SIAM J. COMPUT.
Vol. 39, No. 4, pp. 1637–1665

UNIFORM DIRECT PRODUCT THEOREMS: SIMPLIFIED, OPTIMIZED, AND DERANDOMIZED*

RUSSELL IMPAGLIAZZO†, RAGESH JAISWAL‡, VALENTINE KABANETS§, AND AVI WIGDERSON¶

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 [O. Goldreich and L. A. Levin, *A hard-core predicate for all one-way functions*, in Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing, Seattle, 1989, pp. 25–32], we also get a *fully uniform* version of Yao's XOR lemma [A. C. Yao, *Theory and applications of trapdoor functions*, in Proceedings of the Twenty-Third Annual IEEE Symposium on Foundations of Computer Science, Chicago, 1982, pp. 80–91] with *optimal* parameters, up to constant factors. Our results simplify and improve those in [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, Berkeley, CA, 2006, pp. 187–196]. 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.

Key words. direct product theorem, direct product code, Yao's XOR lemma, error-correcting code, list-decoding algorithms

AMS subject classifications. 68Q15, 68Q17, 68Q25, 68P30

DOI. 10.1137/080734030

*Received by the editors September 2, 2008; accepted for publication (in revised form) October 23, 2009; published electronically January 13, 2010.

http://www.siam.org/journals/sicomp/39-4/73403.html

†School of Mathematics, Institute for Advanced Study, Princeton, NJ 08540, and Department of Computer Science, University of California, San Diego, La Jolla, CA 92093 (russell@cs.ucsd.edu). This author's work was supported by the Simonyi Fund, The Bell Company Fellowship, and Fund for Math, and NSF grants DMS-0835373, CNS-0716790, and CCF-0832797. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

‡Department of Computer Science, University of California, San Diego, La Jolla, CA 92093 (rjaiswal@cs.ucsd.edu).

§Department of Computer Science, Simon Fraser University, Burnaby V5A 1S6, BC, Canada (kabanets@cs.sfu.ca). This author's work was partially supported by an NSERC Discovery grant.

¶School of Mathematics, Institute for Advanced Study, Princeton, NJ 08540 (avi@ias.edu). This author's work supported by NSF grants DMS-0835373 and CCF-0832797.

1. Introduction. Applications of complexity theory such as cryptography and derandomization require reliably hard problems that cannot be solved by any algorithm with a nontrivial 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 nonnegligible 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 pseudorandomly.) Many strong direct product theorems are known for nonuniform models, such as Boolean circuits [Yao82, Lev87, GNW95, Imp95, IW97, STV01]. Unfortunately, in general, direct product theorems fail in completely uniform models such as probabilistic computation.

For further discussion, it will be more convenient to view direct product theorems in the language of error-correcting codes. Impagliazzo [Imp02] and Trevisan [Tre05] pointed out that proofs of direct product theorems correspond to (approximate) local error-correction of sparse codes. Using this view, we think of a function f as being encoded by $\mathit{Code}(f) = f^k$, its values on all k-tuples. That is, the message is the truth table of the function f, and the encoding of f is the truth table of the direct product function f^k.[1] Given a highly corrupted encoding C' of some function f, we would like to recover f. We want *local* decoding in the sense that the decoding algorithm, given oracle access to C', should produce an efficient circuit for f (which may also use oracle access to C'). Having efficient local decoding of the direct product code immediately translates into the hardness amplification properties of the direct product construction. Intuitively, if the decoder can recover a small circuit computing f well on average (thereby contradicting the assumed average-case hardness of f) from a small circuit C' that has only ϵ agreement with f^k, then f^k must be hard to compute by small circuits on all but less than ϵ fraction of inputs.

