Uniform Direct Product Theorems: Simplified, Optimized, and Derandomized

[Extended Abstract]

Russell Impagliazzo [*]
UC San Diego
La Jolla, CA, USA
russell@cs.ucsd.edu

Ragesh Jaiswal [†]
UC San Diego
La Jolla, CA, USA
rjaiswal@cs.ucsd.edu

Valentine Kabanets
Simon Fraser University
Vancouver, BC, Canada
kabanets@cs.sfu.ca

Avi Wigderson
Institute of Advanced Studies
Princeton, NJ, USA
avi@ias.edu

ABSTRACT

The classical Direct-Product Theorem for circuits says that if a Boolean function $f : \{0,1\}^n \to \{0,1\}$ is somewhat hard to compute on average by small circuits, then the corresponding k-wise direct product function $f^k(x_1, \ldots, x_k) = (f(x_1), \ldots, f(x_k))$ (where each $x_i \in \{0,1\}^n$) is significantly harder to compute on average by slightly smaller circuits. We prove a *fully uniform* version of the Direct-Product Theorem with information-theoretically *optimal* parameters, up to constant factors. Namely, we show that for given k and ϵ, there is an efficient randomized algorithm A with the following property. Given a circuit C that computes f^k on at least ϵ fraction of inputs, the algorithm A outputs with probability at least $3/4$ a list of $O(1/\epsilon)$ circuits such that at least one of the circuits on the list computes f on more than $1-\delta$ fraction of inputs, for $\delta = O((\log 1/\epsilon)/k)$. Moreover, each output circuit is an AC^0 circuit (of size $\mathrm{poly}(n, k, \log 1/\delta, 1/\epsilon)$), with oracle access to the circuit C.

Using the Goldreich-Levin decoding algorithm [5], we also get a *fully uniform* version of Yao's XOR Lemma [18] with *optimal* parameters, up to constant factors. Our results simplify and improve those in [10].

Our main result may be viewed as an efficient approximate, local, list-decoding algorithm for direct-product codes (encoding a function by its values on all k-tuples) with optimal parameters. We generalize it to a family of "derandomized" direct-product codes, which we call *intersection codes*, where the encoding provides values of the function only on a subfamily of k-tuples. The quality of the decoding algorithm is then determined by sampling properties of the sets in this family and their intersections. As a direct consequence of this generalization we obtain the first derandomized direct product result in the uniform setting, allowing hardness amplification with only constant (as opposed to a factor of k) increase in the input length. Finally, this general setting naturally allows the decoding of concatenated codes, which further yields nearly optimal derandomized amplification.

Categories and Subject Descriptors

F.2.0 [**Analysis of Algorithms and Problem Complexity**]: General; G.3 [**Probability and Statistics**]: probabilistic algorithms

General Terms

Theory

Keywords

Direct Product Theorem, Direct Product Code, XOR Code

1. INTRODUCTION

Applications such as cryptography and derandomization require reliably hard problems, ones that cannot be solved by any fast algorithm with even a non-trivial advantage over random guessing. Direct-product theorems are a primary tool in hardness amplification, allowing one to convert problems that are somewhat hard into problems that are more reliably hard. In a direct-product theorem, we start with a function f such that any feasible algorithm has a non-negligible chance of failing to compute $f(x)$ given a random x. We then show that no feasible algorithm can, given multiple instances of the problem $x_1, \ldots, x_k$, compute all of the values $f(x_i)$, with even a small probability of success. (Usually, the x_i's are chosen independently, but there are also derandomized direct-product theorems where the x_i's are chosen pseudo-randomly.) Many strong direct product theorems are known for non-uniform models, such as Boolean

[*]The author is also currently a member of the School of Mathematics at the Institute of Advanced Studies, Princeton, NJ, USA. Research partially supported by NSF Grants 0716790 and 0515332. Views expressed here are not endorsed by the NSF.

[†]Research partially supported by NSF Grant 0716790. Views expressed here are not endorsed by the NSF.

STOC'08, May 17–20, 2008, Victoria, British Columbia, Canada.

circuits [18, 13, 6, 9, 12, 15]. Unfortunately, in general, direct-product theorems fail in completely uniform models such as probabilistic computation.

However, Trevisan [16] pointed out that proofs of direct product theorems correspond to (approximate) error-correction of sparse codes. Using this view, we think of a function f as being encoded by $Code(f) = f^k$, its values on all k-tuples. We seek a decoding algorithm which will generate efficient circuit(s) for f (on most inputs), given access to a circuit C' which is a highly corrupted codeword, agreeing with f^k only on an ϵ-fraction of all k-tuples. (Note that this code, like any code that can be computed locally with oracle access to f, has extremely poor distance. This precludes exact decoding, i.e., recovering a circuit that computes f on all inputs, but not approximate decoding.)

The strictly uniform direct-product theorem fails because these codes are not uniquely decodable. A circuit C' might agree on ϵ-fraction of k-tuples for each of $1/\epsilon$ different functions. Thus list decoding is essential, and one can quantify uniformity in terms of the list size. However, the non-uniform direct-product theorems yield list sizes which are all *exponential* in $1/\epsilon$. In contrast, a strong uniform direct-product theorems should have the list size which is polynomial in $1/\epsilon$. [10] gave the first such proof of the direct-product theorem. However, their reduction was quite complex and fell short of the information-theoretic bounds in many respects.

Here, we give a new uniform direct-product theorem that has the following features:

- **Optimality:** The parameters achieved by our list decoding algorithm are information theoretically optimal (to within constant factors).
- **Efficiency:** The decoding algorithm is simply a projection, namely implementable in uniform NC^0 with oracle access to the corrupted circuit C'. The circuits it produces are implementable in uniform AC^0. Thus, our hardness amplification applies to much simpler uniform classes than P.
- **Simplicity:** Both the decoding algorithm and the proof of correctness are extremely simple (even when compared with proofs in the non-uniform setting!).
- **Generality:** The decoding algorithm and its proof turns out to work without change for a general family of codes of which the above direct-product code is just an example. We define this class of *intersection codes*, which is simply specified by the family of k-subsets used to record values of f in $Code(f)$. We explain how the quality of the decoding (and thus of the amplification) depend on the sampling properties of the family of sets, and of their pairwise intersections.
- **Derandomization:** As an immediate bonus of the above setting we get the first derandomized direct-product theorems in the uniform setting. A direct application of the above intersection codes to subspaces yields amplification with input size $O(n)$, instead of the trivial bound of $O(kn)$ when using all subsets. In a more sophisticated application, using a concatenation of two intersection codes, we get similar savings in randomness, but with hardly any loss in other parameters.
- **Consequences:** As observed by [17, 16], efficient list-decoding has the same consequences as unique decoding in terms of hardness amplification within many natural complexity classes, e.g., NP, $\mathsf{P}^{\mathsf{NP}\|}$, $\#\mathsf{P}$, PSPACE and EXP.

1.1 Statement of the Uniform Direct-Product theorem

We say that a circuit C ϵ-computes a function F if $C(z) = F(z)$ for at least ϵ fraction of inputs z. A function F is $(1-\epsilon)$-hard for size $t(n)$ if no circuit of size $t(n)$ ϵ-computes F.

Following [17], we define the "semi-uniform" class $\mathsf{BPP}//\log$ as the class of probabilistic algorithms with advice of length $O(\log n)$ that depends on the random coin tosses of the algorithm, but not on the input. We can view such an algorithm as producing a polynomial-sized list of polynomial-size circuits: the algorithm then is judged by how well the best circuit on its list does. A probabilistic polynomial-time algorithm with advice, $A(x, r, z)$, ϵ-computes F if, for every length n, there is a function $z(r)$ taking a polynomial-size string r to a logarithmic length output, so that $\mathbf{Pr}_{x,r}[A(x, r, z(r)) = F(x)] \geq \epsilon$. A function F is $(1-\epsilon)$-hard for $\mathsf{BPP}//\log$ if no such algorithm and function $z(r)$ exist. For superpolynomial time complexity $t = t(n)$, we can generalize in the obvious way to the class $\mathsf{BPTIME}(\mathrm{poly}(t))//\log t$.

Given a Boolean function $f : \{0,1\}^n \to \{0,1\}$, the k-wise direct-product function f^k maps every k-set $(x_1, \ldots, x_k)$ of n-bit strings (ordered according to some fixed ordering of the universe $\{0,1\}^n$) to the k-tuple $(f(x_1), \ldots, f(x_k))$.[1] One of our main results is the following.

THEOREM 1.1 (UNIFORM DIRECT-PRODUCT THEOREM). *There is an absolute constant $c > 0$ so that for any functions $\delta = \delta(n)$, $k = k(n)$, $t = t(n)$, and $\epsilon = \epsilon(n) \geq e^{-\delta k/c}$ and $\epsilon > t^{-1/c}$, if f is δ-hard for $\mathsf{BPTIME}(poly(t(nk)))//\log t(nk)$, then f^k is $(1-\epsilon)$-hard for $\mathsf{BPTIME}(poly(t))//\log t$.*

The proof is via the following reconstruction algorithm, which is information-theoretically optimal up to constant factors.

