

[4] Of course the distributions are correlated.

at random a v such that $S_{i,x,v}$ belongs to the solution. If no such set belongs to the solution, pick uniformly at random among all possible answers. Consider the behavior of the verifier when it picks a good random choice r, making queries (a, b, c). The solution covers all the elements in U^r, so in particular there has to be an i such that A_i^r, B_i^r and C_i^r are all part of the solution. Let the i-th accepting configuration be such that its answers are (q_1, q_2, q_3). Then A_i^r can be part of the solution only if S_{1,a,q_1} is part of the solution, B_i^r can be part of the solution only if S_{2,b,q_2} is part of the solution, and similarly S_{3,c,q_3} has to be part of the solution. Let now k_1, k_2, k_3 be, respectively, the number of sets in the solution of the form $S_{1,a,v}, S_{2,b,v'}, S_{3,b,v''}$. We know $k_1 + k_2 + k_3 \leq 6\alpha$, and we have just proved that the probability that the randomized proof has q_1, q_2, q_3 as answers to a, b, c is at least $1/3k_1 \cdot 3k_2 \cdot 3k_3$ which is at least $1/(18\alpha)^3$.

If our proof system has soundness $2 \cdot (3/4)^l$ and if $(3/4)^l < 1/(2 \cdot (18\alpha)^3)$, then this acceptance probability is too high if we started from an unsatisfiable φ.

We then proved a hardness of approximation to within a $\Omega((4/3)^{k/3})$ factor, for instances where every element occurs in at most 6^k sets. If we call $B = 6^k$ the hardness factor is $\Omega(B^{(\log_6(4/3))/3}) = \Omega(B^{.0535\cdots})$ where $.0535 > 1/19$.

6. INDEPENDENT SET

The starting point for our hardness of approximation result is the following result.

LEMMA 7 ([28]). *For every $\epsilon > 0$ and every $k > 0$ there is a PCP characterization of NP where the verifier has soundness $1 - \epsilon$, completeness $2^{-k^2} + \epsilon$ and makes $2k + k^2$ queries. Furthermore, for each random string, the acceptance predicate has 2^{2k} satisfying assignment, and is a linear function.*

In the following, we will use $\epsilon = 2^{-k^2}$. We will also use $1/2$ as a crude lower bound to the completeness.

In the FGLSS [12] reduction from PCP to Independent Set, a fundamental notion is that of an *accepting configuration.* An *configuration* is the specification of a random string used by the verifier, together with a specification of the values of the bits of the proof read by the verifier when using such a random string; a configuration is accepting if the verifier accepts when using that particular random string and getting those answers to its queries.

It will be useful to note that in the verifier of [28], for each random string, and for each bit read according to that random string, the number of accepting configurations where the bit is zero equals the number of accepting configurations where the bit is one.

Let r be the number of random bits used by the above verifier. The FGLSS reduction applied to the verifier of Lemma 7 creates a graph with 2^{r+2k} vertices, one for each accepting configuration; two vertices are connected by an edge if and only if the configurations are inconsistent (i.e. the configurations read the same bit and find different values). If there is a proof that the verifier accepts with probability at least $1/2$ then the FGLSS graph has an independent set of size at least $(1/2) \cdot 2^r$; if no proof is accepted with probability greater than $2 \cdot 2^{-k^2}$, then the FGLSS graph has no independent set of size greater than $2 \cdot 2^{-k^2} \cdot 2^r$.

In the FGLSS graph, for every index i in the proof, call O_i the set of vertices where index i is queried, and the answer is one, and call Z_i the set of vertices where index i is queries and the answer is zero. The FGLSS graph contains all possible edges between vertices in O_i and vertices in Z_i, for each i. It is easy to convince oneself that the union of these edge sets is in fact precisely the set of edges of the FGLSS graph. As said before, for each i we have $|O_i| = |Z_i|$. Let us call $n_i = |O_i|$.

