Probabilistically Checkable Proofs with Low Amortized Query Complexity[*]

[Preliminary Version]

Madhu Sudan[†]   Luca Trevisan[†]

July 24, 1998

Abstract

The error probability of Probabilistically Checkable Proof (PCP) systems can be made exponentially small in the number of queries by using sequential repetition. In this paper we are interested in determining the *precise rate* at which the error goes down in an optimal protocol, and we make substantial progress toward a tight resolution of this question.

A PCP verifier uses $\bar{q}$ *amortized query bits* if, for some t, it makes $\bar{q}t$ queries and has error probability at most 2^{-t}. A PCP characterization of NP using 2.5 amortized query bits is known [25], and, unless P=NP, no such characterization is possible using 1 amortized query bits [7]. We present a PCP characterization of NP that uses roughly 1.5 amortized query bits. Our result has two main implications.

Separating PCP from 2-Provers 1-Round. In the 2-Provers 1-Round (2P1R) model the verifier has access to two oracles (or provers) and can make one query to each oracle. Each answer is a string of l bits (l is called the *answer size*). A 2P1R protocol with answer size l can be simulated by a PCP that reads $2l$ bits; we show that the converse does not hold for $l \geq 7$, unless P=NP. No such separation was known before.

The Max kCSP problem. The Boolean constraint satisfaction problem with constraints involving at most k variables, usually called Max kCSP, is known to be hard to approximate within a factor $2^{-.4k}$ [25], and a $2 \cdot 2^{-k}$-approximation algorithm is also known [24]. We prove that Max kCSP is NP-hard to approximate within a factor of roughly $2^{-2k/3}$.

[*] An extended abstract of this work will appear in the Proceedings of FOCS'98.

[†] `{madhu,luca}@theory.lcs.mit.edu`. MIT Laboratory for Computer Science. 545 Technology Square, Cambridge MA 02139.

ISSN 1433-8092

1 Introduction

PCP characterizations of NP [6, 5, 12, 4, 3, 8, 13, 9, 7, 15, 16, 25] are the best known tool to prove results about the hardness of approximation of combinatorial optimization problems. Progress in this area has been driven by the goal of characterizing NP via increasingly *more efficient* PCP verifiers, under various formalizations of the notion of efficiency, and we now stand on a point where PCP constructions are known that are *optimal* with respect to some trade-off of these parameters.

The most important efficiency parameters in PCP constructions are the *number of queries* that the verifier asks to the oracle proof and the *soundness.* The soundness of a verifier is the probability that it accepts a "proof" of a wrong statement, that is, the error probability in the case of "no-instances". The verifier may make errors also in the case of yes-instances, i.e. it may reject the valid proof of a correct statement. In this paper we will restrict ourselves to protocols that accept valid proofs with probability at least $1 - \varepsilon$, where $\varepsilon > 0$ is a constant that can be made arbitrarily small independently of the other parameters of interest (*almost-perfect completeness*), therefore whenever we will use the term "error" this should always be interpreted as "soundness".

One direction of research is to concentrate on protocols using a minimal amount of queries (i.e. 3) and then reduce the soundness as much as possible. An optimal protocol of this kind has been constructed by Håstad in [16], where he describes a verifier that makes 3 queries, has almost perfect completeness, and soundness 1/2.

A somewhat orthogonal line of research is to fix some small error probability and ask what is the minimal number of queries that suffice to characterize NP with a PCP protocol having that error. Iterating Håstad's protocol t times we get a PCP system that asks $3t$ queries, has almost perfect completeness and soundness 2^{-t}. Is it possible to achieve error 2^{-t} with significantly less than $3t$ queries? This question has some interesting applications, that will be described later. For starters, we can observe that at least t queries are necessary (see [7]); therefore the optimal protocol will have query complexity $\bar{q}t$ for some $1 < \bar{q} \leq 3$. $\bar{q}$ is the *amortized query complexity* of the PCP.

One possible approach to creating PCPs with low amortized query complexity is to iterate a basic protocol several times, while recycling queries between various iterations. This approach is similar to the approaches used to reduce error in PCPs when measuring other resources: in particular, randomness and "free bits". In the former case, the methods used for recycling randomness while reducing error in general probabilistic computation [1, 17] turns out to be quite useful and are used, for instance, in [3, 27, 2]. In the latter case (minimizing "free bits") also, the notion of recycling can be analyzed and the works of [9, 7, 14, 15] show that this method leads to significant benefits. Our task, however differs from the previous cases in some critical aspects. For instance, in the context of recycling randomness, a random bit is counted as a "recycled" bit, if it is obtained by applying some (arbitrary) function to the previously used random bits. In contrast, while recycling queries, the recycled query has to be identical to a previously issued query. The contrast between recycling free bits and query bits is exhibited by the following example: In the case of free bits, the known analyses yield protocols in which the error decreases as a polynomial in the number of iterations (and this suffices!); in the case of recycling queries, the error of the protocol needs to go down exponentially in the number of iterations.

Despite these difficulties, the idea of repeating a basic protocol several times with recycling of queries has been pursued by Trevisan [25] with some success. His analysis yields query-efficient Linearity Tests and PCP constructions. The PCP verifier of [25] has error 1/4 and makes 5 queries (therefore, the amortized query complexity is 2.5.) The verifier repeats twice the 3-query verifier of Håstad [16]; one query is recycled between the two executions. Other, possibly more efficient, recycling schemes were also described in [25] and one of them was analyzed for the simpler problem of Linearity Testing, resulting in a linearity tester having amortized query complexity 1.5.

The latter result mentioned above, however, does not immediately translate to the context of PCP constructions. (An intrinsic reason for this is given by a lower bound of Bellare et al. [7] — this is discussed further in the next section.) In this paper we abstract a new "proof-composition" technique based on the recent work of Håstad in [15]. We then create a new verifier to use with the composition technique by modifying the verifier of Trevisan [25]. The result is a family of PCP verifiers that, for any k, make $3k+2$ queries "non-adaptively" and have error 2^{-2k}. The term non-adaptive implies that the queries are chosen purely as a function of the input and the random coins and are not a function of answers to previous queries. This aspect is needed for one the applications given below. In order to state our main result more formally, recall that a probabilistically checkable proof system is described by an (r, q)-restricted non-adaptive PCP verifier, i.e., a probabilistic polynomial time oracle machine, who on input x, tosses $r(|x|)$ random coins and makes $q(|x|)$ non-adaptive queries to a proof oracle P. A language $L \in \mathrm{naPCP}_{c,s}[r, q]$ ("na" stands for non-adaptive queries) if there exists an (r, q)-restricted non-adaptive verifier V satisfying: (1) *(completeness)* If $x \in L$, then $\exists$ P s.t. $\Pr_R[V^P(x : R)$ accepts $] \geq c$. (2) *(soundness)* If $x \notin L$, then $\forall$ P $\Pr_R[V^P(x : R)$ accepts $] \leq s$ (where $V^P(x; R)$ denotes the computation of V on input x and random string R with oracle P). Our PCP construction proves the following theorem.

Theorem 1 (Main) *For every $\varepsilon > 0$ and positive integer k,* $\mathrm{NP} = \mathrm{naPCP}_{1-\varepsilon, 2^{-2k}}[\log, 3k+2]$.

The amortized query complexity of our family of protocols tends to 1.5. The two main consequences of our result are described below.

PCP VERSUS 2-PROVERS 1-ROUND. In a 2-Provers 1-Round (2P1R) protocol the verifier has access to two oracles (or provers) P and Q representing a membership proof of an NP statement. The verifier is allowed to make only one query to each oracle; upon being queried, the oracle answers with a string of l bits (l is said to be the *answer size* of the protocol.) The query complexity of such a 2P1R proof system is defined to be $2l$. The completeness and soundness of this proof system are defined in the usual way and thus the notion of amortized query complexity also extends naturally. (The amortized query complexity is $\bar{q}$ if the query complexity is $\bar{q}k$ and the soundness error is 2^{-k}.)

It is clear that a 2P1R protocol can be simulated by a PCP system with no larger query complexity, but the 2P1R seems to be a weaker model. In particular, it is non-trivial to show that the error of 2P1R proof systems can be reduced by increasing answer sizes, while an analogous result is quite straightforward for PCPs. The former question was a subject of significant attention in the past and was finally resolved by Raz [21]; and an ensuing 2P1R protocol for NP turns out to be a critical ingredient in many efficient constructions of PCPs (including ours). Despite this significant difference in behavior and utility of 2P1R proof systems and PCPs, no separation between the two was known. The only limitation known on the power of 2P1R proof systems is due to Serna et al. [22], who show a lower bound of 2 on the amortized query complexity of any 2P1R proof system recognizing an NP-complete language. The results of [22] are even stronger since they imply that even a PCP that can query two entries from a non-Boolean proof (each entry being a string of l bits) can only recognize languages in P as long as the error is less than 2^{-l}.

By constructing a PCP verifier for NP with amortized query complexity of 1.5, we derive a separation between PCPs and 2P1R. In fact the separation actually holds for any answer size $l \geq 7$. In the full version of this paper, we describe a generalization of the 2P1R model and of the 2-query non-Boolean PCP model. In this model the verifier accesses a binary table but can only read two "blocks" of l consecutive bits. (This model was proposed to us by Shafi Goldwasser and Amit Sahai.) We extend the result of Serna et al. to this model as well, thereby concluding that the separation between PCPs and 2P1R is due to the "locality" of the access mechanism of the latter model.

APPLICATIONS TO THE MAX kCSP PROBLEM. For an integer $k > 1$, the Max kCSP is the Boolean constraint satisfaction problem with constraints involving at most k variables (see [18, 10, 24, 23, 26, 19, 28, 29].) Max kCSP was known to be hard to approximate within $2^{-.4k}$ [25], our result implies that it is hard to approximate within a factor of roughly $2^{-2k/3}$. The best known algorithm has an approximation ratio $2^{-(k-1)}$ [24] (note that a random solution is 2^{-k}-approximate.)

2 Techniques: Previous Works and Our Contribution

In this section we review the main technical ideas from previous papers and we explain our contribution.

2.1 The standard proof composition paradigm and limitations

All the recent constructions of Probabilistically Checkable Proofs rely on the *proof-composition* methodology, invented by Arora and Safra [4]. The main idea is to construct two proof systems, that optimize different efficiency parameters, and then combine them together in order to build a composed system that is efficient under all the parameters. Composition is done between an "outer verifier" V^{out} that is typically a 2-Provers 1-Round (2P1R) protocol[1] and an "inner" verifier V^{in}. The verifier of the composed system V^{comp} expects a proof that be the entry-wise encoding of the proof of V^{out} using an error correcting code. V^{comp} simulates the behavior of V^{out}, chooses two entries of the proof, and then calls as a "subroutine" V^{in} to determine whether the encoding of these entries "look like" being encodings of something that V^{out} would have accepted. Therefore the properties of V^{in} are the following: it knows the acceptance predicate of V^{out}, and it has oracle access to two strings that are allegedly encodings of answers that V^{out} would have accepted. V^{in} tests whether this is the case. An inner verifier with, say, almost perfect completeness and soundness $1/2$, has the following properties:

- Whenever the conditions are satisfied, then V^{in} accepts with probability $1 - \varepsilon$;
- Whenever V^{in} accepts with probability $> 1/2$, the strings it is accessing are "close" to being correct encodings of consistent answers.