THEOREM 1.2 (APPROXIMATE LIST-DECODING ALGORITHM). *There is a constant c and a probabilistic algorithm $\mathcal{A}$ with the following property. Let $k \in \mathbb{N}$, and $0 < \epsilon, \delta < 1$ be such that $\epsilon > e^{-\delta k/c}$. Let C' be a circuit that ϵ-computes the Direct-Product f^k, for some Boolean function $f : \{0,1\}^n \to \{0,1\}$. Given such a circuit C', algorithm $\mathcal{A}$ outputs with probability $\Omega(\epsilon)$ a circuit C that $(1-\delta)$-computes f. The algorithm $\mathcal{A}$ is a uniform randomized NC^0 algorithm (with one C'-oracle gate), and the produced circuit C is an AC^0 circuit of size* $\mathrm{poly}(n, k, \log 1/\delta, 1/\epsilon)$ *(with C'-oracle gates).*

The circuit output by algorithm $\mathcal{A}$ will have the following structure. Fix $s = k/2$. Let $A = (a_1, \ldots, a_s)$ be an s-subset of $\{0,1\}^n$, and let $v = (v_1, \ldots, v_s)$ be an s-bit string. For intuition, imagine that $v_i = f(a_i)$ for all $1 \leqslant i \leqslant s$.

We define the following randomized **circuit $\mathbf{C_{A,v}}$:**

"On input $x \in \{0,1\}^n$, check if $x = a_i$ for some $a_i \in A$; if so, then output v_i. Otherwise, repeatedly sample random k-sets B such that $A \cup \{x\} \subseteq B$, discarding any B where C' is inconsistent with our answers v for A (i.e., where $C'(B)|_A \neq v$). For the first consistent B, output $C'(B)|_x$. Produce

[1] The usual definition of k-wise direct product is in terms of k-tuples rather than k-sets, but it is easily seen to be equivalent, by randomly reordering the input tuple.

some default (error) output if no consistent B is found even after $100 \cdot (\ln 1/\delta)/\epsilon$ iterations."

Intuitively, if each $v_i = f(a_i)$, we are checking consistency on a subset of inputs to see whether we believe $C'(B)$ on another input. In addition to having the correct values, the algorithm $C_{A,v}$ requires that $C'(B)$ is correct for many B with $A \subset B$ (and thus, the algorithm usually finds a consistent B). Finally, we need that consistency implies correctness: for most B for which $C'(B)$ is consistent with v on A, it should be the case that $C'(B)$ is (almost) the same as $f|B$. We show that these three properties are strongly correlated, so that they simultaneously happen with good probability.

The main algorithm **algorithm** $\mathcal{A}$ is simply:

"Pick at random a k-set B_0, an s-subset $A \subseteq B_0$. Set $v = C'(B_0)|_A$. Output the circuit $C_{A,v}$."

1.2 Generalized direct-product encoding: intersection codes

Our analysis for direct products immediately generalize to decoding the following form of possibly derandomized direct product codes. Let $f : U \to R$, where U is some universe. Usually, U will be $\{0,1\}^n$, or $\mathbb{F}_q^m$, an m-dimensional vector space over a finite field $\mathbb{F}_q$. The range R is an arbitrary set ($R = \{0,1\}$ for Boolean f). Without loss of generality, we identify an s-tuple of elements of U with the s-subset of elements appearing in the tuple.

For $1 \leq s < k \in \mathbb{N}$, a *$k$-intersection code* is specified by two families of subsets of U, $\mathcal{T}$ a family of k-subsets of U, and $\mathcal{S}$, a family of s-subsets of U (with $s < k$). The family $\mathcal{S}$ is only used in the analysis. The encoding of f using $Code = Code(\mathcal{T}, \mathcal{S})$ is the restriction of the direct product f^k to sets $B \in \mathcal{T}$.

Our two running examples of these families are:

- **Independent**: $\mathcal{T}$ are all k-subsets of U, and $\mathcal{S}$ are all s-subsets of U; we only use the case $s = k/2$.
- **Subspaces**: We identify U with the vector space $\mathbb{F}_q^m$. For positive integers $d \geqslant 8$ and $r = d/2$, we take $\mathcal{T}$ to be all d-dimensional affine subspaces of U, and $\mathcal{S}$ to be all r-dimensional affine subspaces of U. Here we have $k = q^d$ and $s = q^r = \sqrt{k}$.

The Independent example is the k-wise direct-product function considered earlier. The Subspaces example will give us a derandomized version of the direct-product theorem, where inputs of f^k will be all points in a given affine d-dimensional subspace of U. Note that to specify $k = q^d$ such points, we only need to specify the $d+1$ vectors of U that define the d-dimensional affine subspace (d basis vectors plus a shift vector). In our case, d and r will be constants, and so these affine subspaces are specified with only $O(n)$ bits.

The code $Code$ is *δ-approximately (ϵ, ℓ)-list decodable* if for every function $C' : \mathcal{T} \to R^k$ there is a collection of at most ℓ functions $g_1, g_2, \cdots, g_\ell$ such that, for every function $f : U \to R$, if $Code(f)$ ϵ-agrees with C', then f will $(1-\delta)$-agree with some g_i, for $1 \leqslant i \leqslant \ell$. The code $Code$ is *efficiently locally decodable* if there is an efficient algorithm that uses oracle access to C' to generate circuits for the functions g_i's (which also use that oracle).

Our decoding algorithm for $Code(\mathcal{S}, \mathcal{T})$ is exactly the same as the algorithm $\mathcal{A}$ described in the previous section, with sets A coming from $\mathcal{S}$, and sets B from $\mathcal{T}$. We show that this algorithm $\mathcal{A}$ produces a good circuit for f, provided that families $\mathcal{S}, \mathcal{T}$ satisfy certain sampling conditions. In particular, we prove the following.

THEOREM 1.3. *Both Independent and Subspaces codes are efficiently, locally, δ-approximately $(\epsilon, O(1/\epsilon))$-list decodable, where*

- ***Independent:*** $\delta = O((\log 1/\epsilon)/k)$,
- ***Subspaces:*** $\delta = O(1/(\epsilon^2 k^{1/4}))$.

Moreover, the decoder for the Independent code is a uniform randomized NC^0 algorithm that outputs AC^0 circuits.

Informally, the only properties of $\mathcal{T}, \mathcal{S}$ we use are:

Computational assumptions: It is efficiently possible to: choose B uniformly in $\mathcal{T}$; given $B \in \mathcal{T}$, uniformly pick $A \in \mathcal{S}$ with $A \subset B$; given $A \in \mathcal{S}$ and $x \in U \setminus A$, uniformly pick $B \in \mathcal{T}$ with $A \cup \{x\} \subset B$.

Symmetry: For a fixed $B \in \mathcal{T}$, for a random $A \in \mathcal{S}$ with $A \subset B$, the elements of A are individually uniform over B. For a fixed $A \in \mathcal{S}$, and random $B \in \mathcal{T}$ with $A \subset B$, the elements in $B \setminus A$ are individually uniform over $U \setminus A$.

Sampling: For a fixed $B \in \mathcal{T}$ and any sufficiently large subset $W \subset B$, with high probability over a random $A \in \mathcal{S}, A \subset B$, $|A \cap W|/|A|$ is approximately the same as $|W|/|B|$. For a fixed $A \in \mathcal{S}$, and any sufficiently large subset $H \subset U \setminus A$, with high probability over a random $B \in \mathcal{T}, A \subset B$, we have that $|(B \setminus A) \cap H|/|B \setminus A|$ is approximately the same as $|H|/|U \setminus A|$.

1.3 Concatenated codes and hardness condensing

We also prove a stronger version of Theorem 1.3 for the case where we allow an oracle circuit C' for the direct-product f^k to be only *approximately* correct on at least ϵ fraction of inputs to f^k. More precisely, we allow a circuit C' such that, for at least ϵ fraction of $T \in \mathcal{T}$, $C'(T)$ and $f^k(T)$ agree on at least $(1-\delta')$ fraction of elements of T. Note that the usual version of direct-product decoding assumes $\delta' = 0$. Given such a circuit C', we show how to obtain a circuit C which $(1-\delta)$-computes f, for $\delta = O(\delta')$.

This relaxed notion of approximate list decoding can be formalized as follows. The code $Code$ is *(δ, δ')-approximately (ϵ, ℓ)-list decodable* if for every function $C' : \mathcal{T} \to R^k$ there is a collection of at most ℓ functions $g_1, g_2, \cdots, g_\ell$ such that, for every function $f : U \to R$, if the k-tuples $f^k(T)$ and $C'(T)$ $(1-\delta')$-agree on at least ϵ fraction of sets $T \in \mathcal{T}$, then f will $(1-\delta)$-agree with some g_i, for $1 \leqslant i \leqslant \ell$. *Efficient local decodability* means, as before, that a collection of circuits for such g_i's can be efficiently generated, given oracle access to a circuit C'.