A completely *uniform* decoding algorithm for the direct product encoding $\mathit{Code}(f)$ is an algorithm that constructs a *single* circuit C computing f well on average, when given as input some circuit C' that agrees with f^k on a small, say ϵ, fraction of all k-tuples. When ϵ is sufficiently close to 1, e.g., if $\epsilon \geqslant 0.9$, then such uniform decoding is possible (and easy to analyze). However, if $\epsilon \leqslant 1/2$, it is easy to see that C' does not uniquely determine f. Indeed, consider $t = 1/\epsilon$ different (e.g., randomly chosen) functions $f_1, \ldots, f_t$. Partition the set of all k-tuples into t sets (of measure ϵ each). Define C' so that C' agrees with f_i^k on the k-tuples in the ith set of the partition. Then this C' has agreement ϵ with each of the $t = 1/\epsilon$ functions $f_1, \ldots, f_t$.

The example given above shows that the direct product code $\mathit{Code}(f) = f^k$ is not uniquely decodable when the fraction of corrupted symbols in the codeword is at least $1/2$. In order to tolerate high corruption rates (which is the interesting case for hardness amplification), we need to allow list-decoding: Given C', a corrupted version of f^k, a decoding algorithm may output a list of circuits such that one of them computes f well on average. The list size is an important parameter that we would like to minimize. The example above shows the list size lower bound $1/\epsilon$ (for list-decoding from C' that has ϵ agreement with the function f^k, which we wish to decode).

[1] Note that if f is a Boolean function, then the message is a string over the binary alphabet $\{0, 1\}$, whereas its encoding is a string over the larger alphabet $\{0, 1\}^k$.

The list size may also be viewed as corresponding to a certain amount of nonuniform advice used by the decoding algorithm. More precisely, if the list-decoding algorithm outputs a list of t circuits, we can use $\log t$ bits of advice to indicate which of the circuits computes the right function f (by specifying the index of that circuit on the list). Conversely, a decoding algorithm that uses a bits of nonuniform advice can be viewed as a list-decoding algorithm producing a list of size 2^a (one circuit per advice string).

Most previously known proofs of the direct product theorem are highly nonuniform in the sense that they yield decoding algorithms with the list size *exponential* in $1/\epsilon$. In contrast, more uniform proofs of the direct product theorem should yield list-decoding algorithms with the list size at most polynomial in $1/\epsilon$. Impagliazzo, Jaiswal, and Kabanets [IJK06] gave the first such proof of the direct product theorem achieving the list size $\text{poly}(1/\epsilon)$; however, the proof was quite complex and fell short of the information-theoretic bounds in many respects. We elaborate more on the uniform versus nonuniform direct product theorems in section 1.4.

In this paper, 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). In particular, the list size is $O(1/\epsilon)$, which matches the list size lower bound given in the example above (up to a constant factor).[2]
- *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 nonuniform setting!).
- *Generality.* The decoding algorithm and its proof turn 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 [TV02, Tre05], efficient list-decoding has the same consequences as unique decoding in terms of hardness amplification within many natural complexity classes, e.g., $\mathsf{NP}, \mathsf{P}^{\mathsf{NP}\|}, \#\mathsf{P}, \mathsf{PSPACE}$, and EXP. Impagliazzo, Jaiswal, and Kabanets [IJK06] used their (suboptimal) direct product decoding algorithm for hardness amplification in this uniform

[2] To the best of our knowledge, it was not known prior to this work whether $O(1/\epsilon)$ is the correct upper bound on the list size, even just information-theoretically. Here we prove this upper bound constructively, by providing an efficient list-decoding algorithm.

setting. The improved parameters of the direct product decoding obtained in the present paper immediately yield improved hardness amplification results using the approach of [IJK06].

1.1. Statement of the uniform direct product theorem. Let us describe now the hardness amplification results in computational complexity terms. For this we need the following terminology. 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 [TV02], we define the "semiuniform" 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 probabilistically 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 $\mathsf{BPTIME}(\mathrm{poly}(t))//O(\log t)$.