It is not immediate to come up with a usable formalization of the second property. One way is to define a decoding procedure that given a string, that is an alleged encoding of an answer, returns a possible answer for the 2P1R protocol. V^{in} satisfies the second condition if whenever it accepts a pair of strings with probability $> 1/2$ the decoding procedure, applied independently to the two strings, will produce consistent answers. This is still a bit too much restrictive. In the most useful formulation, the decoding procedures are randomized and the guarantee is that if V^{in} accepts with probability greater than $1/2 + \delta$ then the decoding procedures produce a consistent pair of strings with probability at least δ', where δ' depends only on δ. Such a decoding procedure is implicit in the work of Bellare et al. [8]. In what follows, a recursive composition scheme using a randomized decoding procedure and a 2P1R outer verifier will be called the *canonical composition methodology.* Observe that the definition of inner verifier depends on the *error correcting code* and the *decoding procedure* that is being used. An inner verifier has to test both that the two strings are valid

[1] The 2-Prover 1-Round construction of Raz [21] is currently the standard one for this application.

codewords (or, at least, "close" to being valid codewords) of the error-correcting code being used (*codeword test*) *and* that the decodings of the strings are likely to be consistent answers of the outer verifier (*consistency test*). Each of this tasks gives rise to different difficulties and limitations.

EFFICIENT CODEWORD TEST. Much of the recent progress in designing inner verifiers and so PCP constructions came from improved ways of testing whether the given strings are correct codeword of the used error correcting code. The current standard for the error correcting code is the Long code introduced by Bellare, Goldreich and Sudan [7]. The encoding with the Long code of a message $a \in \{0,1\}^n$ is the evaluation of $f(a)$ for any $f : \{0,1\}^n \to \{0,1\}$. Therefore the length of the Long code of a is 2^{2^n}. The best known methodology to analyze codeword tests for the Long code uses Fourier analysis on $\{0,1\}^n$. This technique was introduced and applied with great success by Håstad in his recent works on PCP constructions [14, 16]. Trevisan [25] uses this techniques to show how to test the Hadamard code and the Long code with roughly 1.5 amortized query bit. These constructions could not be extended to a full PCP constructions, due to an inherent bottleneck in the consistency test that holds for any inner verifier in the canonical composition methodology, that we describe next.

EFFICIENT CONSISTENCY TEST. It is not hard to see that proof systems designed with the canonical composition methodology cannot achieve an amortized query complexity better than 2. Let A and B be the two strings given in input to the inner verifier, q_A and q_B be the number of queries that the verifier asks to A and B respectively, and $q = q_A + q_B$ be the total query complexity. If A and B are random Long codes, and the verifier has perfect or almost perfect completeness, then there is a probability $2^{-\min\{q_A,q_B\}} \geq 2^{-q/2}$ that the verifier accepts (we will have to subtract a factor ε for the case of almost perfect completeness.) A slightly more involved argument shows that 1 amortized free bit is also a lower bound for the canonical composition methodology (see [7].) Free bits are an efficiency parameter of PCPs that is motivated by applications to proving hardness of approximation for the Max Clique problem. The number of free bits of a PCP system is always no more than the number of query bits. In systems designed to optimize free bits, the number of queries can be doubly exponentially larger than the number of free bits, or more. Bellare et al. [7] have shown that a protocol with a certain number $\bar{q}$ of amortized query bits can always be transformed into another that has $\bar{q} - 1$ (average) amortized free bits, so the lower bound of one amortized free bit implies the lower bound of two amortized query bits.

2.2 Overcoming the limitations: A new composition theorem

The free bit lower bound has been overcome in a recent work by Håstad [15], where for any $\varepsilon > 0$ he describes a construction that uses ε amortized free bits. To avoid the lower bound, Håstad considers an inner verifier that looks at tables A and $B_1, \ldots, B_k$, where each pair (A, B_i) would have been a possible input for an inner verifier in the canonical composition methodology. The advantage of working with several tables is that the decoding of A can now be done as a function of $B_1, \ldots, B_k$, and so the argument showing that 2 amortized query bits are a lower bound does not hold any more. Håstad does not present his result with respect to an explicit composition theorem and, in his proof, it is hard to distinguish the protocol-specific difficulties from the ones related to the general idea of using several tables. One of the main contributions of our work is to extract the explicit composition theorem from the work of [15] and then adapt it to our purpose. The novel ingredient in this composition theorem is a new definition of an outer verifier that makes several queries, specifically $k+1$ where the parameter k is a positive integer. A verifier with similar soundness conditions was used to avoid a related lower bound (but not for use in composition of proofs) in the work of Feige on Set Cover [11]. The composition theorem composes such an outer verifier with inner verifiers that look at several tables $A, B_1, \ldots, B_k$. The composition theorem and the associated properties of the inner verifier are described in Section 3.

The composition theorem reduces the task of constructing an efficient PCP verifier (with respect to amortized query complexity) to the task of constructing the appropriate inner verifier. At this point we need to deviate from the work of Håstad, as argued next: The "inner verifier" of Håstad first reads a certain number of (free) bits l from A, and then applies a codeword test on each B_i, using l/k free bits in each codeword test. A separate analysis shows that each codeword test has error probability at most $p = p_{k,l}$, and then a union bound establishes that the probability that one or more tests fail is at most kp. The final soundness will be a little more than this bound. The bad news of this method is that the free bit complexity of the composed verifier is $2k$ times greater than the free bit complexity of the codeword test (each codeword test uses l/k free bit, and the total number of free bits is $2l$, including the bits read in A) and the error is *worse* than the error of the codeword test. However the *amortized* free bit complexity of the codeword test can be made arbitrarily small, and despite the increase that happens during the composition, the final amortized free bit complexity can still be made arbitrarily small. In our case, however we can not afford such luxuries. Since a codeword test must use at least one amortized query bit, the multiplicative factors involved in the composition can not be hidden any more, and the composition scheme of Håstad would blow the amortized query complexity out of control.

The second part of our work is thus a new inner verifier that is obtained by iterating k times (with recycling) a 5 query protocol of [25] (which is, in turn, a 2-fold iteration, with recycling, of the 3-query protocol of [16]). The novelty in this verifier is that in each iteration it uses a different B-table, while recycling the queries made on the A-table. In contrast, the basic protocol of [25] would expect two tables A and B and would read 2 bits in A and 3 bits in B; therefore our iterated protocol reads $3k + 2$ bits. A tight analysis shows that the soundness of the iterated protocol is 2^{-2k}. We stress the following point of difference from [15]: As in [15] we do k tests, one for each B_i; each test has individually soundness $p = 1/4$, however our analysis of the soundness of the iterated verifier does not give an error kp, but rather p^k, that is, the error goes down exponentially in k, instead of growing with k as in [15]. Details of this inner verifier are given in Section 5.

OPEN QUESTIONS. The eventual goal in this line of work is to find, for any $\varepsilon > 0$, a PCP characterization of NP where the verifier has amortized query complexity $1 + \varepsilon$. Since this result would also imply a characterization of NP with ε amortized free bits for any $\varepsilon > 0$, and since only a very complicated proof is known of this latter result [14, 15], we do not expect this goal to be easy to achieve. Towards this goal, it would be interesting to first find a codeword test having amortized query complexity $1 + \varepsilon$. Tests are presented in [25] which are conjectured to have such efficiency. As discussed in [25], the Fourier analysis of such protocols cannot prove an amortized query complexity better than 1.5 unless the proof is somehow "specialized" on the Fourier spectrum of Boolean functions.[2] Progress in this direction promises to have exciting mathematical content. Once a codeword test with a better Fourier analysis will be known, the techniques of the present paper (the way of splitting queries between tables, and our composition theorem) should suffice to extend the result to a full PCP construction.

[2] Current proof techniques use properties of the Fourier spectrum of Boolean functions which are shared by all the functions of unit ℓ_2 norm; one can show that techniques of this kind do not suffice to go below 1.5 amortized query bits.

3 Our New Composition Scheme

In this section we introduce our new definition of outer verifier, an appropriate corresponding notion of inner verifier, and describe the composition theorem. (The actual construction of the outer verifier, the inner verifier and the proof of the composition theorems are deferred to later sections.)

As mentioned earlier all known constructions use the verifier of Raz [21] as the outer verifier. We will also use it in order to derive our new outer verifier.

Recall that the verifier of Raz works in the following way: it generates, according to a certain distribution, a triple (p, q, π) where p is a query to the oracle P, q is a query to the oracle Q and π is a function mapping from the domain of answers of Q to the domain of answers of P. The verifier asks query p to oracle P, receiving a certain answer a, and then asks query q to oracle Q, receiving answer b, and it accepts iff $\pi(b) = a$. When the canonical composition method is used, the composed verifier expects a proof that be the entry-wise Long code of all the answers of P and Q. The composed verifier generates a triple (p, q, π) according to the same distribution of Raz's verifier, will look at the tables A and B, being the encoding of the answers to p and q respectively, and will execute the inner verification procedure on them. Thus, the inner verification procedure is given π and has access to A and B and the task is to determine whether B is the Long code of some b and A is the Long code of some a such that $\pi(b) = a$.[3]

At the same abstract level, our outer verifier generates k triples $(p, q_i, \pi_1), \ldots, (p, q_k, \pi_k)$, where p is a random query to P according to the distribution of Raz's verifier and (p, q_i, π_i) are sampled on the marginal distribution of the triples of Raz's verifier given that the first entry is p. The verifier queries p to P receiving a as an answer, and queries $q_1, \ldots, q_k$ to Q receiving $b_1, \ldots, b_k$ as answers. We say that the verifier *strongly accepts* if $a = \pi_1(b_1) = \cdots \pi_k(b_k)$, we say that it *weakly accepts* when at least two of the values $a, \pi_1(b_1), \ldots, \pi_k(b_k)$ are the same, and we say that it *rejects* when the values $a, \pi_1(b_1), \ldots, \pi_k(b_k)$ are all different. On input a valid statement and a correct proof, our verifier strongly accepts with probability one. On input an invalid statement, and for every pair of proofs, our verifier rejects with high probability. The composed verifier looks at the table A that is the encoding of the answer to p and to $B_1, \ldots, B_k$, that are respectively the encodings of the answers to $q_1, \ldots, q_k$, and then executes the inner verification procedure. Therefore an inner verifier with completeness c and soundness s has to accept with probability at least c encodings of answers that would make the outer verifier strongly accept, and if the inner verifier accepts with probability $s + \delta$ its proofs, then a decoding procedure should produce decodings that make the outer verifier at least weakly accept with probability at least δ', where δ' depends only on δ.