We prove the following "approximate" version of Theorem 1.3.

THEOREM 1.4. *Both Independent and Subspaces codes are efficiently, locally, $(\delta, \Omega(\delta))$-approximately $(\epsilon, O(1/\epsilon))$-list decodable, where*

- ***Independent:*** $\delta = O((\log 1/\epsilon)/k)$,
- ***Subspaces:*** $\delta = O(1/(\epsilon^2 k^{1/4}))$.

While interesting in its own right, Theorem 1.4 will also allow us to obtain a strong derandomized version of uniform direct product theorem for a Boolean function $f : \{0,1\}^n \to$

$\{0,1\}$. The direct product code using affine subspaces already yields a harder function on inputs of size $O(n)$, but only with hardness polynomial in $1/k$. In non-uniform settings, there are derandomized direct product theorems with input size $O(n)$ and hardness exponentially small in n([12, 15]). We will be able to meet this goal partially: we define a function h of hardness $\epsilon = e^{-\Omega(\sqrt{n})}$ with input size $O(n)$ and $k = O(\log 1/\epsilon)$.

The function h combines the two direct product theorems. For $k = \sqrt{n}$ and a field $\mathbb{F}$ of size $q = 2^{\sqrt{n}}$, h is the restriction of f^k to k-subsets of inputs that all lie within a low dimensional affine subspace of $\mathbb{F}^{\sqrt{n}}$. We can specify the input to h by specifying a basis for the subspace with $O(n)$ bits, and then specifying $\sqrt{n}$ elements of the subspace in terms of this basis, using $O(\sqrt{n})$ bits each. We view h as a concatenation of two encodings. First, for $K = q^d = 2^{O(\sqrt{n})}$, we think of the K-direct product code for f using affine subspaces. This code, for each subspace, lists the value of f for all inputs in the subspace. This would be very large, so instead, we encode each block of the subspace code with the Independent k-direct product code, for $k = \sqrt{n}$, listing the values on subsets within each a affine subspace. To decode, we use the Independent direct product decoding within a given affine subspace as a subroutine in the affine subspace decoding procedure. Since the Independent direct product decoding is only approximate, we need Theorem 1.4 to handle errors created in the decoding of the inner code. The complete proof is deferred to the full paper.

THEOREM 1.5. (UNIFORM DERANDOMIZED DIRECT PRODUCT THEOREM) *There is an absolute constant $c > 0$ so that for any constant $0 < \delta < 1$, and any functions $t = t(n)$, $k = k(n)$, $\epsilon = \epsilon(n) \geq \max\{e^{-\delta k/c}, e^{-\Omega(\sqrt{n})}, t^{-1/c}\}$, and $K = K(n) = O(1/(\epsilon\delta)^8)$, if $f : \{0,1\}^n \to \{0,1\}$ is δ-hard for $\mathsf{BPTIME}(poly(t(cn))//\log t(cn)$, then the function h defined from f as described above is $(1-\epsilon)$-hard for $\mathsf{BPTIME}(poly(t))//\log t$. The input size of h is $O(n)$.*

We give an interpretation of Theorem 1.5 in terms of "hardness condensing" in the spirit of [3]. We obtain some form of "hardness condensing" with respect to $\mathsf{BPTIME}(t)//\log t$. For an affine subspace $B \in \mathcal{T}$, think of $g(B) = f|_B$ as the truth table of the Boolean function mapping $b \in B$ to $f(b)$. Since B is an affine d-dimensional subspace, each element of B can be described by a d-tuple of field elements $(\alpha_1, \ldots, \alpha_d) \in \mathbb{F}_q^d$, and so each $f|_B : \mathbb{F}_q^d \to \{0,1\}$ is a Boolean function on $d \log q$-size inputs. Also, each $B \in \mathcal{T}$ can be described with $(d+1)m \log q$ bits, and so each function in the function family $\{f|_B\}_{B \in \mathcal{T}}$ has a short description.

Consider the problem: Given (a description of) $B \in \mathcal{T}$, construct a circuit that computes $f|_B$ well on average. We show the following.

THEOREM 1.6 (HARDNESS CONDENSING). *There is an absolute constant $c > 0$, so that, if a function f is δ-hard for $\mathsf{BPTIME}(t)//\log t$, then every probabilistic $t^{1/c}$-time algorithm $\mathcal{C}$ has probability at most $\epsilon = \max\{q^{-d/16}, t^{-1/c}\}$ (over random $B \in \mathcal{T}$ and the internal randomness of $\mathcal{C}$) of producing a circuit that $(1 - \Omega(\delta))$-computes $f|_B$.*

Intuitively, for almost every B, the function $f|_B$ has almost the same hardness as f, but is defined on inputs of smaller size. Thus the reduction from f to f_B can be thought of as "hardness condensing".

Finally, we can convert our uniform direct product theorem into a uniform version of the Yao XOR Lemma [18]. While a qualitative version of this conversion follows immediately from [5], we obtain a quantitatively optimal version, up to constant factors. A uniform version of XOR Lemma is an approximate list decoding algorithm for a truncated version of the Hadamard code, and optimality is defined in terms of the information-theoretic coding properties of this code ([10]). For $f : \{0,1\}^n \to \{0,1\}$ and $k \in \mathbb{N}$, the k-XOR encoding of f is the function $f^{\oplus k}$ mapping each k-subset of n-bit strings $(x_1, \ldots, x_k)$ to the value $\oplus_{i=1}^k f(x_i)$.

THEOREM 1.7. *The k-XOR code is efficiently, locally, δ-approximately $(1/2 + \epsilon, O(1/\epsilon^2))$-list decodable, for $\delta = O((\log 1/\epsilon)/k)$.*

Again the proof is deferred to the full paper.

1.4 Relation to previous work

1.4.1 Non-uniform Direct Product Theorem

The classical Direct-Product Theorem (and closely related Yao's XOR Lemma [18]) for circuits has many proofs [13, 9, 6, 12]. The basic idea behind all these proofs is the following: If a given circuit C' ϵ-computes $f^k(x_1, \ldots, x_k)$, for some δ-hard function $f : \{0,1\}^n \to \{0,1\}$, with $\epsilon > (1-\delta)^k$, then it must be the case that the correctness of the answers of C' at some position i is *correlated* with the correctness of its answers in the remaining positions (since otherwise it would be the same as trying to compute $f(x_1), \ldots, f(x_k)$ independently sequentially, which obviously cannot be done with probability greater than $(1-\delta)^k$).

This correlation of C''s answers can be exploited in various ways to get a circuit $(1-\delta)$-computing f from the circuit C' (yielding different proofs of the direct-product theorem in [13, 9, 6, 12]). Usually, one takes a random k-tuple $(x_1, \ldots, x_k)$ containing a given input x in some position i, runs C' on that tuple, and checks how well C' did in positions other than i. To perform such a check, one obviously needs to know the true values of f at the inputs x_j for $j \neq i$; these are provided in the form of non-uniform advice in the circuit model. Then one decides on the guess for the value $f(x)$ based on the quality of C''s answers for x_j, $j \neq i$. For example, in [12], one flips a random coin with probability that is some function of the number of incorrect answers given by C' outside position i.

1.4.2 Uniform Direct Product Theorem, and decoding vs. testing

To get a uniform algorithm for f, we need to remove (or at least minimize the amount of) the non-uniform advice $f(x_j)$, $j \neq i$. The first result of that type was obtained in [10]. Their idea was to use the circuit C' itself in order to get enough labeled examples $(x, f(x))$, and then run the direct-product decoding algorithm of [12] on C' and the obtained examples.

To get sufficiently many examples, [10] use a method they called direct product amplification, which is to take an algorithm solving the k-wise direct product to one that (approximately) solves the k'-wise direct product problem with $k' \gg k$. This amplification is essentially equivalent to approximate list decoding when there are only k' possible instances in the domain of the function f. Their list-decoding

algorithm used one random "advice set" (where the algorithm produced correct answers) as a consistency check for another set that contains the instance to be solved. To be a meaningful consistency check, the advice set and instance-containing set need to have a large intersection. For independent random sets, this implies by the birthday-paradox bounds, that $k' \ll k^2$. Because of this constraint, [10] had to use direct-product amplification iteratively, to cover the whole domain size of 2^n instances. These iterations complicated the construction and made the parameters far from optimal.

We instead pick the instance-containing set *conditioned* on having a large intersection with the advice set. This can be done at one shot, on any domain size, so no iterations are needed.

This idea is similar in spirit to the *direct-product testing* methods used by [7, 4], and we were inspired by these papers. However, while they showed that this is sufficient in the unique decoding regime (where the algorithm is computing the direct product with high probability), we were surprised that this one idea sufficed in the list-decoding case as well. Our derandomized subspace construction was also inspired by [14, 1], who list-decode functions correlated to multivariable polynomials by using consistency checks on small dimensional subspaces.