The main idea in [10] is to replace each of this bipartite complete subgraphs with a bounded-degree expander. In particular we can use the following result

LEMMA 8 ([27]). *For every δ and sufficiently large n, there is a bipartite graph $([n], [n], E)$ of degree $O((1/\delta)\text{poly}\log(1/\delta))$ such that for each subsets $A, B \subseteq [n]$, $|A| \geq \delta n$, $|B| \geq \delta n$ there is at least an edge $(i, j) \in E$ such that $i \in A$ and $j \in B$. Furthermore the graph can be constructed in time* $\text{poly}(n, 2^{1/\delta})$.

Starting from the PCP system, we create a sub-graph of the FGLSS graph where for every i, we include edges between O_i and Z_i only as prescribed by the above expander.[5] As in [10] we can argue that the maximum independent set in the new graph can only be bigger, since we have less edges, but that every independent set in the new graph can be transformed into an independent set for the old graph by adding at most $q\delta N$ vertices. This is true because, for every i, the independent set can contain at most δn_i elements of O_i, or at most δn_i elements of Z_i. Therefore, we can remove no more than δn_i vertices from the independent set, and make sure that, at least restricted to the vertices in $O_i \cup Z_i$ we also have an independent set in the original FGLSS graph. We can repeat this process for each i, thus getting an independent set for the FGLSS graph, and removing no more than $\sum_i \delta n_i$ vertices. Since each configuration can belong to at most q sets of the form O_i or Z_i, we have $\sum_i n_i \leq qN$.

Therefore it is hard to decide whether the new graph has an independent set of size $(1/2) \cdot 2^r$ or whether all independent sets have size less than $2 \cdot 2^{-k^2} \cdot 2^r + q\delta N$, where the second expression is less than $3 \cdot 2^{-k^2} \cdot 2^r$ if we fix $\delta = 2^{-k^2} 2^r / qN = 2^{-k^2-2k}/q$. For such choice of δ, the degree of the graph becomes $2^{k^2+2k}\text{poly}k$, and the hardness of approximation is $\Omega(2^{k^2})$. If we call B the degree of the graph, then the hardness of approximation takes the form $B/2^{O(\sqrt{\log B})}\text{poly}\log B = B/2^{O(\sqrt{\log B})}$.

Acknowledgements

I thank Reuven Bar-Yehuda for raising (in 1997, at a Dagstuhl meeting on approximation algorithms) the issue of the hardness of Hitting Set for sets of bounded size. Sanjeev Arora set me on the right track by suggesting that the reduction in [11] already proves strong bounds for Set Cover and Hitting Set for sets of bounded size. Johan Håstad introduced me to the question of the approximability of Max SAT on instances with bounded occurrences, and gave comments that simplified the presentation of my reduction. Part of this work was done while the author was at Columbia University.

[5]There could be a difficulty if O_i and Z_i are not large enough. However one can always get more configurations by letting the verifier toss additional random bits, and then not using them in its computation.

7. REFERENCES

[1] N. Alon, U. Feige, A. Wigderson, and D. Zuckerman. Derandomized graph products. *Computational Complexity*, 5(1):60–75, 1995.

[2] N. Alon and N. Kahale. Approximating the independence number via the θ function. *Mathematical Programming*, 80:253–264, 1998.

[3] N. Alon and J. Spencer. *The Probabilistic Method*. Wiley Interscience, 1992.

[4] S. Arora. *Probabilistic Checking of Proofs and Hardness of Approximation Problems*. PhD thesis, University of California at Berkeley, 1994.

[5] S. Arora and C. Lund. Hardness of approximations. In *Approximation Algorithms for NP-hard Problems*. PWS Publishing, 1996.

[6] M. Bellare, O. Goldreich, and M. Sudan. Free bits, PCP's and non-approximability – towards tight results. *SIAM Journal on Computing*, 27(3):804–915, 1998. Preliminary version in *Proc. of FOCS'95*.

[7] M Bellare, S. Goldwasser, C. Lund, and A. Russell. Efficient probabilistically checkable proofs and applications to approximation. In *Proceedings of the 25th ACM Symposium on Theory of Computing*, pages 294–304, 1993. See also the errata sheet in *Proc of STOC'94*.