Given a Boolean function $f : \{0,1\}^n \to \{0,1\}$, the k-wise direct product function f^k is mapping 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 $(f(x_1), \ldots, f(x_k))$.[3]

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 $\epsilon = \epsilon(n)$ such that $\epsilon \geq e^{-\delta k/c}$ and $\epsilon > t^{-1/c}$, if f is δ-hard for $\mathsf{BPTIME}(t)//\log t$, then f^k is $(1-\epsilon)$-hard for $\mathsf{BPTIME}(t^{1/c})//(1/c)\log t$.*

The proof can be obtained 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 let $0 < \epsilon, \delta < 1$ 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 $O((\log 1/\delta)/\epsilon)$ of C'-oracle gates).*

Remark 1.3 (optimality of the parameters). If a function f is computable on at most $1-\delta$ fraction of inputs by some circuit, then using k (independent) copies of that circuit we can compute f^k on $(1-\delta)^k \approx e^{-\delta k}$ fraction of input k-tuples. Theorem 1.2 shows that this is the best possible, up to a constant factor c in the exponent of $e^{-\delta k}$. That is, the relationship among k, δ, and ϵ is tight, up to constant factors. Also, running the algorithm $\mathcal{A}$ of Theorem 1.2 $O(1/\epsilon)$ times, one gets a list of $O(1/\epsilon)$ circuits that with constant probability (say, at least 3/4) contains a good circuit for f. Up to a constant factor, this matches the simple list size lower bound for list-decoding from agreement ϵ described earlier. Finally, it is also possible to show

[3] This is slightly different from the usual definition of k-wise direct product, where one allows as inputs to f^k all k-tuples $(x_1, \ldots, x_k)$ rather than k-sets; the case of k-tuples can be easily deduced from the case of k-sets.

that $\Omega((\log 1/\delta)/\epsilon)$ oracle queries to C' are necessary for successful decoding, and so our decoding algorithm is optimal (up to a constant factor) also in this parameter.

In our proof, 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 (ordered) 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 $C_{A,v}$ (see Algorithm 1).

Algorithm 1 Circuit $C_{A,v}$.

On input $x \in \{0,1\}^n$, check whether $x = a_i$ 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 some default (error) output if no consistent B is found even after $100 \cdot (\ln 1/\delta)/\epsilon$ iterations.

The algorithm $C_{A,v}$ makes sense when $v_i = f(a_i)$ for all $1 \leqslant i \leqslant s$. This alone would not be sufficient to ensure that $C_{A,v}$ is a good circuit for f. We will need a stronger property of the set A (that for many sets B, where $C'(B)$ is consistent with the values v for A, the values $C'(B)$ are mostly correct), which will still be ensured with reasonable probability for a random choice of (A, v) as described above.

Here is the complete description of our decoding algorithm $\mathcal{A}$ (see Algorithm 2).

Algorithm 2 Algorithm $\mathcal{A}$.

Pick at random a k-set B_0 and 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 proof technique will allow us to analyze a certain generalized direct product code, which we define below. 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). We assume some fixed ordering of the elements of U, and identify a size-s subset of U with the s-tuple of ordered elements.

DEFINITION 1.4 (intersection codes). *For a parameter $k \in \mathbb{N}$, we specify a generalized k-wise direct product encoding of f by two families of subsets of U. Let $\mathcal{T}$ be a family of k-subsets of U, and let $\mathcal{S}$ be a family of s-subsets of U (with $s < k$); the family $\mathcal{S}$ is used only in the decoding algorithm and its analysis. The* intersection code *$Code = Code(\mathcal{T}, \mathcal{S})$ is defined by $Code(f) : \mathcal{T} \to R^k$, giving for every k-set $B \in \mathcal{T}$ the values $f(b)$ for all $b \in B$.*

The length of the message in *Code* is $|U|$, and the length of the encoding is $|\mathcal{T}|$. We would like to maximize the rate $|U|/|\mathcal{T}|$.

Our two running examples of such families $(\mathcal{T}, \mathcal{S})$ are the following:

- *Independent.* $\mathcal{T}$ are all k-subsets of U, and $\mathcal{S}$ are all s-subsets of U. Here we fix $s = k/2$.
- *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 rate of this code is very bad, as it maps $|U|$-size messages to approximately $|U|^k$-size codewords.

The Subspaces example gives a code with much better rate, and it will give us a derandomized version of the direct product theorem. The 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 need only 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, where $|U| = q^m = 2^n$. That is, the Subspaces code maps $|U|$-size messages to $|U|^{O(1)}$-size codewords.