While our results were inspired by similar results on direct-product testing, we have not found any formal connection between the testing and decoding problems. In particular, passing the consistency test with non-negligible probability is not sufficient to test non-negligible correlation with a direct-product function. It would be very interesting to find such a connection.

Remainder of the paper.

Section 2 contains some background facts, and basic sampling properties of graphs used in decoding of intersection codes. The analysis of our algorithm $\mathcal{A}$ is given in Section 3, where we state the conditions on the pair $(\mathcal{S}, \mathcal{T})$ that are sufficient for $\mathcal{A}$ to produce a good circuit $C_{A,v}$. Section 4 contains sketches of proofs of Theorems 1.4, 1.5, 1.6, and 1.7. Section 5 contains concluding remarks and open questions.

2. PRELIMINARIES

2.1 Concentration bounds

The standard form of the Hoeffding bound [8] says that, for any finite subset F of measure α in some universe $\mathcal{U}$, a random subset R of size t is very likely to contain close to αt points from F. The following is a natural generalization for the case where F is any $[0,1]$-valued function over $\mathcal{U}$.

LEMMA 2.1 (HOEFFDING [8]). *Let $F : \mathcal{U} \to [0,1]$ be any function over a finite universe $\mathcal{U}$ with the expectation $\mathbf{Exp}_{x\in\mathcal{U}}[F(x)] = \alpha$, for any $0 \leqslant \alpha \leqslant 1$. Let $R \subseteq \mathcal{U}$ be a random subset of size t. Define a random variable $X = \sum_{x\in R} F(x)$. Then the expectation of X is $\mu = \alpha t$, and for any $0 < \gamma \leqslant 1$, $\mathbf{Pr}\left[|X-\mu| \geqslant \gamma\mu\right] \leqslant 2 \cdot e^{-\gamma^2\mu/3}$.*

LEMMA 2.2. *Let $X_1, \ldots, X_t$ be random variables taking values in the interval $[0,1]$, with expectations μ_i, $1 \leqslant i \leqslant t$. Let $X = \sum_{i=1}^{t} X_i$, and let $\mu = \sum_{i=1}^{t} \mu_i$ be the expectation of X. For any $0 < \gamma \leqslant 1$, we have the following:*

- ***[Chernoff-Hoeffding]*** *If $X_1, \ldots, X_t$ are independent, then $\mathbf{Pr}[|X-\mu| \geqslant \gamma\mu] \leqslant 2 \cdot e^{-\gamma^2\mu/3}$.*
- ***[Chebyshev]*** *If $X_1, \ldots, X_t$ are pairwise independent, then $\mathbf{Pr}[|X-\mu| \geqslant \gamma\mu] \leqslant 1/(\gamma^2\mu)$.*

2.2 Pairwise independence of subspaces

Let $U = \mathbb{F}_q^m$ be an m-dimensional linear space over a finite field $\mathbb{F}_q$. An *affine* d-dimensional subspace A of U is specified by a collection of d linearly independent vectors $a_1, \ldots, a_d \in U$ and an arbitrary vector $b \in U$ so that $A = \{b + \sum_{i=1}^{d} \alpha_i a_i \mid \alpha_i \in \mathbb{F}_q, 1 \leqslant i \leqslant d\}$. Thus the elements of A are in one-to-one correspondence with d-tuples of scalars $(\alpha_1, \ldots, \alpha_d)$.

We will use the following easy facts.

CLAIM 2.3. *A sequence of all q^d elements of a randomly chosen d-dimensional affine subspace of U are pairwise independent and uniform over U.*

CLAIM 2.4. *For $t = (q^d-1)/(q-1)$, let $\bar{\alpha}_1, \ldots, \bar{\alpha}_t \in \mathbb{F}_q^d$ be pairwise linearly independent vectors. Let A be a random d-dimensional linear subspace of U. Then the t vectors of A that correspond to $\bar{\alpha}_1, \ldots, \bar{\alpha}_t$ are pairwise independent and uniform over U.*

2.3 Graphs

We will consider bipartite graphs $G = G(L, R)$ defined on a bipartition $L \cup R$ of vertices; we think of L as left vertices, and R as right vertices of the graph G. For a vertex v of G, we denote by $N_G(v)$ the set of its neighbors in G; if the graph G is clear from the context, we will drop the subscript and simply write $N(v)$. We say that G is *bi-regular* if the degrees of vertices in L are the same, and the degrees of vertices in R are the same.

2.3.1 Auxiliary graphs for $(\mathcal{S}, \mathcal{T})$-codes

The following three graphs will be useful for the analysis of our intersection codes. Let U be any finite set. Let $\mathcal{T}$ be a family of k-subsets of U, and let $\mathcal{S}$ be a family of s-subsets of U, for some $s < k$.

DEFINITION 2.5 (INCLUSION GRAPH). *The* inclusion graph *$I(\mathcal{S}, \mathcal{T})$ is the bipartite graph that has an edge (A, B) for every $A \in \mathcal{S}$ and $B \in \mathcal{T}$ such that $A \subseteq B$.*

The inclusion graph $I(\mathcal{S}, \mathcal{T})$ is called *transitive* if, for every $B, B' \in \mathcal{T}$, there is a permutation π of U which moves B to B' and induces an isomorphism of I, and similarly, for every $A, A' \in \mathcal{S}$, there is a permutation σ of U which moves A to A' and induces an isomorphism of I.

DEFINITION 2.6 ($\mathcal{S}$-GRAPH). *For every $B \in \mathcal{T}$, the $\mathcal{S}$-graph $H(B, N_I(B))$ is the bipartite graph that has an edge (x, A) for every $x \in B$ and $A \in N_I(B)$ such that $x \in A$.*

DEFINITION 2.7 ($\mathcal{T}$-GRAPH). *For every $A \in \mathcal{S}$, the $\mathcal{T}$-graph $G(U \setminus A, N_I(A))$ is the bipartite graph that has an edge (x, B) for every $x \in U \setminus A$ and $B \in N_I(A)$ such that $x \in B \setminus A$.*

Note that if $I(\mathcal{S}, \mathcal{T})$ is transitive, then the structure of the $\mathcal{S}$-graph $H(B, N(B))$ is independent of the choice of B, and similarly, the structure of the $\mathcal{T}$-graph $G(U \setminus A, N(A))$

is independent of the choice of A. This will simplify the analysis of the properties of these graphs. One can easily check that the inclusion graph I for both of our running examples of families $(\mathcal{S}, \mathcal{T})$, Independent and Subspaces, is transitive.

2.3.2 Samplers

Let $G = G(L, R)$ be any bi-regular bipartite graph. For a function $\lambda : [0,1] \to [0,1]$, we say that G is a $(\mu, \lambda(\mu))$-*sampler* if, for every function $F : L \to [0,1]$ with the average value $\mu \stackrel{\text{def}}{=} \mathbf{Exp}_{x \in L}[F(x)]$, there are at most $\lambda(\mu) \cdot |R|$ vertices $r \in R$ where

$$\left| \mathbf{Exp}_{y \in N(r)}[F(y)] - \mu \right| \geqslant \mu/2.$$

Note that the case of a Boolean function $F : L \to \{0,1\}$ with the average μ corresponds to the property that all but $\lambda(\mu)$ fraction of nodes $r \in R$ have close to the expected number of neighbors in the set $\{x \mid F(x) = 1\}$ of measure μ. The sampler defined above is a natural generalization to the case of $[0,1]$-valued F; it is also a special case of an *oblivious approximator* [2] or *approximating disperser* [19].

For the analysis of intersection codes $Code(\mathcal{S}, \mathcal{T})$ based on families $\mathcal{S}$ and $\mathcal{T}$, we will need that the corresponding $\mathcal{S}$-graphs and $\mathcal{T}$-graphs be samplers. We show that this is true for both of our running examples. Since both our inclusion graphs (for Independent and Subspaces cases) are transitive, the structure of the $\mathcal{S}$-graphs and $\mathcal{T}$-graphs is independent of the choices of $B \in \mathcal{T}$ and $A \in \mathcal{S}$, respectively.

LEMMA 2.8. *For both Independent and Subspaces families $(\mathcal{S}, \mathcal{T})$, the $\mathcal{S}$-graph H is $(\alpha, \nu(\alpha))$-sampler, where*

- ***Independent:*** $\nu(\alpha) = 2 \cdot e^{-\alpha k/24}$,
- ***Subspaces:*** $\nu(\alpha) = 4/(\alpha\sqrt{k})$.

PROOF. For Independent, we use the Hoeffding bound of Lemma 2.1. For Subspaces, we use the fact that points in a random affine subspace of a given affine space are uniformly distributed and pairwise independent (cf. Claim 2.3), and then apply Chebyshev's bound of Lemma 2.2. □

The proof of the following lemma is given in the full version of the paper.

LEMMA 2.9. *For both Independent and Subspaces families $(\mathcal{S}, \mathcal{T})$, the $\mathcal{T}$-graph G is $(\beta, \lambda(\beta))$-sampler, where*

- ***Independent:*** $\lambda(\beta) = 2 \cdot e^{-\beta k/24}$,
- ***Subspaces:*** $\lambda(\beta) = 4q^2/(\beta\sqrt{k})$.