Before formalizing the above discussion we need to introduce some notation in order to specify the encoding scheme used. From now on Boolean functions will be defined with values in $\{1, -1\}$ rather than $\{0, 1\}$. The association is that -1 stands for 1 (or true) and 1 stands for 0 (or false). Observe that multiplication in $\{1, -1\}$ acts as Boolean xor in $\{0, 1\}$. For an integer k, we denote by $[k]$ the set $\{1, \ldots, k\}$. For two sets α and β we denote by $\alpha \Delta \beta = (\alpha \cup \beta) - (\alpha \cap \beta)$ their symmetric difference. Recall that Δ is commutative and associative.

For an integer n, we denote by $\mathcal{F}_n$ the set of functions $f : [n] \to \{1, -1\}$. The operator $\circ$ denotes composition of functions, i.e. if $f \in \mathcal{F}_n$ and $\pi : [m] \to [n]$ then the function $f \circ \pi \in \mathcal{F}_m$ is defined as $(f \circ \pi)(b) = f(\pi(b))$ for any $b \in [m]$.

We say that a function $A : \mathcal{F}_n \to \{1, -1\}$ is *linear* iff $A(f)A(g) = A(fg)$ for all $f, g \in \mathcal{F}_n$. There

[3] Normally, the acceptance condition of Raz's verifier is described as "$\pi(b) = a$ and $h(b) = 1$", where h is a boolean function generated by the verifier together with p, q, π. Following [25], we avoid this additional complication by encoding π into h.

are 2^n linear functions. There is a linear function l_α for any set $\alpha \subseteq \{1,-1\}^n$; it is defined as

$$l_\alpha(f) = \prod_{a\in\alpha} f(a) \; .$$

By convention, we say that a product ranging over the empty set equals 1. The Long code is the set of linear functions whose support is a singleton, i.e. $\mathsf{LONG}_n = \{l_{\{a\}} : a \in [n]\}$. We say that $l_{\{a\}}$ is the Long code of a. Thus, the Long code is formed by n codewords of length 2^n. This definition is equivalent to the definition mentioned earlier in Section 2, but will be more convenient in our analysis.

Finally, we need a notion analogous to that of *folding* from [7]. Observe that if $A = l_{\{a\}}$ is a codeword of the Long code, then $A(f) = f(a) = -(-f(a)) = -A(-f)$ for any f; for any function $A : \mathcal{F}_n \to \{1,-1\}$ we will define a new function A' that satisfies such a property. The definition of A' is as follows:

$$A'(f) = \begin{cases} A(f) & \text{If } f(1) = 1 \\ -A(-f) & \text{If } f(1) = -1. \end{cases}$$

We stress that, for any f, $A'(f)$ can be evaluated with one query to A, moreover A' is equal to A if A is a codeword of the Long code.

We are now ready to define our outer and inner verifier and the composition theorem.

Definition 2 (k-Outer Verifier) *A* k-outer verifier *for a language L with soundness c and completeness s, and answer size l is a randomized polynomial time oracle algorithm V that is given oracle access to two oracles P and Q with the properties that for every input string x,*

- [Efficiency] *each oracle answers a query with at most l bits. The verifier uses at most* $O(\log(|x|))$ *random bits.*
- [Oracle Access] *After tossing its random coins, the verifier generates queries $p, q_1, \ldots, q_k$ and functions $\pi_1, \ldots, \pi_k : [m] \ldots [n]$. The verifier queries p to P receiving answer a and $q_1, \ldots, q_k$ to Q receiving answers $b_1, \ldots, b_k$.*
- [Completeness] *If $x \in L$, there exists oracles P and Q such that with probability at least c V strongly accepts.*
- [Soundness] *If $x \notin L$, for every oracles P, Q, V rejects with probability at least* $1 - s$.

A 1-outer verifier corresponds to the standard notion of canonical inner verifier, as in [8, 9, 7]. For any k, we are able to construct k-outer verifiers.

Theorem 3 (Construction of k-outer verifiers) *For every $k \geq 1$ and for every $s > 0$, there exists a k-outer verifier with perfect completeness, soundness s and answer size* $O(\log k/s)$.

The proof is postponed to Section 4.

Definition 4 (k-Inner Verifier) *A* k-inner verifier *is a randomized oracle algorithm V that is given a sequence of functions $\pi_1, \ldots, \pi_k$ where $\pi_j : \{1,-1\}^m \to \{1,-1\}^n$, and has oracle access to a function $A : \mathcal{F}_n \to \{1,-1\}$ and to a sequence of functions $B_1, \ldots, B_k$ where $B_j : \mathcal{F}_m \to \{1,-1\}$.*

Definition 5 (Decoding Procedure) *A* decoding procedure *is a randomized algorithm D_n such that on input a function $A : \mathcal{F}_n \to \{1,-1\}$ an element of $[n]$.*

Definition 6 (Good Inner Verifier) *A k-inner verifier V is (c,s,q)-good with respect to a decoding procedure D if for any $\pi_1,\ldots,\pi_k : [m] \to [n]$, any $A : \mathcal{F}_n \to \{1,-1\}$, and any $B_1,\ldots,B_k : \mathcal{F}_m \to \{1,-1\}$, the following properties hold.*

- [Number of Queries] *V makes at total number of at most q non-adaptive oracle queries.*
- [Completeness] *if A is the Long code of a, and B_i is the long code of b_i, and $\pi_i(b_i) = a$, then*
$$\mathbf{Pr}[V(A', B_1', \ldots, B_k', \pi_1, \ldots, \pi_k)\text{accepts}] \geq c\ .$$
- [Soundness] *For any constant $\delta > 0$, there is a positive constant $\delta' > 0$ independent of m, n, (but possibly dependent on δ) such that*
$$\textit{If } \mathbf{Pr}[V(A', B_1', \ldots, B_k', \pi_1, \ldots, \pi_k)\text{accepts}] \geq s + \delta$$
$$\textit{Then } \mathbf{Pr}[\textit{ at least two values out of } D(A), \pi_1(D(B_1)), \ldots, \pi_k(D(B_k)) \textit{ are equal}] \geq \delta'\ .$$

Our Composition Theorem is as follows. (The proof is deferred to Section 4.3.)

Theorem 7 *If there exists a (c,s,q)-good k-inner verifier V with respect to a decoding procedure D then for any $\varepsilon > 0$* NP = naPCP$_{c,s+\varepsilon}[\log, q]$.

4 Construction of k-Outer Verifiers and the Composition Theorem

In this section we shall prove Theorem 3 and Theorem 7. Our construction of outer verifiers uses the 2-Prover 1-Round protocol of Raz [21] (indeed, a slight revisitation of it), therefore we will start reviewing its construction, even though it has appeared in several places, including [7, 16].

4.1 A 1-Outer Verifier

It is a consequence of the PCP Theorem [3] that there exists a polynomial time reduction that given an instance φ of 3SAT generates an instance $\hat{\varphi}$ of 3SAT such that if φ is satisfiable then also $\hat{\varphi}$ is satisfiable, and if φ is not satisfiable then every assignment satisfies less then a fraction ρ of the clauses of $\hat{\varphi}$, where $\rho < 1$ is an absolute constant. Using a reduction of Papadimitriou and Yannakakis [20] (see also further elaboration by Feige [11]) we can make sure that every variable in $\hat{\varphi}$ occurs in exactly the same number of clauses. (This will not be really necessary for our purposes, but will simplify the exposition.) The transformation of φ in $\hat{\varphi}$ defines a simple 2-Prover 1-Round proof system: on input a formula φ with N variables and M variables, the verifier has oracle access to two tables $P : [N] \to B$ and $Q : [M] \to [7]$ that are supposedly two encodings of the same satisfying assignment for $\hat{\varphi}$. Specifically, for every variable x, $P(x)$ contains the value of x in the assignment, and for every clause C, $Q(C)$ contains the values of the three variables occurring in C according to the same assignment (the value is encoded as a number between 1 and 7, that is the index of the partial assignment in the lexicographic order among the assignments that satisfy C). The verifier picks at random a clause C in $\hat{\varphi}$ and one of the three variables occurring in C (say, x, the i-th variable in C) and reads $a = P(x)$ and $b = Q(C)$. If b encodes a satisfying assignment b_1, b_2, b_3 for C and $b_i = a$ then the verifier accepts, otherwise it rejects. It is easy to show that this verifier has perfect completeness and soundness $1 - (1-\rho)/3 < 1$.

The soundness of the previously described protocol can be reduced by iterating the protocol several times in parallel. The protocol obtained by t parallel repetitions does the following: it

```
Verifier $V^{\mathrm{out}}$ $(\varphi, P, Q)$
    Randomly pick $p \in [N]$
    Pick $(q, \pi)$ according to $\mathcal{D}(p)$
    Let $a = P(p)$ and $b = Q(q)$
    **accept** iff $\pi(b) = a$
```

Figure 1: A description of the 2-Provers 1-Round protocol of Raz [21].

```
Verifier $Vk^{\mathrm{out}}(\varphi, P, Q)$
    Randomly pick $p \in [N]$
    Sample $k$ pairs $(q_1, \pi_1), \ldots, (q_k, \pi_k)$ from $\mathcal{D}(p)$
    Let $a = P(p)$ and $b_j = Q(q_j)$ for $j = 1, \ldots, k$
    **strongly accept** if $a = \pi_1(b_1) = \cdots = \pi_k(b_k)$
    **weakly accept** if the values $a, \pi_1(b_1), \ldots, \pi_k(b_k)$ are not all different
    **reject** if the values $a, \pi_1(b_1), \ldots, \pi_k(b_k)$ are all different
```

Figure 2: Our k-outer verifier.

picks at random clauses $C_1, \ldots, C_t$ (possibly with repetitions), and picks a variable x_j for every clause C_j. Prover P is supposed to contain an assignment to every t-tuple of variables, and Q an assignment to the variables occurring in every t-tuple of clauses (encoded as an element of $[7]^t$). The verifier asks $(a_1, \ldots, a_t) = P(x_1, \ldots, x_t)$ and $(b_1, \ldots, b_t) = Q(C_1, \ldots, C_t)$ and checks that $\pi(b_1, \ldots, b_t) = (a_1, \ldots, a_t)$ where $\pi : [7]^t \to \{0,1\}^t$ is the function that "extracts" (or "projects") from the values of all the variables occurring in $C_1, \ldots, C_t$ the values of $x_1, \ldots, x_t$. We notice that since every variable occurs in exactly the same number of clauses (and every clause contains exactly the same number of variables) the verifier generates the same distribution if it first picks at random t variables $x_1, \ldots, x_t$ and then a clauses C_i for every x_i, where C_i is chosen uniformly among the clauses where x_i occurs.