Next we define approximately list-decodable codes, which naturally generalize list-decodable codes. In the case of list-decodable codes, one is interested in the upper bound on the list size one needs in order to capture all messages whose encodings have some nontrivial agreement with a given corrupted codeword (with the list size being a function of the agreement). In the approximate list-decoding setting, the strings on the list do not have to be exactly equal to the messages, but rather "almost equal." If we think of the messages as (the truth tables of) some functions f_i's, then it is sufficient for the list to consist of functions g_i's so that, for each message f (whose encoding has nontrivial correlation with the given corrupted codeword C'), there is some g_j on the list such that f and g_j agree almost everywhere. For *efficient local* list-decoding, we require an efficient algorithm that produces small circuits for these functions g_i's on the list (given oracle access to the corrupted codeword C'). The decoding is local in the sense that we can compute the function g_i on any given input, as opposed to the usual decoding where we would simply output the entire truth table of g_i.

More formally, we have the following definition.

DEFINITION 1.5 (approximately list-decodable codes). *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* $g_1, g_2, \ldots, g_\ell$ *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 *(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 1.6. *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.*[4]

We use very little about the set systems $\mathcal{S}$ and $\mathcal{T}$. The following is an informal summary of the properties we need.

Computational assumptions. It is possible to efficiently sample from the following distributions: B uniformly chosen in $\mathcal{T}$; given $B \in \mathcal{T}$, uniformly pick $A \in \mathcal{S}$

[4]This yields a much simpler construction of nonbinary codes, locally list-decodable in uniform randomized AC^0, than the one given by [GGH+07].

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 for 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 for 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 1.6 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, generalizing our earlier definition of approximately list-decodable codes (Definition 1.5).

DEFINITION 1.7. *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* $g_1, g_2, \ldots, g_\ell$ *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 1.6.

THEOREM 1.8. *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 1.8 will also allow us to obtain a strong derandomized version of the uniform direct product theorem for a Boolean function $f : \{0,1\}^n \to \{0,1\}$. The k-wise direct product encoding based on affine subspaces already yields a harder function on inputs of size $O(n)$. However, Theorem 1.6 (for Subspaces) says that to obtain a function with hardness $1-\epsilon$ from a function with constant hardness δ, one needs to take $k = \text{poly}(1/\epsilon)$, which is too big when ϵ is exponentially small in n. For nonuniform derandomized hardness amplification, [IW97, STV01] show how to take an n-variate Boolean function of constant hardness, and construct a new $O(n)$-variate Boolean function of hardness $1 - 2^{-\Omega(n)}$. By analogy with this result, we would like to obtain a derandomized direct product encoding h of f such that h has hardness $1-\epsilon$ for $\epsilon = e^{-\Omega(n)}$, while the input size of h is $O(n)$ and the output size $k = O(\log 1/\epsilon)$. That is, we would like to have the exponential relationship between k and ϵ (as in the Independent case), and, at the same time, have the k-tuples that are describable with only $O(n)$ bits so that the new function has input size $O(n)$ (as in the Subspaces case). Such a derandomized direct product construction would have the best possible relationship among the parameters k, ϵ, and the input size.

We will be able to meet this goal partially: we define a function h of hardness $1-\epsilon$ for $\epsilon = e^{-\Omega(\sqrt{n})}$ with input size $O(n)$ and $k = O(\log 1/\epsilon)$. The function h

is a concatenation of two encodings: first we encode a given Boolean function $f : \{0,1\}^n \to \{0,1\}$ with the Subspaces K-wise direct product code, obtaining a function $g : \mathcal{T} \to \{0,1\}^K$, where $K = O(1/(\epsilon\delta)^8)$ and $\mathcal{T}$ is the collection of all d-dimensional affine subspaces of $\mathbb{F}_q^m$; here we identify $\{0,1\}^n$ with $\mathbb{F}_q^m$. Then we encode each output $g(B)$, for $B \in \mathcal{T}$, with the Independent k-wise direct product code, for the universe $\{0,1\}^K$ and $k = O((\log 1/\epsilon)/\delta)$. Theorem 1.8 is needed to handle possible errors created in the decoding of the inner code.