2.3.3 Properties of samplers and their subgraphs

Here we prove two properties of samplers, which will be useful for the analysis of our decoding algorithm. These properties basically show that samplers are "robust" to deletions of vertices.

The first property says that for any two large vertex subsets W and F of a sampler, the fraction of edges between W and F is close to the product of the densities of W and F.

LEMMA 2.10. *Suppose $G = G(L, R)$ is a (β, λ)-sampler. Let $W \subseteq R$ be any set of measure ρ, and let $F \subseteq L$ be any set of measure β. Then we have*

$$\mathbf{Pr}_{x \in L, y \in N(x)}[x \in F \ \&\ y \in W] \geqslant \beta(\rho - \lambda)/2.$$

PROOF. We need to estimate the probability of picking an edge between F and W in a random experiment where we first choose a random $x \in L$ and then its random neighbor y. Since the graph G is assumed to be bi-regular, this probability remains the same in the experiment where we first pick a random $y \in R$ and its random neighbor $x \in N(y)$. The latter is easy to estimate using the sampling property of the graph G, as we show next.

Let $F' \subseteq F$ be of density exactly β. Let $W' \subseteq W$ be the subset of vertices that have at least $\beta/2$ fraction of their neighbors in F. Since G is a (β, λ)-sampler and W is of measure ρ, we get that W' is of measure at least $\rho - \lambda$. Then conditioned on picking a vertex $y \in W'$, the probability that its random neighbor is in F is at least $\beta/2$. The lemma follows. □

The second property deals with edge-colored samplers. Suppose that all edges in a bi-regular graph $G = G(L, R)$ are colored with two colors, red and green, so that the number of red edges is at most t, for some $t \geqslant 0$. Since G is bi-regular, picking a random vertex $x \in L$ and its random incident edge is the same as picking a random $y \in R$ and its random incident edge, and clearly, the probability of getting a red edge in both cases is $t/|E|$, where E is the edge set of G. Now suppose that we are given a subgraph G' obtained from G by removing some vertices from R (and all the edges incident upon the removed vertices). Let $W \subseteq R$ be a subset of the remaining vertices in G', and suppose that G' has at most t red edges. Since G' is still right-regular (i.e., all vertices $w \in W$ have the same degree), sampling a random incident edge of a random vertex $w \in W$ still yields a red edge with probability at most $t/|E'|$, where E' is the edge set of G'. For general graphs G, we can't say that the probability of getting a red edge remains the same when we pick a random incident edge of a random vertex $x \in L$ (since G' may not be bi-regular). However, we prove that this is approximately true when G is a sampler.

LEMMA 2.11. *Suppose $G = G(L, R)$ is a (β, λ)-sampler, with the right degree D. Let $W \subseteq R$ be any subset of density ρ, and let $G' = G(L, W)$ be the induced subgraph of G (obtained after removing all vertices in $R \setminus W$), with the edge set E'. Let $Col : E' \to \{red, green\}$ be any coloring of the edges of G' such that at most $\alpha D|W|$ edges are colored red, for some $0 \leqslant \alpha \leqslant 1$. Then*

$$\mathbf{Pr}_{x \in L, y \in N_{G'}(x)}[Col(\{x, y\}) = red] \leqslant \max\{2\alpha/(1 - \lambda/\rho), \beta\}.$$

PROOF. We need to estimate the probability of picking a red edge in G' when we first pick a random $x \in L$ and then pick its random neighbor y in G'. For every $x \in L$, let d_x be the degree of x in G', and let $\xi(x)$ be the fraction of red edges incident to x in G'. The probability we want to estimate is exactly $\mu = \mathbf{Exp}_{x \in L}[\xi(x)]$. If $\mu \leqslant \beta$, then we are done. So for the rest of the proof, we will suppose that $\mu > \beta$.

Let $W' \subseteq W$ be the subset of those vertices w where $\mathbf{Exp}_{x \in N(w)}[\xi(x)] \geqslant \mu/2$. (Here we use $N(w)$ to denote the neighborhood $N_{G'}(w)$ of w in G', which is the same as $N_G(w)$ by the definition of G'.) Since G is a (β, λ)-sampler and W has measure ρ in G, we get that W' has measure at least $\rho - \lambda$ in G, and hence measure $1 - \lambda/\rho$ in G'. Hence,

we have

$$\sum_{y\in W}\mathbf{Exp}_{x\in N(y)}[\xi(x)] \geqslant \sum_{y\in W'}\mathbf{Exp}_{x\in N(y)}[\xi(x)] \geqslant |W|(1-\lambda/\rho)\mu/2. \quad (1)$$

On the other hand, $\sum_{y\in W}\left(D\cdot\mathbf{Exp}_{x\in N(y)}[\xi(x)]\right)$ is simply the summation over all edges (x,y) in G' where each edge (x,y) with $x\in L$ contributes $\xi(x)$ to the sum. Since the degree of each x is d_x, each $x\in L$ contributes exactly $d_x\xi(x)$, which is the number of incident red edges at x. Hence, the total sum is exactly the number of red edges in G', which is at most $\alpha D|W|$ by assumption. It follows that

$$\sum_{y\in W}\mathbf{Exp}_{x\in N(y)}[\xi(x)] = (1/D)\sum_{x\in L}d_x\xi(x) \leqslant |W|\alpha. \quad (2)$$

Finally, comparing the bounds in Eqs. (1) and (2), we conclude that $\mu\leqslant 2\alpha/(1-\lambda/\rho)$. □

3. DECODING INTERSECTION CODES

Let $(\mathcal{S},\mathcal{T})$ be a pair of families of subsets of U, and let $Code(\mathcal{S},\mathcal{T})$ be the intersection code defined for these families. Fix a function $f:U\to R$. Let C' be a circuit that ϵ-computes $Code(f)$. We will show how to compute from C' a deterministic circuit C that $(1-\delta)$-computes f, for $\delta>0$ being the parameter that depends on ϵ and $(\mathcal{S},\mathcal{T})$.

Our decoding algorithm $\mathcal{A}$ for $Code(\mathcal{S},\mathcal{T})$ can be defined in terms of the inclusion and $\mathcal{T}$-graphs. Fix any edge (A,B) of the inclusion graph $I(\mathcal{S},\mathcal{T})$. Let $v=C'(B)|_A$ be the values that the circuit $C'(B)$ gives for the elements in A.

Let $G=G(U\setminus A,N(A))$ be the $\mathcal{T}$-graph for A. Let $Cons\subseteq N(A)$ be the subset of those $B'\in N(A)$ for which $C'(B')|_A=v$. We will say that such sets B' are *consistent with* B.

Define the **circuit $\mathbf{C_{A,v}}$**:

"On input $x\in U$, if $x\in A$, then output the corresponding value v_x. Otherwise, repeatedly sample random neighbors B' of x in the $\mathcal{T}$-graph G, discarding any $B'\notin Cons$, until the first $B'\in Cons$ is obtained. For this $B'\in Cons$, output the value $C'(B')|_x$. Produce the default (error) output if no $B'\in Cons$ is found even after $O((\ln 1/\delta)/\epsilon)$ steps."

Define the **decoding algorithm $\mathcal{A}$**:

"On an input circuit C', pick a random edge (A,B) of the inclusion graph $I(\mathcal{S},\mathcal{T})$, set $v=C'(B)|_A$, and output the circuit $C_{A,v}$."

REMARK 3.1. *For the described algorithm $C_{A,v}$ to be efficient, we need an efficient procedure for sampling random neighbors of a given left vertex in the $\mathcal{T}$-graph $G(U\setminus A,N(A))$. For both of our running examples, one can easily argue that such efficient sampling is possible.*

We now state the main technical result of our paper: the conditions on $(\mathcal{S},\mathcal{T})$ under which the decoding algorithm $\mathcal{A}$ produces a good circuit $C_{A,v}$. For the rest of this section, we set $\epsilon'=\epsilon/2$.

THEOREM 3.2. *Suppose that the inclusion graph $I(\mathcal{S},\mathcal{T})$ is transitive (and hence also bi-regular), the $\mathcal{S}$-graph H is a $(\mu,\delta\epsilon'^2/(128\mu))$-sampler for every $\mu>\delta/64$, and the $\mathcal{T}$-graph G is a $(\delta/16,\epsilon'/2)$-sampler. Then the algorithm $\mathcal{A}$ produces with probability $\epsilon'/2$ a randomized circuit $C_{A,v}$ satisfying*

$$\mathbf{Pr}[C_{A,v}\ \textit{computes}\ f]\geqslant 1-\delta/4,$$

where the probability is over the inputs and the internal randomness of $C_{A,v}$.

REMARK 3.3. *Note that if a randomized circuit $C_{A,v}$ satisfies the conclusion of Theorem 3.2, then by randomly fixing its internal randomness we get (with probability at least 3/4) a* deterministic *circuit C that $(1-\delta)$-computes f.*

We postpone the proof of Theorem 3.2, and use it to prove Theorem 1.3.