Raz proves that the verifier obtained by making t parallel repetitions has soundness $2^{-\Omega(t)}$ and, by definition, it has perfect completeness and answer size $t \log 7$. From now on, we will abstract all the details of Raz verifier that are not necessary for our proof, and we will use the following description of it (see Figure 1): it has oracle access to tables $P \in \{0,1\}^{n \times N}$ and $Q \in \{0,1\}^{m \times M}$. It first picks a random entry in P (i.e. a uniformly distributed number p between 1 and N) and then decides the query q for P and the projection function π; we make no assumption on how q and π are selected given that p is selected, and we call their distribution $\mathcal{D}(p)$. For every $\sigma > 0$, such a verifier exists with perfect completeness, soundness σ and answer size $\max\{m, n\} = O(\log 1/\sigma)$.

4.2 Construction of k-Outer Verifiers

Our k-outer verifier is depicted in Figure 2.

We want to prove that whenever the k-outer verifier accepts with probability larger than σ then the formula is satisfiable. We will prove the latter statement by using the soundness condition of Raz's verifier, and showing how to construct proofs for Raz verifier that make it accept with sufficiently large probability.

Let P and Q be proofs that the k-inner verifier weakly accepts with probability at least s, that is

$$\mathop{\mathbf{E}}_{p\in[N],(q_1,\pi_1),\ldots,(q_k,\pi_k)\in\mathcal{D}(p)}[P(p),\pi_1(Q(q_1)),\ldots,\pi_k(Q(q_k)) \text{ not all different }] \geq s$$

We will consider the pair of proofs (P',Q) where P' is constructed randomly as follows:

- For every $p \in [N]$, we sample $k-1$ pairs $(q_1,\pi_1),\ldots,(q_{k-1},\pi_{k-1})$ from $\mathcal{D}(p)$, and then we select a random element a from the multiset $P(p),\pi_1(Q(q_1)),\ldots,\pi_k(Q(q_{k-1}))$, and we let $P'(p) = a$.

Now we claim that Raz's verifier accepts (P',Q) with probability at least s/k^2, where the probability is taken both over the verifier's coin tosses and over the construction of P'.

We first observe that the probability that Raz's verifier accepts is equal to the probability that the following experiment succeeds:

- Pick randomly $p \in [N]$, then sample $k-1$ pairs $(q_1,\pi_1),\ldots,(q_{k-1},\pi_{k-1})$ from $\mathcal{D}(p)$, choose at random another pair (q,π) from $\mathcal{D}(p)$, choose at random an element a from $P(p),\pi_1(Q(q_1)),\ldots,\pi_{k-1}(Q_{k-1}(q_{k-1}))$, accept iff $a = \pi(Q(q))$.

This probability is clearly the same as the probability that the following random process succeeds.

- Pick randomly $p \in [N]$, sample k pairs $(q_1,\pi_1),\ldots,(q_k,\pi_k)$, pick a random $j \in [k]$ pick a random element a in the multiset $\{P(p)\} \cup \{\pi_i(Q(q_i))\}_{i\neq j}$, accept iff $a = \pi_j(Q(q_j))$.

Conditioned upon the k-outer verifier weakly accepting when it selects $p,(q_1,\pi_1),\ldots,(q_k,\pi_k)$, the previous process accepts with probability at least $1/k^2$. It follows that Raz's verifier accepts with probability at least s/k^2. This acceptance probability is expected over the choices of P', but there must be a choice of P' for which the acceptance probability is at least that much.

4.3 The Composition Theorem

We now come to the proof of the Composition Theorem. Let V^{in} be a (c,s,q)-good k-inner verifier, and let $\varepsilon > 0$ be fixed. The PCP verifier V^{comp} that we are claiming to exist will expect as a proof a pair of tables LP and LQ that be the entry-wise encoding with the Long code of a valid pair of proof oracles P and Q for the k-outer verifier. We will use a k-outer verifier with perfect completeness and soundness σ; we will specify σ later but we anticipate that it will be a constant depending only on ε. We denote by $FP(q)$ (respectively $FQ(q)$) the *folding of* the q-th entry of LP (respectively, LQ). Notice that even though V^{comp} has only oracle access to LP and LQ, it can simulate an oracle access to FP and FQ as described in Section 3.

The V^{comp} verifier is described in Figure 3. It picks queries $p,q_1,\ldots,q_k$ and projections $\pi_1,\ldots,\pi_k$ as the k-outer verifier would do, and then it executes the inner verification procedure.

Claim 1 *V^{comp} has completeness c and asks at most q queries.*

PROOF: V^{comp} accesses the proof only by running V^{in}, and by hypothesis V^{in} reads at most q bits. When LP and LQ are valid proof, the input that is passed to V^{in} satisfies the completeness condition of V^{in}. Therefore V^{in} accepts with probability at least c over its coin tosses, for every particular coin toss of V^{comp}. This implies that V^{comp} accepts with probability at least c. □

```
Verifier $V^{\mathrm{comp}}(\varphi, LP, LQ)$
    Randomly pick $p \in [N]$
    Sample $k$ pairs $(q_1, \pi_1), \ldots, (q_k, \pi_k)$ from $\mathcal{D}(p)$
    Let $A = FP(p)$ and $B_j = FQ(q_j)$ for $j = 1, \ldots, k$
    Run $V^{\mathrm{in}}(A, B_1, \ldots, B_k, \pi_1, \ldots, \pi_k)$
```

Figure 3: The composed verifier that uses a k-inner verifier V^{in}.

Claim 2 *V^{comp} has soundness at least $s + \varepsilon$.*

Proof: We have to prove that when V^{comp} accepts its oracle proofs LP and LQ with probability at least $s + \varepsilon$ then φ is satisfiable. Using the soundness condition of the k-outer verifier, in order to show that φ is satisfiable it is enough to exhibit oracle proofs P and Q that would make the k-outer verifier weakly accept with probability at least σ.

Let LP, LQ and φ be such that

$$\mathbf{Pr}[V^{\mathrm{comp}}(\varphi, LP, LQ) \text{ accepts }] \geq s + \varepsilon \tag{1}$$

and let N and M be the number of entries of LP and LQ, respectively. Given LP and LQ, we define oracle proofs P and Q for the k-outer verifier with the following randomized procedure:

1. Independently for $p = 1, \ldots, N$: set $P(p) = D(FP(p))$.

2. Independently for $q = 1, \ldots, M$: set $Q(q) = D(FQ(q))$.

Remember that D is a randomized algorithm. In the above definition of P and Q the executions of D have to be independent each time.

An averaging argument using (1) shows that for at least a fraction $\varepsilon/2$ of the random choices of V^{comp}, i.e. for at least a fraction $\varepsilon/2$ of the $p, (q_1, \pi_1), \ldots, (q_k, \pi_k)$, the inner verifier accepts with probability at least $s + \varepsilon/2$; we call G the set of such good $k + 1$-tuples. By the soundness condition of the inner verifier we have that there exists some constant $\delta = \delta_{\varepsilon/2}$ such that for every $(p, (q_1, \pi_1), \ldots, (q_k, \pi_k)) \in G$, it is the case that

$$\mathbf{Pr}_{\text{random choices of}D}[P(p), \pi_1(Q(q_1)), \ldots, \pi_k(Q(q_k)) \text{ not all different }] \geq \delta$$

The probability that the k-outer verifier weakly accepts P and Q (expected over the way P and Q are chosen) is

$$\begin{aligned}
& \mathbf{Pr}_{P,Q,p \in [N],\ (q_1,\pi_1),\ldots,(q_k,\pi_k) \in \mathcal{D}(p)}[P(p), \pi_1(Q(q_1)), \ldots, \pi_k(Q(q_k)) \text{ not all different}] \\
\geq\ & \mathbf{Pr}_{P,Q}[P(p), \pi_1(Q(q_1)), \ldots, \pi_k(Q(q_k)) \text{ not all different} | (p, (q_1, \pi_1), \ldots, (q_k, \pi_k)) \in G] \\
& \cdot \mathbf{Pr}_{p \in [N],\ (q_1,\pi_1),\ldots,(q_k,\pi_k) \in \mathcal{D}(p)}[(p, (q_1, \pi_1), \ldots, (q_k, \pi_k)) \in G] \\
\geq\ & \frac{\varepsilon}{2}\delta > \sigma
\end{aligned}$$

This completes the proof of the Composition Theorem. □

```
Inner$_{k,\varepsilon}$($A, B_1, \ldots, B_k, \pi_1, \ldots, \pi_k$)
  Choose uniformly at random $f_1, f_2 \in \mathcal{F}_n$ and $g_1, \ldots, g_k \in \mathcal{F}_m$
  For $i = 1, 2$ and $j = 1, \ldots, k$
    choose at random $e_{i,j} \in \mathcal{F}_m$ such that
      $\forall b \in \{1, -1\}^m . \mathbf{Pr}[e_{i,j}(b) = 1] = 1 - \varepsilon$
  if for all $i = 1, 2$ and $j = 1, \ldots, k$
    $A(f_i)B_j(g_j) = B_j((f_i \circ \pi_j)g_j e_{i,j})$
      then accept
      else reject
```

Figure 4: The inner verifier.

5 Main Result

In this section we describe the inner verifier used in our paper and give an outline of its analysis.

5.1 The Inner Verifier

For any k and $\varepsilon > 0$, our inner verifier Inner$_{k,\varepsilon}$ is described in Figure 4. Inner$_{k,\varepsilon}$ is obtained by iterating a basic 3-query inner verifier by Håstad [16]. The basic protocol would access two tables A and B, would pick a function f uniformly from the domain of A, a function g uniformly from the domain of B, and a function e from the domain of B but with a non-uniform distribution; the verifier would accept iff $A(f)B(g) = B((f \circ \pi)ge)$. By recycling queries, we manage to execute $2k$ iterations of the basic protocol while using only $3k + 2$ queries instead of $6k$ queries. Specifically, each of the two queries that we ask on A is used k times, and some of the queries that we ask on B are used twice. The recycling mechanism that we employ in Inner$_{k,\varepsilon}$ is similar to the one used in the $K_{2,k}$ test of [25]. The latter was, however, only a codeword test, and so it had as input a single table.