[8] P. Berman and M. Karpinski. On some tighter inapproximability results. Technical Report TR98-65, Electronic Colloquium on Computational Complexity, 1998.

[9] V. Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4:233–235, 1979.

[10] A.E.F. Clementi and L. Trevisan. Improved non-approximability results for vertex cover with density constraints. *Theoretical Computer Science*, 225:113–128, 1999.

[11] U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

[12] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM*, 43(2):268–292, 1996. Preliminary version in *Proc. of FOCS91*.

[13] O. Goldreich. *Modern Cryptography, Probabilistic Proofs and Pseudorandomness*. Springer-Verlag, 1999.

[14] V. Guruswami. Query-efficient checking of proofs and improved PCP characterizations of NP. Master Thesis, MIT Laboratory for Computer Science, 1999.

[15] V. Guruswami, D. Lewin, M. Sudan, and L. Trevisan. A tight characterization of NP with 3 query PCPs. In *Proceedings of the 39th IEEE Symposium on Foundations of Computer Science*, pages 8–17, 1998.

[16] M. Halldorsson. A survey on independent set approximations. In *APPROX'98*, pages 1–14, 1998. LNCS 1444, Springer-Verlag.

[17] J. Håstad. Some optimal inapproximability results. In *Proceedings of the 29th ACM Symposium on Theory of Computing*, pages 1–10, 1997.

[18] J. Håstad. On bounded occurrence constraint satisfaction. *Information Processing Letters*, 74(1):1–6, 2000.

[19] D. Hochbaum. Approximation algorithms for set covering and vertex cover problems. *SIAM Journal on Computing*, 11:555–556, 1982.

[20] D.S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9:256–278, 1974.

[21] D. Karger, R. Motwani, and M. Sudan. Approximate graph coloring by semi-definite programming. *Journal of the ACM*, 45(2):246–265, 1998.

[22] F.T. Leighton. *Introduction to Parallel Algorithms and Architectures*. Morgan Kaufmann, 1992.

[23] L. Lovasz. On the ratio of optimal integral and fractional covers. *Discrete Mathematics*, 13:383–390, 1975.

[24] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM*, 41:960–981, 1994. Preliminary version in *Proc. of STOC'93*.

[25] C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43:425–440, 1991. Preliminary version in *Proc. of STOC'88*.

[26] R. Raz. A parallel repetition theorem. *SIAM Journal on Computing*, 27(3):763–803, 1998. Preliminary version in *Proc. of STOC'95*.

[27] O. Reingold, S. Vadhan, and A. Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors. In *Proceedings of the 41st IEEE Symposium on Foundations of Computer Science*, 2000.

[28] A. Samorodnitsky and L. Trevisan. A PCP characterization of NP with optimal amortized query complexity. In *Proceedings of the 32nd ACM Symposium on Theory of Computing*, 2000.

[29] M. Sudan and L. Trevisan. Probabilistically checkable proofs with low amortized query complexity. In *Proceedings of the 39th IEEE Symposium on Foundations of Computer Science*, 1998.

[30] S. Vishwanathan. Personal communication to M. Halldorsson. Cited in [16], 1996.

APPENDIX

We state two versions of the Chernoff bound that are used in the body of the paper.

A more general statement (with a proof) of the following result is in [22, Lemma 1.7].

LEMMA 9. *Let $X_1, \ldots, X_m$ be independent random variables with range $\{0,1\}$, and let $k \geq e^2\, \mathbf{E}[\sum_i X_i]$. Then*

$$\mathbf{Pr}\left[\sum_i X_i \geq k\right] \leq e^{-k}$$

A more general statement of the following result is in [13, Page 109]. Proofs of more general statements can be found in [3].

LEMMA 10. *Let $X_1, \ldots, X_m$ be independent, identically distributed, random variables with range $\{0,1\}$, and let $0 < \epsilon < 1/4$. Then $\mathbf{Pr}\left[\sum_i X_i - \mathbf{E}[\sum_i X_i] \geq \epsilon m\right] \leq e^{-2\epsilon^2 m}$*