PROOF. (Proof of Theorem 1.3) For Independent, we get by Lemmas 2.8 and 2.9 that both H and G are $(\mu,\lambda(\mu))$-samplers for $\lambda(\mu)\leqslant e^{-\Omega(\mu k)}$. For $\mu>\delta/64$, write $\mu=c\delta$ where $c=\mu/\delta>1/64$. For the graph H, we get that $\mu\cdot\lambda(\mu)\leqslant c\delta e^{-\Omega(c\delta k)}$. For $\delta=d\log(1/\epsilon)/k$ for large enough constant d, we get $e^{-\Omega(cd\log 1/\epsilon)}=\epsilon^{\Omega(cd)}\leqslant\epsilon'^2\epsilon^{cd'}$, for some large constant d' dependent on d. Assume that $\epsilon<0.9$ (if a circuit C' ϵ-computes f^k for $\epsilon\geqslant 0.9$, it obviously 0.9-computes f^k). Choosing sufficiently large constant d, we can ensure that $\epsilon^{cd'}<2^{-c}/128$, and so $c\delta e^{-\Omega(c\delta k)}\leqslant c\delta\epsilon'^2 2^{-c}/128\leqslant\delta\epsilon'^2/128$. Thus H satisfies the assumptions of Theorem 3.2. Setting $\delta=d(\log 1/\epsilon)/k$ for a large enough $d\in\mathbb{N}$ will also make the $\mathcal{T}$-graph G satisfy the assumptions of of Theorem 3.2.

For Subspaces, Lemma 2.8 gives us that H is $(\mu,\lambda(\mu))$-sampler for $\lambda(\mu)=4/(\mu\sqrt{k})$. Hence, $\mu\cdot\lambda(\mu)\leqslant 4/\sqrt{k}$. The latter is at most $\delta\epsilon'^2/128$ for $\delta\geqslant 512/\epsilon'^2\sqrt{k}$. Lemma 2.9 says that the graph G is $(\delta/16,\epsilon'/2)$-sampler for $\delta\geqslant 128q^2/(\epsilon'\sqrt{k})$. Thus, to satisfy the conditions of Theorem 3.2, we can set $\delta\leqslant 512q^2/(\epsilon'^2\sqrt{k}))$, which is $O(1/(\epsilon'^2k^{1/4}))$ for $q\leqslant k^{1/8}$.

By Remark 3.3, we get in both cases a required deterministic circuit $(1-\delta)$-computing f. □

3.1 Why $C_{A,v}$ works

Here we describe the conditions on our auxiliary graphs (inclusion, $\mathcal{S}$- and $\mathcal{T}$-graphs) and an edge (A,B) of the inclusion graph, which are sufficient for the circuit $C_{A,v}$ described above to satisfy the conclusion of Theorem 3.2. Intuitively, we are using (A,v) as a consistency check to see whether to believe $C'(B')$. To be useful as a consistency check, we should have:

- $v=f(A)$, so if $C'(B')$ is correct, it will always be consistent with v on A.
- There are many B' for A where $C'(B')$ is correct.
- On average over B' where $C'(B')$ is consistent with A, $C'(B')$ is correct for most $x\in B'\setminus A$.

We show that these conditions suffice, and that many such sets A exist.

We need the following definitions. For a set $B\in\mathcal{T}$, let $Err(B)$ denote the subset of those x's in B where $C'(B)$ disagrees with $f^k(B)$, and let $err(B)=|Err(B)|/|B|$. A set $B\in\mathcal{T}$ is called *correct* if $err(B)=0$. A set $B\in\mathcal{T}$ is called *α-incorrect* if $err(B)\geqslant\alpha$. For the inclusion graph $I(\mathcal{S},\mathcal{T})$, we call an edge (A,B) *correct* if B is correct. As before, we set $\epsilon'=\epsilon/2$. Call an edge (A,B) *good* if it is

correct and at least ϵ'-fraction of all edges (A, B') incident to A are correct. An edge (A, B) of the inclusion graph is called α-*excellent* if it is *good*, and moreover,

$$\mathbf{Exp}_{B' \in Cons}[err(B')] \leqslant \alpha,$$

where the expectation is over uniformly random B' that are consistent with B.

In words, for an excellent edge (A, B), we have at least ϵ' of correct edges (A, B') (and so these $B' \in Cons$), and at the same time, the average fraction of errors in the neighbors of A that are consistent with B is less than α. So, conditioned on sampling a random $B' \in Cons$, we expect to get a B' such that $C'(B')|_x = f(x)$ for most $x \in B'$.

Our circuit $C_{A,v}$ is defined so that it only considers random $B' \in Cons$. This circuit will agree with f well on average, assuming that A, v came from some excellent edge (A, B), and assuming that the $\mathcal{T}$-graph is a sampler.

LEMMA 3.4. *Let an edge (A, B) of the inclusion graph I be α-excellent, and let the $\mathcal{T}$-graph $G(U \setminus A, N(A))$ be a (β, λ)-sampler. Suppose that $\lambda \leqslant \epsilon'/2$, $\alpha \leqslant \beta/2$, and $\beta \leqslant \delta/16$. Then $\mathbf{Pr}[C_{A,v}$ computes $f] \geqslant 1 - \delta/4$, where the probability is over uniform x's and the internal randomness of $C_{A,v}$.*

To prove Lemma 3.4, we consider two cases. First we consider the set $F \subseteq U \setminus A$ of x's that have too few edges (x, B') with $B' \in Cons$ in the $\mathcal{T}$-graph $G(U \setminus A, N(A))$. These are the x's for which $C_{A,v}$ is unlikely to produce any answer and hence fails. Secondly, we bound the average conditional probability of $C_{A,v}$ producing an incorrect answer given that the circuit produces some answer. Note that for every $x \in U \setminus A$ this conditional probability is the same for all sampling steps of $C_{A,v}$. So, we can just analyze this conditional probability for one sampling step.

First, we bound the size of F.

LEMMA 3.5. *Suppose an edge (A, B) of I is good, and the $\mathcal{T}$-graph $G(U \setminus A, N(A))$ is a (β, λ)-sampler. Let F be the subset of $U \setminus A$ with less than μ fraction of their edges into $Cons$, where $\mu = (\epsilon' - \lambda)/2$. Then the measure of F is at most β.*

PROOF. Suppose that F has density at least β. Let $F' \subseteq F$ be of density exactly β. By the assumption of the lemma, we have that $\mathbf{Pr}_{x \in U \setminus A, y \in N(x)}[x \in F' \ \& \ y \in Cons] < \beta\mu = \beta(\epsilon' - \lambda)/2$.

On the other hand, we know that $Cons$ has density at least ϵ' (by the definition of goodness of (A, B)). By Lemma 2.10, the fraction of edges in G that go between F and $Cons$ is at least $\beta(\epsilon' - \lambda)/2$, which contradicts our earlier upper bound. □

For a given $x \in U \setminus A$, let $h(x)$ denote the conditional probability that $C_{A,v}$ produces an incorrect answer, given that it produces some answer. We will show that the expectation $\mathbf{Exp}_{x \in U \setminus A}[h(x)]$ is small.

LEMMA 3.6. *Suppose (A, B) is α-excellent, and the $\mathcal{T}$-graph G is a (β, λ)-sampler. Further suppose that $\alpha \leqslant \beta/2$ and $\lambda \leqslant \epsilon'/2$. Then $\mathbf{Exp}_{x \in U \setminus A}[h(x)] \leqslant \beta$.*

PROOF. Since $C_{A,v}$ produces an answer on a given input x only if it samples a consistent neighbor B' of x in the $\mathcal{T}$-graph $G(U \setminus A, N(A))$, we can view $h(x)$ as follows. Let $G' = G(U \setminus A, Cons)$ be the induced subgraph of G where we remove all inconsistent vertices from $N(A)$. For each edge (x, B') of G', we color it red if $x \in Err(B')$, and color it green otherwise. Then $h(x)$ is the fraction of red edges incident to x in the graph G'.

Let ρ be the measure of $Cons$ in G. We know that $\rho \geqslant \epsilon'$. Let $D = |B|$ be the right degree of the $\mathcal{T}$-graph G (and hence also of G'). The total number of red edges in G' is at most $\alpha D|Cons|$, by the definition of α-excellence.

By Lemma 2.11, we conclude that $\mathbf{Pr}_{x \in U \setminus A, B' \in N_{G'}(x)}[x \in Err(B')] \leqslant \max\{2\alpha/(1 - \lambda/\rho), \beta\}$. By assumptions, $1 - \lambda/\rho \geqslant 1 - \lambda/\epsilon' \geqslant 1/2$, and so $\alpha/(1 - \lambda/\epsilon') \leqslant 2\alpha \leqslant \beta$. □

Now we can finish the proof of Lemma 3.4.