5.2 Background on Fourier Analysis

To analyze the properties (i.e., the soundness) of this verifier, we need to resort to Fourier analysis. We give some background here. Recall the definition of linear functions from Section 3. We will be using three standard properties of linear functions, the fact that they are linear in α, linear in f and that they are equally often 1 and -1, except for the case of the function that is identically 1:

$$l_\alpha(f)l_\beta(f) = l_{\alpha\Delta\beta}(f) \ , \ l_\alpha(f)l_\alpha(g) = l_\alpha(fg) \ , \ \mathop{\mathbf{E}}_f l_\alpha(f) = \begin{cases} 1 & \text{If } \alpha = \emptyset \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

For a function $\pi : \{1, -1\}^m \to \{1, -1\}^n$ and a set $\beta \subseteq \{1, -1\}^m$ we define $\pi(\beta) = \{\pi(b) : b \in \beta\}$ and also the less standard notation $\pi^\oplus(\beta) = \Delta_{b\in\beta}\{\pi(b)\}$. In words, $\pi^\oplus(\beta)$ contains all the elements $a \in \{1, -1\}^m$ such that a is the image of an *odd* number of elements of β. Specifically, we will use the fact that $l_{\pi^\oplus(\beta)}(f) = l_\beta(f \circ \pi)$ (where the l's come from the appropriate domains) and that $\pi^\oplus(\beta) \subseteq \pi(\beta)$.

We can see a function $A : \mathcal{F}_n \to \{1, -1\}$ as a real-valued function $A : \mathcal{F}_n \to \mathcal{R}$. The set of functions $A : \mathcal{F}_n \to \mathcal{R}$ is a vector space over the reals of dimension 2^{2^n}. We define the following scalar product between functions.

$$A \cdot B = \frac{1}{2^{2^n}} \sum_{f \in \mathcal{F}_n} A(f)B(f) = \mathop{\mathbf{E}}_f [A(f)B(f)] \ .$$

The set of linear functions is easily seen to form an orthonormal basis for the set of functions $A : \mathcal{F}_n \to \mathcal{R}$. This implies that for any function $A : \mathcal{F}_n \to \mathcal{R}$ we have

$$A(f) = \sum_\alpha \hat{A}_\alpha l_\alpha(f) \text{ where } \hat{A}_\alpha = A \cdot l_\alpha$$

Parseval's identity implies that for every $A : \{1, -1\}^n \to \{1, -1\}$ it holds $\sum_\alpha \hat{A}_\alpha^2 = 1$. Among other things, this implies that for a function $A : \mathcal{F}_n \to \{1, -1\}$, we have $|\hat{A}_\alpha| \leq 1$ for any α.

Finally, from the definition of folding (i.e., $A'(f) = -A'(-f)$ for any f) it follows that $\hat{A}'_\alpha = 0$ for any α of even size, in particular for $\alpha = \emptyset$.

5.3 The Decoding Procedure

The decoding procedure D is based on the fact that, by Parseval's identity, the squares of the Fourier coefficients $\hat{A}_\alpha$'s and $\hat{B}_\beta$'s sum to 1 and can hence be thought of as a probability distribution.

For a table $A : \{0,1\}^n \to \{0,1\}$, the decoding procedure is defined as follows:

- Pick a set $\alpha \subseteq [n]$ with probability $\hat{A}_\alpha^2$; pick a random element $a \in \alpha$, return a. (Notice that this is well defined only when $\hat{A}_\emptyset = 0$, which is true for a folded A.)

The claims about the number of queries made by the inner verifier and about its completeness property are easily verified. Hence we turn our attention to its soundness. In order to do so, we need to analyze the quantity $\mathbf{Pr}[D(A), \pi_1(D(B_1)), \ldots, \pi_k(D(B_k))$ not all different]. A simple computation yields that

$$\begin{aligned}
& \mathbf{Pr}[D(A), \pi_1(D(B_1)), \ldots, \pi_k(D(B_k)) \text{ not all different }] \\
\geq \ & \max\{\mathbf{Pr}[\exists j. D(A) = \pi_j(D(B_j))], \mathbf{Pr}[\exists i, j. \pi_i(D(B_i)) = \pi_j(D(B_j))]\} \\
\geq \ & \max\{\max_j \mathbf{Pr}[D(A) = \pi_j(D(B_j))], \max_{i \neq j} \mathbf{Pr}[\pi_i(D(B_i)) = \pi_j(D(B_j))]\} \\
\geq \ & \max\left\{ \max_j \left(\sum_{\alpha,\beta : \pi_j(\beta) \cap \alpha \neq \emptyset} \frac{1}{|\alpha|} \hat{A}_\alpha^2 \frac{1}{|\beta|} \hat{B}_{j,\beta}^2 \right), \right. \\
& \left. \max_{i \neq j} \left(\sum_{\beta_1, \beta_2 : \pi_i(\beta_1) \cap \pi_j(\beta_2) \neq \emptyset} \frac{1}{|\beta_1|} \hat{B}_{i,\beta_1}^2 \frac{1}{|\beta_2|} \hat{B}_{j,\beta_2}^2 \right) \right\} \\
\geq \ & \max\left\{ \mathop{\mathbf{E}}_j \left(\sum_{\alpha,\beta : \pi_j(\beta) \cap \alpha \neq \emptyset} \frac{1}{|\alpha|} \hat{A}_\alpha^2 \frac{1}{|\beta|} \hat{B}_{j,\beta}^2 \right), \right. \\
& \left. \mathop{\mathbf{E}}_{i \neq j} \left(\sum_{\beta_1, \beta_2 : \pi_i(\beta_1) \cap \pi_j(\beta_2) \neq \emptyset} \frac{1}{|\beta_1|} \hat{B}_{i,\beta_1}^2 \frac{1}{|\beta_2|} \hat{B}_{j,\beta_2}^2 \right) \right\}
\end{aligned}$$

To lower bound the right hand side we try to derive some lower bounds on the Fourier coefficients of A and B_i's using the acceptance probability of the inner verifier.

5.4 The Analysis

We let k and ε be fixed for the rest of this section.

Proposition 8 *The acceptance probability of* $\mathsf{Inner}_{k,\varepsilon}$ *is*

$$\frac{1}{2^{2k}} \sum_{S\subseteq[2]\times[k]} T_S \text{ where } T_S \stackrel{\triangle}{=} \mathop{\mathbf{E}}_{f_1,f_2,g_1,\ldots,g_k e_{1,1},\ldots,e_{2,k}} \left[\prod_{(i,j)\in S} A(f_i) B_j(g_j) B_j((f_i \circ \pi_j) g_j e_{i,j}) \right]$$

This proposition is proven as in [25]; it follows from the arithmetization of the acceptance condition of the inner verifier, which is a function of the $A(f_i)$'s, $B_j(g_j)$'s etc. To analyze the expression above, we need to analyze the T_S's. Of course when S is empty then T_S is 1. We want to show that if T_S is high for any other set S then the success probability of the decoding procedure is high. We start by analyzing T_S and deriving upper bounds for these in terms of the Fourier coefficients of A and B_i's. We divide the analysis into two cases, depending on whether none or at least one of the sets $V_S \stackrel{\triangle}{=} \{j : (1,j) \in S\}$ and $W_S \stackrel{\triangle}{=} \{j : (2,j) \in S\}$ have odd cardinality.

Lemma 9 *Let $S \subseteq [2] \times [k]$ be a non-empty set such that both V_S and W_S have even cardinality. Then, there exists some $J \subseteq [k]$ such that*

$$T_S \leq \sum_{\{\beta_j\}_{j\in J} : \exists i,j \in J \pi_i(\beta_i) \cap \pi_j(\beta_j) \neq \emptyset} \prod_{j\in J} \hat{B}^2_{j,\beta_j} (1-2\varepsilon)^{|\beta_j|} \tag{3}$$

Proof: Let $L = V_S - W_S$, $U = V_S \cap W_S$ and $R = W_S - V_S$. Since T_S contains an even number of occurrences of $A(f_1)$ and an even number of occurrences of $A(f_2)$, it can be rewritten as:

$$T_S = \mathop{\mathbf{E}}_{f_1,f_2,g_1,\ldots,g_k,e_{1,1},\ldots,e_{2,k}} \left[\prod_{(i,j)\in S} B_j(g_j) B_j((f_i \circ \pi_j) g_j e_{i,j}) \right] \tag{4}$$

We now further split the product to take care of possible cancellations of the terms of the form $B_j(g_j)$, thus getting

$$T_S = \mathop{\mathbf{E}}_{f_1,f_2,g_1,\ldots,g_k,e_{1,1},\ldots,e_{2,k}} \left[\left(\prod_{j\in L} B_j(g_j) B_j((f_1 \circ \pi_j) g_j e_{1,j}) \right) \right. \tag{5}$$

$$\left. \left(\prod_{j\in R} B_j(g_j) B_j((f_2 \circ \pi_j) g_j e_{2,j}) \right) \left(\prod_{j\in U} B_j((f_1 \circ \pi_j) g_j e_{1,j}) B_j((f_2 \circ \pi_j) g_j e_{2,j}) \right) \right]$$

Now, we expand each term in the above product.

$$\text{For } j \in L \cup R, \quad B(g_j) = \sum_{\beta_j} \hat{B}_{j,\beta_j} l_{\beta_j}(g_j)$$

$$\text{For } j \in L, \quad B_j((f_1 \circ \pi_j) g_j e_{1,j}) = \sum_{\gamma_j} \hat{B}_{j,\gamma_j} l_{\pi_j^{\oplus}(\gamma_j)}(f_1) l_{\gamma_j}(g_j) l_{\gamma_j}(e_{1,j})$$

$$\text{For } j \in R, \quad B_j((f_2 \circ \pi_j) g_j e_{2,j}) = \sum_{\gamma_j} \hat{B}_{j,\gamma_j} l_{\pi_j^{\oplus}(\gamma_j)}(f_2) l_{\gamma_j}(g_j) l_{\gamma_j}(e_{2,j})$$

$$\text{For } j \in U, \quad B_j((f_1 \circ \pi_j) g_j e_{1,j}) = \sum_{\beta_j} \hat{B}_{j,\beta_j} l_{\pi_j^{\oplus}(\beta_j)}(f_i) l_{\beta_j}(g_j) l_{\beta_j}(e_{1,j})$$

$$\text{and} \quad B_j((f_2 \circ \pi_j) g_j e_{2,j}) = \sum_{\gamma_j} \hat{B}_{j,\gamma_j} l_{\pi_j^{\oplus}(\beta_j)}(f_2) l_{\gamma_j}(g_j) l_{\gamma_j}(e_{2,j})$$

We now distribute the summations, use the linearity of expectation, the properties of linear functions, and observe that

$$\mathop{\mathbf{E}}_{e_{i,j}} l_\eta(e_{i,j}) = (1-2\varepsilon)^{|\eta|}.$$

The result of these manipulations is the following expression

$$\sum_{\{\beta_j\},\{\gamma_j\}} \hat{B}_{j,\beta_j} \hat{B}_{j,\gamma_j} \mathop{\mathbf{E}}_{f_1,f_2,g_1,\ldots,g_k} \Bigg[l_{\Delta_{j\in L}\pi_j^{\oplus}(\gamma_j)\Delta_{j\in U}\pi_j^{\oplus}(\beta_j)}(f_1) l_{\Delta_{j\in R\cup U}\pi_j^{\oplus}(\gamma_j)}(f_2)$$

$$\left(\prod_{j\in L\cup R\cup U} l_{\beta_j \Delta \gamma_j}(g_j) \right) (1-2\varepsilon)^{\sum_{j\in U}|\beta_j| + \sum_{j\in L\cup R\cup U}|\gamma_j|} \Bigg]$$

For any fixed value of the β_j's, the expectation above is zero unless all of the following conditions hold; and if the conditions hold, then the expectation is one:

$$\beta_j = \gamma_j, \ \forall j \in L \cup R \cup U$$

$$\Delta_{j\in L}\pi_j^{\oplus}(\gamma_j)\Delta_{j\in U}\pi_j^{\oplus}(\beta_j) = \Delta_{j\in L\cup U}\pi_j^{\oplus}(\beta_j) = \emptyset$$

$$\Delta_{j\in R\cup U}\pi_j^{\oplus}(\gamma_j) = \Delta_{j\in R\cup U}\pi_j^{\oplus}(\beta_j) = \emptyset$$

Thus T_S "simplifies" to:

$$T_S = \sum_{\{\beta_j, j\in L\cup R\cup U : \Delta_{j\in R\cup U}\pi_j^{\oplus}(\beta_j) = \emptyset\}} \hat{A}_{\Delta_{j\in L\cup U}\pi_j^{\oplus}(\beta_j)} \left(\prod_{j\in L\cup R\cup U} \hat{B}^2_{j,\beta_j} \right) (1-2\varepsilon)^{\sum_{j\in L\cup U}|\beta_j| + \sum_{j\in R\cup U}|\beta_j|}.$$

To simplify the above expression, we ignore some of the $(1-2\varepsilon)$ terms. We also ignore the constraint $\Delta_{j\in R\cup U}\pi_j^{\oplus}(\beta_j) = \emptyset\}$. Then by using Parseval's Identity on every $j \in R$, we get the following upper bound on T_S.

$$\begin{aligned} T_S &\le \sum_{\{\beta_j, j\in L\cup U : \Delta_{j\in L\cup U}\pi_j^{\oplus}(\beta_j) = \emptyset\}} \left(\prod_{j\in L\cup U} \hat{B}^2_{j,\beta_j} \right) (1-2\varepsilon)^{\sum_{j\in L\cup U}|\beta_j|} \\ &\le \sum_{\{\beta_l, l\in L\cup U : \exists i,j\in L\cup U \text{ s.t. } \pi_i(\beta_i)\cap\pi_j(\beta_j)\neq\emptyset\}} \left(\prod_{j\in L\cup U} \hat{B}^2_{j,\beta_j} \right) (1-2\varepsilon)^{\sum_{j\in L\cup U}|\beta_j|}, \end{aligned}$$

where the second inequality above follows since the summands are positive and the condition $\Delta_{j\in L\cup U}\pi_j^{\oplus}(\beta_j) = \emptyset$ implies the condition $\exists i,j \in L \cup U$ s.t. $\pi_i(\beta_i) \cap \pi_j(\beta_j) \neq \emptyset$. □

Thus if T_S is large for a set S for which both V_S and W_S are even, then the lemma above implies that some (complicated) expression related to the Fourier coefficients is large. The following lemma relates this (complicated) expression to the success probability of the decoding procedure in Section 5.3.

Lemma 10 *Let $B_1, \ldots, B_k$ be such that $\hat{B}_{i,\emptyset} = 0$ for every $i \in [k]$. Assume that for some $\delta, \varepsilon > 0$, $J \subseteq [k]$ the following holds:*

$$\sum_{\{\beta_j\}_{j\in J}: \exists i,j \in J, i\neq j \text{ s.t. } \pi_i(\beta_i)\cap\pi_j(\beta_j)\neq\emptyset} \prod_{l=1} \hat{B}^2_{l,\beta_l}(1-2\varepsilon)^{|\beta_l|} \geq \delta.$$

Then

$$\mathop{\mathbf{E}}_{i,j\in[k], i\neq j}\left[\sum_{\beta_1,\beta_2:\pi_i(\beta_1)\cap\pi_j(\beta_2)\neq\emptyset} \frac{1}{|\beta_1|}\hat{B}^2_{i,\beta_1}\frac{1}{|\beta_2|}\hat{B}^2_{j,\beta_2}\right] \geq \frac{\delta}{ck^2}$$

where $c = c_\varepsilon$ is a constant that depends on ε but not on δ nor on k.

Proof: We prove the lemma in two steps. First we show that

$$\mathop{\mathbf{E}}_{i,j\in[k], i\neq j}\left[\sum_{\beta_1,\beta_2:\pi_i(\beta_1)\cap\pi_j(\beta_2)\neq\emptyset} (1-\varepsilon)^{|\beta_1|}\hat{B}^2_{i,\beta_1}(1-\varepsilon)^{|\beta_2|}\hat{B}^2_{j,\beta_2}\right]$$
$$\geq \frac{1}{\binom{k}{2}} \sum_{\{\beta_j\}_{j\in J}: \exists i,j \in J, i\neq j \text{ s.t. } \pi_i(\beta_i)\cap\pi_j(\beta_j)\neq\emptyset} \prod_{l=1} \hat{B}^2_{l,\beta_l}(1-2\varepsilon)^{|\beta_l|} \tag{6}$$

In order to obtain this inequality, consider the following probabilistic experiment. Pick two random indices i, j distinct from $[k]$ and then pick β_1 and β_2 independently with probability $B^2_{i,\beta_1}(1-\varepsilon)^{|\beta_i|}$ and $B^2_{j,\beta_2}(1-\varepsilon)^{|\beta_j|}$ respectively. Now consider the probability of the event that $\pi_1(\beta_1)$ and $\pi_2(\beta_2)$ have a non-empty intersection. This is clearly equal to the left-hand side of (6) above. One way to lower bound the probability of this event is by considering the following experiment: For every $i \in J$, pick β_i with probability $B^2_{i,\beta_i}(1-\varepsilon)^{|\beta_i|}$. Now pick i and j distinct at random from $[k]$ and consider the event that both $i, j \in J$ and $\pi_i(\beta_i)$ intersects $\pi_j(\beta_j)$. This probability can be lower bounded by the probability that there exists a pair $(i_0, j_0) \in J$ such that $\pi_{i_0}(\beta_{i_0})$ and $\pi_{j_0}(\beta_{j_0})$ intersect, times the probability that when we pick i, j distinct at random from $[k]$, then we get the pair (i_0, j_0). This probability is easily bounded by

$$\frac{1}{\binom{k}{2}} \sum_{\substack{\{\beta_j\}_{j\in J}: \\ \exists i,j \in J, i\neq j \text{ s.t. } \pi_i(\beta_i)\cap\pi_j(\beta_j)\neq\emptyset}} \prod_{l=1} \hat{B}^2_{l,\beta_l}(1-2\varepsilon)^{|\beta_l|}$$

This yields the inequality (6) above.

To complete the proof of the lemma we observe that for every positive ε and x the following inequality holds:

$$(1-\varepsilon)^x \leq \frac{1}{e\varepsilon x}. \tag{7}$$

To see the above inequality, we use the following two inequalities: 1) For every ε s.t. $0 < \varepsilon < 1$, $(1-\varepsilon)^{1/\varepsilon} \leq e^{-1}$; and 2) For every positive y, $ye^{-y} \leq e^{-1}$.

The lemma now follows from the following series of inequalities:

$$\mathop{\mathbf{E}}_{i,j\in[k], i\neq j}\left[\sum_{\beta_1,\beta_2:\pi_i(\beta_1)\cap\pi_j(\beta_2)\neq\emptyset} \frac{1}{|\beta_1|}\hat{B}^2_{i,\beta_1}\frac{1}{|\beta_2|}\hat{B}^2_{j,\beta_2}\right]$$
$$\geq \frac{1}{4e^2\varepsilon}\mathop{\mathbf{E}}_{i,j\in[k], i\neq j}\left[\sum_{\beta_1,\beta_2:\pi_i(\beta_1)\cap\pi_j(\beta_2)\neq\emptyset} (1-2\varepsilon)^{|\beta_1|}\hat{B}^2_{i,\beta_1}(1-2\varepsilon)^{|\beta_2|}\hat{B}^2_{j,\beta_2}\right] \text{ (Using (7))}$$

$$\begin{aligned}
&\geq \frac{1}{4e^2\varepsilon\binom{k}{2}} \sum_{\substack{\{\beta_j\}_{j\in J}: \\ \exists i,j\in J, i\neq j \text{ s.t. } \pi_i(\beta_i)\cap\pi_j(\beta_j)\neq\emptyset}} \prod_{l=1} \hat{B}^2_{l,\beta_l}(1-2\varepsilon)^{|\beta_l|} \text{ (Using (6))} \\
&\geq \frac{1}{4e^2\varepsilon\binom{k}{2}}\delta \text{ (Using the hypothesis)} \\
&\geq \frac{\delta}{2e^2\varepsilon k^2}.
\end{aligned}$$

□

Notice that the expression being lower bounded in Lemma 10 is a lower bound on the success of the decoding procedure of Section 5.3. Thus the following corollary summarizes the effect of Lemmas 9 and 10.

Corollary 11 *Let $S \subseteq [2]\times[k]$ be such that both V_S and W_S have even cardinality. Then if $T_S \geq \delta$, then the decoding procedure of Section 5.3 leads to weak acceptance with probability at least $\frac{\delta}{ck^2}$, where c is a constant depending only on ε.*

We now move on to the case where at least one of V_S and W_S is odd. We subdivide the analysis into two further cases: 1) When both V_S and W_S are odd and 2) When exactly one is odd. In both cases the bound obtained on T_S is the same, however the analysis is slightly different. Then as in Lemma 10 we relate the new bound to the success of the decoding procedure.