PROOF. (Proof of Lemma 3.4) Lemma 3.5 implies for every $x \in U \setminus (A \cup F)$, where F is of measure at most β, there are at least $\epsilon'/4$ fraction of edges into $Cons$. Hence the probability of $C_{A,v}$ not producing any answer in $t = d(\log 1/\delta)/\epsilon'$ sampling steps for such an x is at most $\delta/8$ for some constant d, e.g., $d = 100$. For each such x, the probability that $C_{A,v}$ is wrong, given that $C_{A,v}$ produces an answer, is $h(x)$. Hence, the overall probability (over random x and internal randomness) that $C_{A,v}$ is wrong is at most $\beta + \delta/8 + \mathbf{Exp}_{x \in U \setminus A}[h(x)]$. By Lemma 3.6, the last summand is at most β, and so the total is at most $2\beta + \delta/8 \leqslant \delta/4$ (since $\beta \leqslant \delta/16$). □

3.2 Choosing an excellent edge (A, B)

Here we show that if the inclusion graph I is bi-regular and if the $\mathcal{S}$-graph H is a sampler, then a random edge (A, B) of I will be excellent with probability $\Omega(\epsilon)$.

LEMMA 3.7. *Suppose the inclusion graph I is bi-regular, and the $\mathcal{S}$-graph H is an $(\mu, \nu(\mu))$-sampler[2]. Moreover, assume that $0 \leqslant \alpha \leqslant 1$ is such that, for every $\alpha/2 < \mu \leqslant 1$, we have $\mu \cdot \nu(\mu) \leqslant \alpha\epsilon'^2/4$. Then a random edge (A, B) of I is α-excellent with probability at least $\epsilon'/2$.*

First, we argue the following.

LEMMA 3.8. *A random edge (A, B) of a bi-regular inclusion graph I is good with probability at least ϵ'.*

PROOF. Choosing a random edge (A, B) of the inclusion graph I is equivalent to choosing a random $B \in \mathcal{T}$ and then choosing a random $A \in N(B)$. By the assumption on C', a random $B \in \mathcal{T}$ is correct with probability at least ϵ. Thus we have $\mathbf{Pr}_{A \in \mathcal{S}, B \in N(A)}[(A, B) \text{ is correct}] \geqslant \epsilon$.

For $A \in \mathcal{S}$, let $P(A)$ be the event (over a random choice of $A \in \mathcal{S}$) that $\mathbf{Pr}_{B' \in N(A)}[B' \text{ is correct}] < \epsilon/2$. Observe that, conditioned on $A \in \mathcal{S}$ such that $P(A)$, we get

$$\mathbf{Pr}_{A \in \mathcal{S}, B \in N(A)}[(A, B) \text{ is correct} \mid P(A)] < \epsilon/2,$$

and so,

$$\mathbf{Pr}_{A \in \mathcal{S}, B \in N(A)}[((A, B) \text{ is correct}) \ \& \ P(A)] < \epsilon/2.$$

Finally, the probability that a random edge (A, B) is good is equal to

$$(\mathbf{Pr}_{A,B}[(A, B) \text{ is correct}] - \mathbf{Pr}_{A,B}[((A, B) \text{ is correct}) \ \& \ P(A)]) > \epsilon - \epsilon/2 = \epsilon/2,$$

which is equal to ϵ', as required. □

[2]Here we only need that, for any measure μ subset F of left vertices of H, the fraction of right vertices with no incident edges into F is at most ν.

Now we can prove Lemma 3.7.

PROOF. (Proof of Lemma 3.7) To show that an edge (A, B) is α-excellent, it suffices to argue that

$$\sum_{B' \in Cons:\ err(B') > \alpha/2} err(B') \leqslant (\alpha/2)|Cons|,$$

where $Cons$ is the set of all $B' \in N(A)$ that are consistent with B. This expression can be equivalently rewritten as

$$\mathbf{Pr}_{B' \in Cons, x \in B'}[err(B') > \alpha/2 \ \&\ x \in Err(B')] \leqslant \alpha/2. \quad (3)$$

For independent random $A \in \mathcal{S}$ and $B \in N(A)$, let $E_1(A, B)$ be the event that (A, B) is good, but the inequality (3) does not hold (i.e., the probability in (3) is greater than $\alpha/2$).

For independent random $A \in \mathcal{S}$, $B \in N(A)$, $B' \in N(A)$, and $x \in B'$, let $E(A, B, B', x)$ be the event that

$$(A, B) \text{ is correct } \ \& \ B' \in Cons \ \& \ err(B') > \alpha/2 \ \& \ x \in Err(B').$$

The probability of E is the average over all $B' \in \mathcal{T}$ of the conditional probabilities of E given B'. Consider any fixed B' with $err(B') > \alpha/2$. For each such B', the set A is a uniform element of $N(B')$ in the inclusion graph. By the sampling property of the $\mathcal{S}$-graph $H(B', N(B'))$, the probability that a random $A \in N(B')$ completely misses the subset $Err(B')$ is at most $\nu(err(B'))$. If A has nonempty intersection with $Err(B')$, then it cannot be the case that both (A, B) is correct and $B' \in Cons$. Hence, given B', the conditional probability of the event E is at most $\nu(err(B')) \cdot err(B')$, and so,

$$\mathbf{Pr}[E] \leqslant \frac{1}{|\mathcal{T}|} \sum_{B' \in \mathcal{T}: err(B') > \alpha/2} err(B') \cdot \nu(err(B')),$$

which is at most $\alpha\epsilon'^2/4$ by the assumption of the lemma.

We have

$$\mathbf{Pr}[E \mid E_1] > (\alpha/2)\mathbf{Pr}_{B' \in \mathcal{T}}[B' \in Cons \mid E_1] \geqslant \alpha\epsilon'/2, \quad (4)$$

where the first inequality is by the definition of the event E_1, and the second inequality by the definition of goodness of (A, B). On the other hand, $\mathbf{Pr}[E \mid E_1] = \mathbf{Pr}[E \ \& \ E_1]/\mathbf{Pr}[E_1] \leqslant \mathbf{Pr}[E]/\mathbf{Pr}[E_1]$. Combined with (4), this implies that $\mathbf{Pr}[E_1] \leqslant \mathbf{Pr}[E] \cdot 2/(\alpha\epsilon') \leqslant \epsilon'/2$.

Clearly, $\mathbf{Pr}_{A \in \mathcal{S}, B \in N(A)}[(A, B) \text{ is } \alpha\text{-excellent}]$ is at least

$$\mathbf{Pr}_{A \in \mathcal{S}, B \in N(A)}[(A, B) \text{ is good}] - \mathbf{Pr}_{A \in \mathcal{S}, B \in N(A)}[E_1].$$

By Lemma 3.8, the first probability in the difference above is at least ϵ', and, by what we showed earlier, the second probability is at most $\epsilon'/2$. The lemma follows. □

PROOF. (Proof of Theorem 3.2) The proof follows easily from Lemmas 3.4 and 3.7. We simply set $\beta = \delta/16$, $\lambda = \epsilon'/2$, $\alpha = \beta/2 = \delta/32$, and $\nu(\mu) = \alpha\epsilon'^2/(4\mu) = \delta\epsilon'^2/(128\mu)$. □

4. EXTENSIONS

Here, we give the intuition for some of the extensions mentioned in the introduction, Theorems 1.4, 1.5, 1.6, and 1.7. The proofs appear in the full paper. [3]

[3]Full paper available on: http://www.cse.ucsd.edu/users/russell/

4.1 Approximate version of the Uniform Direct-Product Theorem

To prove the approximate version of the Uniform Direct Product Theorem, we follow the same outline as for the exact version. The main changes we need to make are:

Before, if $C'(B)$ was correct, it was correct on the subset A. Here, we need to bound the chance that, even if $C'(B)$ is almost correct, its number of mistakes on A is disproportionately high. We include this in the definition of "correct edge", so that two correct edges for A will be (mostly) consistent on A. Second, before, we had the correct values for A, and any deviation from these values could be used to rule out $C'(B')$ as inconsistent. Now, our values for even good A and B' are somewhat faulty, and so could be somewhat inconsistent.

4.2 Derandomized Direct-Product Theorems

Here we will sketch the proof of Theorem 1.5. For $K = \text{poly}(1/\epsilon)$ and $k = O(\log 1/\epsilon)$, let $\mathcal{K}$ denote the collection of all k-subsets of $\{1, \ldots, K\}$. We need to analyze the function $h : \mathcal{T} \times \mathcal{K} \to \{0,1\}^k$ mapping $(T, i_1, \ldots, i_k)$ to $g(T)|_{i_1,\ldots,i_k}$, where $\mathcal{T}$ is a collection of affine d-dimensional subspaces of $\mathbb{F}_q^m$.

First we analyze the input size of h. It consists of $O(n)$ bits to describe a constant-dimensional affine subspace T, plus $k \log K = O((\log 1/\epsilon)\delta^{-1} \cdot (\log 1/\epsilon + \log 1/\delta)) = O((\log 1/\epsilon)^2)$ bits to specify the k-subset of $\{1, \ldots, K\}$, for constant δ. For $\epsilon \geq e^{-\Omega(\sqrt{n})}$, we get that the total input size is $O(n)$.