Lemma 12 *Let $S \subseteq [2]\times[k]$ be such that both V_S and W_S have odd cardinality. Then there exists a set $J \subseteq [k]$ such that*

$$T_S \leq \sum_{\{\beta_j\}_{j\in J}} |\hat{A}_{\Delta_{j\in J}\pi_j^{\oplus}(\beta_j)}| \left(\prod_{j\in J} \hat{B}^2_{j,\beta_j}(1-2\varepsilon)^{|\beta_j|}\right) \tag{8}$$

PROOF: Let $L = V_S - W_S$, $U = V_S \cap W_S$ and $R = W_S - V_S$. Since T_S contains an odd number of occurrences of $A(f_1)$ and an odd number of occurrences of $A(f_2)$, it can be rewritten as:

$$T_S = \mathop{\mathbf{E}}_{f_1,f_2,g_1,\ldots,g_k,e_{1,1},\ldots,e_{2,k}} \left[A(f_1)A(f_2)\prod_{(i,j)\in S} B_j(g_j)B_j((f_i\circ\pi_j)g_je_{i,j})\right] \tag{9}$$

We now further split the product to take care of possible cancellations of the terms of the form $B_j(g_j)$, thus getting

$$T_S = \mathop{\mathbf{E}}_{f_1,f_2,g_1,\ldots,g_k,e_{1,1},\ldots,e_{2,k}} \left[A(f_1)A(f_2)\left(\prod_{j\in L} B_j(g_j)B_j((f_1\circ\pi_j)g_je_{1,j})\right)\right. \tag{10}$$

$$\left.\left(\prod_{j\in R} B_j(g_j)B_j((f_2\circ\pi_j)g_je_{2,j})\right)\left(\prod_{j\in U} B_j((f_1\circ\pi_j)g_je_{1,j})B_j((f_2\circ\pi_j)g_je_{2,j})\right)\right]$$

Now, we expand each term in the above product.

$$\text{For } i=1,2, \quad A(f_i) = \sum_{\alpha_i}\hat{A}_{\alpha_i}l_{\alpha_i}(f_i)$$

$$
\begin{aligned}
\text{For } j \in L \cup R, \quad & B(g_j) = \sum_{\beta_j} \hat{B}_{j,\beta_j} l_{\beta_j}(g_j) \\
\text{For } j \in L, \quad & B_j((f_1 \circ \pi_j) g_j e_{1,j}) = \sum_{\gamma_j} \hat{B}_{j,\gamma_j} l_{\pi_j^{\oplus}(\gamma_j)}(f_1) l_{\gamma_j}(g_j) l_{\gamma_j}(e_{1,j}) \\
\text{For } j \in R, \quad & B_j((f_2 \circ \pi_j) g_j e_{2,j}) = \sum_{\gamma_j} \hat{B}_{j,\gamma_j} l_{\pi_j^{\oplus}(\gamma_j)}(f_2) l_{\gamma_j}(g_j) l_{\gamma_j}(e_{2,j}) \\
\text{For } j \in U, \quad & B_j((f_1 \circ \pi_j) g_j e_{1,j}) = \sum_{\beta_j} \hat{B}_{j,\beta_j} l_{\pi_j^{\oplus}(\beta_j)}(f_i) l_{\beta_j}(g_j) l_{\beta_j}(e_{1,j}) \\
\text{and} \quad & B_j((f_2 \circ \pi_j) g_j e_{2,j}) = \sum_{\gamma_j} \hat{B}_{j,\gamma_j} l_{\pi_j^{\oplus}(\beta_j)}(f_2) l_{\gamma_j}(g_j) l_{\gamma_j}(e_{2,j})
\end{aligned}
$$

We now distribute the summations, use the linearity of expectation, the properties of linear functions, and observe that

$$\mathop{\mathbf{E}}_{e_{i,j}} l_\eta(e_{i,j}) = (1-2\varepsilon)^{|\eta|}.$$

The result of these manipulations is the following expression

$$
\sum_{\alpha_1,\alpha_2,\{\beta_j\},\{\gamma_j\}} \hat{A}_{\alpha_1} A_{\alpha_2} \hat{B}_{j,\beta_j} \hat{B}_{j,\gamma_j} \mathop{\mathbf{E}}_{f_1,f_2,g_1,\ldots,g_k} \Bigg[l_{\alpha_1 \Delta_{j\in L} \pi_j^{\oplus}(\gamma_j) \Delta_{j\in U} \pi_j^{\oplus}(\beta_j)}(f_1) l_{\alpha_2 \Delta_{j \in R \cup U} \pi_j^{\oplus}(\gamma_j)}(f_2)
$$

$$
\left(\prod_{j \in L \cup R \cup U} l_{\beta_j \Delta \gamma_j}(g_j) \right) (1-2\varepsilon)^{\sum_{j\in U} |\beta_j| + \sum_{j \in L\cup R\cup U} |\gamma_j|} \Bigg]
$$

For any fixed value of α_1, α_2 and β_j's, the expectation above is zero unless all of the following conditions hold; and if the conditions hold, then the expectation is one:

$$\beta_j = \gamma_j, \ \forall j \in L \cup R \cup U$$

$$\alpha_1 = \Delta_{j\in L} \pi_j^{\oplus}(\gamma_j) \Delta_{j \in U} \pi_j^{\oplus}(\beta_j) = \Delta_{j \in L\cup U} \pi_j^{\oplus}(\beta_j)$$

$$\alpha_2 = \Delta_{j \in R \cup U} \pi_j^{\oplus}(\gamma_j) = \Delta_{j\in R\cup U} \pi_j^{\oplus}(\beta_j)$$

We thus T_S "simplifies" to:

$$
\begin{aligned}
T_S &= \sum_{\{\beta_j, j \in L\cup R\cup U\}} \hat{A}_{\Delta_{j\in L\cup U} \pi_j^{\oplus}(\beta_j)} \hat{A}_{\Delta_{j \in R\cup U} \pi_j^{\oplus}(\beta_j)} \left(\prod_{j\in L\cup R\cup U} \hat{B}^2_{j,\beta_j} \right) (1-2\varepsilon)^{\sum_{j\in L\cup U}|\beta_j| + \sum_{j \in R\cup U} |\beta_j|} \\
&\leq \sum_{\{\beta_j, j\in L\cup R\cup U\}} |\hat{A}_{\Delta_{j\in L\cup U}\pi_j^{\oplus}(\beta_j)}| \left(\prod_{j \in L\cup R\cup U} \hat{B}^2_{j,\beta_j} \right) (1-2\varepsilon)^{\sum_{j\in L\cup U}|\beta_j|} \\
&= \left(\sum_{\{\beta_j, j\in L\cup U\}} |\hat{A}_{\Delta_{j\in L\cup U}\pi_j^{\oplus}(\beta_j)}| \left(\prod_{j\in L\cup U} \hat{B}^2_{j,\beta_j} \right) (1-2\varepsilon)^{\sum_{j\in L\cup U}|\beta_j|} \right) \left(\sum_{\{\beta_j, j\in R\}} \left(\prod_{j\in R} \hat{B}^2_{j,\beta_j} \right) \right) \\
&= \sum_{\{\beta_j, j\in L\cup U\}} |\hat{A}_{\Delta_{j\in L\cup U}\pi_j^{\oplus}(\beta_j)}| \left(\prod_{j\in L\cup U} \hat{B}^2_{j,\beta_j} \right) (1-2\varepsilon)^{\sum_{j\in L\cup U}|\beta_j|}
\end{aligned}
$$

The second inequality is obtained by using $|\hat{A}_\alpha| \leq 1$ and ignoring some of the $(1-\varepsilon)$ terms. The final equality uses Parseval's Identity. We are now done, since the lemma is true for $J = L \cup U$. □

Lemma 13 *Let $S \subseteq [2] \times [k]$ be such that exactly one of V_S and W_S has odd cardinality. Then there exists a set $J \subseteq [k]$ such that*

$$T_S \leq \sum_{\{\beta_j\}_{j \in J}} |\hat{A}_{\Delta_{j \in J} \pi_j^{\oplus}(\beta_j)}| \left(\prod_{j \in J} \hat{B}^2_{j,\beta_j} (1-2\varepsilon)^{|\beta_j|} \right) \tag{11}$$

PROOF: Without loss of generality, assume $|V_S|$ is odd. The proof is similar to the proof of Lemma 12. Define L,R and U as in that proof. Analogous to (11), here we get

$$T_S = \sum_{\{\beta_j, j \in L \cup R \cup U : \Delta_{j \in R \cup U} \pi_j^{\oplus}(\beta_j) = \emptyset\}} \hat{A}_{\Delta_{j \in L \cup U} \pi_j^{\oplus}(\beta_j)} \left(\prod_{j \in L \cup R \cup U} \hat{B}^2_{j,\beta_j} \right) (1-2\varepsilon)^{\sum_{j \in L \cup U} |\beta_j| + \sum_{j \in R \cup U} |\beta_j|}$$

As in the proof of Lemma 12, the RHS can be bounded from above yielding:

$$T_S \leq \sum_{\{\beta_j, j \in L \cup U\}} \hat{A}_{\Delta_{j \in L \cup U} \pi_j^{\oplus}(\beta_j)} \left(\prod_{j \in L \cup U} \hat{B}^2_{j,\beta_j} \right) (1-2\varepsilon)^{\sum_{j \in L \cup U} |\beta_j|}.$$

Thus the lemma is true for $J = L \cup U = V_S$. (Analogously if W_S were odd, then the lemma would be true for $J = W_S$.) □

Finally we show that whenever the RHS of (8), (11) is bigger than some δ, then the decoding procedure will succeed with probability at least δ', where δ' does not depend on n and m.

Lemma 14 *Let A and $B_1, \ldots, B_k$ be such that $\hat{A}_{\emptyset} = 0$ and $\hat{B}_{i,\emptyset} = 0$ for every $i \in [k]$. Assume that for some $\delta, \varepsilon > 0$ and $J \subseteq [k]$ it holds that*

$$\sum_{\alpha, \{\beta_j\}_{j \in J} : \alpha = \Delta_j \pi_j^{\oplus}(\beta_j)} |\hat{A}_\alpha| \left(\prod_j \hat{B}^2_{j,\beta_j} (1-2\varepsilon)^{|\beta_j|} \right) \geq \delta. \tag{12}$$

Then

$$\mathop{\mathbf{E}}_{i \in [k]} \left[\sum_{\alpha, \beta : \pi_i(\beta) \cap \alpha \neq \emptyset} \frac{1}{|\alpha|} \hat{A}^2_\alpha \frac{1}{|\beta|} \hat{B}^2_{i,\beta} \right] \geq \frac{\delta^2}{4ck}$$

where $c = c_\varepsilon$ is a constant that depends on ε but not on δ nor on k.

PROOF: We start by deriving an upper bound on the LHS of (12). First we show that we can essentially ignore the contribution to this term from α such that $\hat{A}_\alpha$ is small.

$$\begin{aligned}
& \sum_{\alpha, \{\beta_j\}_{j \in J} : \alpha = \Delta_j \pi_j^{\oplus}(\beta_j) \text{and} |\hat{A}_\alpha| \leq \delta/2} |\hat{A}_\alpha| \left(\prod_j \hat{B}^2_{j,\beta_j} (1-2\varepsilon)^{|\beta_j|} \right) \\
\leq & \sum_{\alpha, \{\beta_j\}_{j \in J} : \alpha = \Delta_j \pi_j^{\oplus}(\beta_j) \text{and} \hat{A}_\alpha \leq \delta/2} \delta/2 \left(\prod_j \hat{B}^2_{j,\beta_j} (1-2\varepsilon)^{|\beta_j|} \right) \\
= & \; \delta/2,
\end{aligned}$$

where the final equality is another multi-fold application of Parseval's identity.