Suppose h is ϵ-computable in $\mathsf{BPTIME}(t^{1/c})//(1/c)\log t$. Given a circuit ϵ-computing h, we will show how to efficiently compute a list of circuits one of which $(1-\delta)$-computes f. This will imply that f is $(1-\delta)$-computable in $\mathsf{BPTIME}(t)//\log t$, contrary to the assumption of the theorem.

Our argument follows along the lines of a standard analysis of code concatenation (see, e.g., [15]). Suppose we have a circuit C' that ϵ-computes h. By averaging, we get that for at least $\epsilon/2$ fraction of $T \in \mathcal{T}$, the equality $C'(T, \kappa) = g(T)|_\kappa$ holds for at least $\epsilon/2$ fraction of k-subsets $\kappa \in \mathcal{K}$. Call $\mathcal{T}_{good}$ the set of such good Ts.

By Theorem 1.3, we know that the Independent intersection code is δ'-approximately $(\epsilon/2, O(1/\epsilon))$-list decodable. So, for every $T \in \mathcal{T}_{good}$, we can efficiently recover a list of $\ell = O(1/\epsilon)$ length-K strings, one of which $(1-\delta')$-agrees with $g(T)$.

We can then use this algorithm as the input to the decoding algorithm for the K-wise direct product using affine subspace, to get a list of possible functions f.

4.3 Hardness condensing

In this subsection, we reinterpret the results of the previous section to give a version of hardness condensing for the semi-uniform model, proving Theorem 1.6.

Imagine the sets B in the affine subspace construction as being exponentially large but succinctly representable (as is the case when $k = q^d$ is large). The idea is that, instead of $C'(B)$ explicitly giving the values of f on B, we could replace $C'(B)$ with a meta-algorithm that produces a circuit that computes $f|_B$. We could still estimate the agreement of two such circuits on A. Thus, if f is hard, the restricted function $f|_B$ is hard for almost all B.

4.4 k-XOR code

Here we sketch the proof of Theorem 1.7. First, for a Boolean function f, consider the function $f^{H_{2k}}(x_1, ..x_{2k}, r) = \langle f^{2k}(x_1, ..x_{2k}), r\rangle$, where r is a $2k$ bit vector, and we take the inner product mod 2. Equivalently, $f^{H_{2k}}$ is a function that is the xor of the values of f not on a fixed number k of inputs, but of a number of elements between 0 and $2k$ chosen according to the binomial distribution. We can also think of it as the concatenation of the $2k$-wise direct product code and the Hadamard code. By the decoding algorithm of [5], for a fixed k-set $x_1, ..x_k$, whenever a circuit has a conditional advantage ϵ in guessing the inner product, we can list decode to get a list of size $O(1/\epsilon^2)$ containing $f^{2k}(x_1, ..x_{2k})$. Thus, we can reduce approximate list-decoding $f^{H_{2k}}$ to list-decoding f^{2k}, albeit with a larger list size due to the change.

Since there is an $\Omega(1/\sqrt{k})$ probability that a random string of length $2k$ has Hamming weight k, if we have advantage ϵ in predicting $f^{\oplus k}$, we get a non-negligible advantage in predicting $f^{H_{2k}}$. In particular, for each set B of size $2k$ where the conditional advantage of predicting the *xor* of a subset of size k is large, we can get a small list of possible values for $f^{2k}(B)$.

We will show how to select an element from this list so that the overall probability of selecting a string δ-close to $f^{2k}(B)$ is $\Omega(\epsilon^2)$, where $\delta = O(\log 1/\epsilon/k)$. Combining this with the list-decoder for approximate direct product, we get an optimal $O(1/\epsilon^2)$ list decoding for the truncated Hadamard code.

The procedure is: For each candidate, approximate the correlation between the circuit's values on subsets of B and the corresponding inner product with the candidate. Order the candidates from highest rank to lowest. Prune the candidates that have a higher rank candidate that is δ close to it, or $1-\delta$ far from it. Of the ones that are left, pick one with probability proportional to the square of its bias. With probability 1/2, output this candidate, otherwise output its negation.

Let the actual advantage of the algorithm on subset B be γ_B, and assume $\gamma_B \geq \epsilon$. The actual value of $f^{2k}(B)$ probably appears in the list. If it is pruned, there is a higher ranked candidate either very close to it or very far from it, with correlation at least γ_B. The set of unpruned candidates are all between δ and $1-\delta$ far from each other; hence, random *xor*'s of k of their bits are almost uncorrelated. Using an approximate version of Parseval's inequality, we show that the sum of the squares of the correlations for such candidates must be $O(1)$. Thus the probability of picking the candidate very close to or very far from $f^{2k}(B)$ is $\Omega({\gamma_B}^2)$. Since we either keep or flip this output, there is a conditional 1/2 probability of being approximately $f^{2k}(B)$. The global probability is then $\Omega(\mathbf{Exp}_B({\gamma_B}^2)) \geq \Omega((\mathbf{Exp}_B(\gamma_B))^2) = \Omega(\epsilon^2)$.

5. CONCLUSIONS

We gave an efficient, approximate, local list-decoding algorithm for the direct-product code, with information-theoretically optimal parameters (to within constant factors). Our new decoding algorithm is also very efficient (is in uniform randomized AC^0), and has a simple analysis. We also defined a natural generalization of direct-product codes, intersection codes, for families of subsets $(\mathcal{S}, \mathcal{T})$, and gave the conditions on $(\mathcal{S}, \mathcal{T})$ that suffice for efficient (approximate, local) list-decoding of these generalized codes. Finally, we gave a derandomized version of the direct-product code with an efficient decoding algorithm.

An interesting remaining open question is to get a *derandomized* uniform direct-product theorem with better parameters (pushing the error ϵ to $e^{-\Omega(n)}$, while keeping the new input size linear in the original input size). Another question is to improve the parameters of our approximate version of the uniform direct-product theorem (Theorem 1.4), ideally achieving a uniform version of the "Chernoff-type" direct-product theorem in the spirit of [11]. Finally, it is interesting to see if the ideas from our new list-decoding algorithm can help in improving the known uniform hardness amplification results for NP of [16].

6. REFERENCES

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

[2] M. Bellare, O. Goldreich, and S. Goldwasser. Randomness in interactive proofs. *Computational Complexity*, 3:319–354, 1993.

[3] J. Buresh-Oppenheim and R. Santhanam. Making hard problems harder. In *Proceedings of the Twenty-First Annual IEEE Conference on Computational Complexity*, pages 73–87, 2006.

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

[5] O. Goldreich and L. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, pages 25–32, 1989.

[6] O. Goldreich, N. Nisan, and A. Wigderson. On Yao's XOR-Lemma. *Electronic Colloquium on Computational Complexity*, TR95-050, 1995.

[7] O. Goldreich and S. Safra. A combinatorial consistency lemma with application to proving the PCP theorem. *SIAM Journal on Computing*, 29(4):1132–1154, 2000.

[8] W. Hoeffding. Probability inequalities for sums of bounded random variables. *American Statistical Journal*, pages 13–30, 1963.

[9] R. Impagliazzo. Hard-core distributions for somewhat hard problems. In *Proceedings of the Thirty-Sixth Annual IEEE Symposium on Foundations of Computer Science*, pages 538–545, 1995.

[10] R. Impagliazzo, R. Jaiswal, and V. Kabanets. Approximately list-decoding direct product codes and uniform hardness amplification. In *Proceedings of the Forty-Seventh Annual IEEE Symposium on Foundations of Computer Science*, pages 187–196, 2006.

[11] R. Impagliazzo, R. Jaiswal, and V. Kabanets. Chernoff-type direct product theorems. In *Proceeding of the Twenty-Seventh Annual International Cryptology Conference (CRYPTO'07)*, pages 500–516, 2007.

[12] R. Impagliazzo and A. Wigderson. P=BPP if E requires exponential circuits: Derandomizing the XOR Lemma. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 220–229, 1997.

[13] L. Levin. One-way functions and pseudorandom generators. *Combinatorica*, 7(4):357–363, 1987.

[14] R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 475–484, 1997.

[15] M. Sudan, L. Trevisan, and S. Vadhan. Pseudorandom generators without the XOR lemma. *Journal of Computer and System Sciences*, 62(2):236–266, 2001.

[16] L. Trevisan. On uniform amplification of hardness in NP. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, pages 31–38, 2005.

[17] L. Trevisan and S. Vadhan. Pseudorandomness and average-case complexity via uniform reductions. In *Proceedings of the Seventeenth Annual IEEE Conference on Computational Complexity*, pages 103–112, 2002.

[18] A. Yao. Theory and applications of trapdoor functions. In *Proceedings of the Twenty-Third Annual IEEE Symposium on Foundations of Computer Science*, pages 80–91, 1982.

[19] D. Zuckerman. Randomness-optimal oblivious sampling. *Random Structures and Algorithms*, 11(4):345–367, 1997.