Combining the above with (12) we get:

$$\sum_{\alpha,\{\beta_j\}_{j\in J}:\alpha=\Delta_j\pi_j^{\oplus}(\beta_j)\text{and}|\hat{A}_\alpha|\geq\delta/2} |\hat{A}_\alpha| \left(\prod_j \hat{B}_{j,\beta_j}^2 (1-2\varepsilon)^{|\beta_j|}\right) \geq \delta/2 \tag{13}$$

We now upper the LHS of above:

$$\begin{aligned}
&\sum_{\alpha,\{\beta_j\}_{j\in J}:\alpha=\Delta_j\pi_j^{\oplus}(\beta_j)\text{and}|\hat{A}_\alpha|\geq\delta/2} |\hat{A}_\alpha| \left(\prod_j \hat{B}_{j,\beta_j}^2 (1-2\varepsilon)^{|\beta_j|}\right) \\
\leq\; & \frac{2}{\delta} \sum_{\alpha,\{\beta_j\}_{j\in J}:\alpha=\Delta_j\pi_j^{\oplus}(\beta_j)\text{and}|\hat{A}_\alpha|\geq\delta/2} |\hat{A}_\alpha|^2 \left(\prod_j \hat{B}_{j,\beta_j}^2 (1-2\varepsilon)^{|\beta_j|}\right) \text{ (Multiplying by } \tfrac{2}{\delta}|\hat{A}_\alpha| \geq 1.) \\
\leq\; & \frac{2}{\delta} \sum_{\alpha,\{\beta_j\}_{j\in J}:\alpha=\Delta_j\pi_j^{\oplus}(\beta_j)} |\hat{A}_\alpha|^2 \left(\prod_j \hat{B}_{j,\beta_j}^2 (1-2\varepsilon)^{|\beta_j|}\right) \\
\leq\; & \frac{2}{\delta} \sum_{\alpha,\{\beta_j\}_{j\in J}:\alpha=\Delta_j\pi_j^{\oplus}(\beta_j)} |\hat{A}_\alpha|^2 \left(\prod_j \hat{B}_{j,\beta_j}^2 (1-\varepsilon)^{2|\beta_j|}\right) \text{ (Since } (1-2\varepsilon) \leq (1-\varepsilon)^2) \\
\leq\; & \frac{2}{\delta} \sum_{\alpha,\{\beta_j\}_{j\in J}:\alpha=\Delta_j\pi_j^{\oplus}(\beta_j)} |\hat{A}_\alpha|^2 (1-\varepsilon)^{|\alpha|} \left(\prod_j \hat{B}_{j,\beta_j}^2 (1-\varepsilon)^{|\beta_j|}\right) \text{ (Since } |\alpha| \leq \textstyle\sum_j |\beta_j|.) \\
\leq\; & \frac{2}{\delta} \sum_{\alpha,\{\beta_j\}_{j\in J}:\exists i\in J \text{ s.t. } \alpha\cap\pi_i(\beta_i)\neq\emptyset} |\hat{A}_\alpha|^2 (1-\varepsilon)^{|\alpha|} \left(\prod_j \hat{B}_{j,\beta_j}^2 (1-\varepsilon)^{|\beta_j|}\right)
\end{aligned}$$

Thus combining the above inequality with (13) we get

$$\sum_{\alpha,\{\beta_j\}_{j\in J}:\exists i\in J \text{ s.t. } \alpha\cap\pi_i(\beta_i)\neq\emptyset} \hat{A}_\alpha^2 (1-\varepsilon)^{|\alpha|} \left(\prod_j \hat{B}_{j,\beta_j}^2 (1-\varepsilon)^{|\beta_j|}\right) \geq \frac{\delta^2}{4}. \tag{14}$$

From now an argument similar to that in the proof of Lemma 10 suffices to conclude the proof. The only difference is that we save a factor of approximately $2/k$, since we only have to pick one random $j \in J$ (rather than two distinct indices (i,j)). In particular we obtain the lemma with $c = 4e^2\varepsilon^2$. □

Once again we obtain the following corollary from Lemmas 12, 13 and 14 by observing that the expression being lower bounded in Lemma 14 is a lower bound on the success of the decoding procedure of Section 5.3.

Corollary 15 *Let $S \subseteq [2] \times [k]$ be such that at least one of V_S and W_S have odd cardinality. Then if $T_S \geq \delta$, then the decoding procedure of Section 5.3 leads to weak acceptance with probability at least $\frac{\delta}{ck}$, where c is a constant depending only on ε.*

Theorem 16 *For any k and ε, $\mathsf{Inner}_{k,\varepsilon}$ is a $((1-2\varepsilon)^{2k}, 2^{-2k}, 3k+2)$-good inner verifier with respect to (D_1, D_2) (the decoding procedure defined in Section 5.3.)*

PROOF: The verifier certainly makes $3k+2$ non-adaptive queries. If the input of the verifier satisfies the completeness conditions, then let A be the long code of a and B_i be the long code of b_i (we also have $\pi_i(b_i) = a$.) The test accepts if and only if $e_{i,j}(b_j) = 1$ for all $(i,j) \in [2] \times [k]$, an event that happens with probability $(1-2\varepsilon)^{2k}$.

Let now A, $B_1, \ldots, B_k$ and $\pi_1, \ldots, \pi_k$ be such that $\mathsf{Inner}_{k,\varepsilon}$ accepts with probability at least $2^{-2k} + \delta$. Then, from Proposition 8, there must be at least one non-empty $S \subseteq [2] \times [k]$ such that

$$\mathop{\mathbf{E}}_{f_1,f_2,g_1,\ldots,g_k,e_{1,1},\ldots,e_{2,k}} \left[\prod_{(i,j)\in S} A(f_i)B_j(g_j)B_j((f \circ \pi_j)g_j e_{i,j}) \right] \geq \delta.$$

Then, by Corollaries 11 and 15, we have that the decoding procedure of Section 5.3 succeeds with probability at least

$$\frac{1}{2}\frac{\delta^2}{ck^2} = \mathrm{poly}(\delta)$$

where c is a constant that depends only on ε. □

Theorem 1 follows from Theorem 7 and Theorem 16.

Acknowledgments

We thank Shafi Goldwasser and Amit Sahai for coming up with the model of PCPs that read two consecutive blocks of data. We thank Oded Goldreich for valuable comments.

References

[1] M. Ajtai, J. Komlós, and E. Szemerédi. An $O(n \log n)$ sorting network. In *Proceedings of the 15th ACM Symposium on Theory of Computing*, pages 1–9, 1983.

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

[3] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. *Journal of the Association for Computing Machinery*, 45(3):501–555, 1998. Preliminary version in *Proc. of FOCS'92.*

[4] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. *Journal of the Association for Computing Machinery*, 45(1):70–122, 1998. Preliminary version in *Proc. of FOCS'92.*

[5] L. Babai, L. Fortnow, L. Levin, and M. Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the 23rd ACM Symposium on Theory of Computing*, pages 21–31, 1991.

[6] L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991. Preliminary version in *Proc. of FOCS'90.*

[7] 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.*

[8] 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*.

[9] M. Bellare and M. Sudan. Improved non-approximability results. In *Proceedings of the 26th ACM Symposium on Theory of Computing*, pages 184–193, 1994.

[10] N. Creignou. A dichotomy theorem for maximum generalized satisfiability problems. *Journal of Computer and System Sciences*, 51(3):511–522, 1995.

[11] U. Feige. A threshold of $\ln n$ for approximating set cover. In *Proceedings of the 28th ACM Symposium on Theory of Computing*, pages 314–318, 1996.

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

[13] U. Feige and J. Kilian. Two prover protocols - low error at affordable rates. In *Proceedings of the 26th ACM Symposium on Theory of Computing*, pages 172–183, 1994.

[14] J. Håstad. Testing of the long code and hardness for clique. In *Proceedings of the 28th ACM Symposium on Theory of Computing*, pages 11–19, 1996.

[15] J. Håstad. Clique is hard to approximate within $n^{1-\varepsilon}$. In *Proceedings of the 37th IEEE Symposium on Foundations of Computer Science*, pages 627–636, 1996.

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

[17] R. Impagliazzo and D. Zuckerman. How to recycle random bits. In *Proceedings of the 30th IEEE Symposium on Foundations of Computer Science*, pages 248–253, 1989.

[18] S. Khanna, R. Motwani, M. Sudan, and U. Vazirani. On syntactic versus computational views of approximability. *SIAM Journal on Computing*, 28(1):164–191, 1999. Preliminary version in *Proc. of FOCS'94.*

[19] S. Khanna, M. Sudan, and D.P. Williamson. A complete classification of the approximability of maximization problems derived from boolean constraint satisfaction. In *Proceedings of the 29th ACM Symposium on Theory of Computing*, pages 11–20, 1997.

[20] 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.*

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

[22] M. Serna, L. Trevisan, and F. Xhafa. The parallel approximability of non-boolean constraint satisfaction and restricted integer linear programming. In *Proceedings of the 15th Symposium on Theoretical Aspects of Computer Science*, pages 488–498. LNCS 1373, Springer-Verlag, 1998.

[23] L. Trevisan. Approximating satisfiable satisfiability problems. In *Proceedings of the 5th European Symposium on Algorithms*, pages 472–485. LNCS 1284, Springer-Verlag, 1997.

[24] L. Trevisan. Parallel approximation algorithms by positive linear programming. *Algorithmica*, 21(1):72–88, 1998. Preliminary version in *Proc. of ESA'96.*

[25] L. Trevisan. Recycling queries in PCPs and in linearity tests. In *Proceedings of the 30th ACM Symposium on Theory of Computing*, 1998.

[26] L. Trevisan, G.B. Sorkin, M. Sudan, and D.P. Williamson. Gadgets, approximation, and linear programming. In *Proceedings of the 37th IEEE Symposium on Foundations of Computer Science*, pages 617–626, 1996.

[27] D. Zuckerman. On unapproximable versions of *NP*-complete problems. *SIAM Journal on Computing*, 25(6):1293–1304, 1996. Preliminary Version in *Proc. of Structures'93.*

[28] U. Zwick. Approximation algorithms for constraint satisfaction problems involving at most three variables per constraint. In *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms*, 1998.

[29] U. Zwick. Finding almost satisfying assignment. In *Proceedings of the 30th ACM Symposium on Theory of Computing*, 1998.