THEOREM 1.9 (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 $\mathsf{BPTIME}(t)//\log t$, then the function h defined from f as described above is $(1-\epsilon)$-hard for $\mathsf{BPTIME}(t^{1/c})//(1/c)\log t$. The input size of h is $O(n)$.*

We give an interpretation of Theorem 1.9 in terms of "hardness condensing" in the spirit of [BOS06]. 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 following problem: Given (a description of) $B \in \mathcal{T}$, construct a circuit that computes $f|_B$ well on average. We show the following.

THEOREM 1.10 (hardness condensing). *For an absolute constant $c > 0$, 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 also consider a truncated version of the Hadamard code, and argue that it is approximately, efficiently, locally, list-decodable, with information-theoretically optimal parameters up to constant factors. For a Boolean function $f : \{0,1\}^n \to \{0,1\}$ and a parameter $k \in \mathbb{N}$, the k-XOR encoding of f is defined as the function $f^{\oplus k}$ mapping each k-subset of n-bit strings $(x_1, \ldots, x_k)$ to the value $\oplus_{i=1}^k f(x_i)$. This binary encoding of f is essentially the encoding used in Yao's XOR lemma [Yao82] (hence our motivation for defining the k-XOR code above), with the only difference that we consider k-sets rather than k-tuples of n-bit strings.

We have the following theorem.

THEOREM 1.11. *The k-XOR code is efficiently, locally, δ-approximately $(1/2 + \epsilon, O(1/\epsilon^2))$–list-decodable, for $\delta = O((\log 1/\epsilon)/k)$.*

As one might expect, the proof of Theorem 1.11 goes by analyzing the concatenation of the direct product code and a certain variant of the Hadamard code. However, to achieve the information-theoretically optimal list size $O(1/\epsilon^2)$ (see, e.g., [IJK06] for an argument concerning why this bound is optimal), one needs to be careful in the decoding analysis of such a concatenated code. Our proof of Theorem 1.11 ends up relying on the efficient $(\delta, \Omega(\delta))$-approximate list-decoder of Theorem 1.8 for the Independent codes.

1.4. Relation to previous work.

1.4.1. Nonuniform direct product theorem. The classical direct product theorem (and closely related Yao's XOR lemma [Yao82]) for circuits has many proofs [Lev87, Imp95, GNW95, IW97]. 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 [Lev87, Imp95, GNW95, IW97]). 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 nonuniform 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 [IW97], 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 nonuniform advice $f(x_j)$, $j \neq i$. The first result of that type was obtained in [IJK06]. 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 [IW97] on C' and the obtained examples.

To get sufficiently many examples, [IJK06] 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, [IJK06] 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 [GS00, DR06], and we were inspired by these papers. However, while those authors showed that this is sufficient in the unique decoding regime (where the circuit 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 [RS97, AS03], 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 nonnegligible probability is not sufficient for testing nonnegligible correlation with a direct product function. Despite the lack of any formal connection, some ideas of the present paper have been used in the follow-up work by Impagliazzo, Kabanets, and Wigderson [IKW09] to get direct product testing results, which were then applied to constructing new PCPs.

Remainder of the paper. Section 2 contains some background facts and basic sampling properties of graphs used in the 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 the proofs of Theorems 1.8, 1.9, and 1.10. Theorem 1.11 is proved in sction 5. Section 6 contains concluding remarks and open questions.

2. Preliminaries.

2.1. Concentration bounds. The standard form of the Hoeffding bound [Hoe63] 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 (see Hoeffding [Hoe63]). *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 $0 < \gamma \leqslant 1$, $\mathbf{Pr}[|X - \mu| \geqslant \gamma\mu] \leqslant 2 \cdot e^{-\gamma^2\mu/3}$.*

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. Here we will argue that points in a random affine (or linear) subspace of $U = \mathbb{F}_q^m$ (for a finite field $\mathbb{F}_q$) are, essentially, pairwise independent and uniform over U.

Let M be a collection d vectors $a_1, \ldots, a_d \in U$, where a_i's are picked independently uniformly at random from U. Let $b \in U$ also be picked uniformly at random from U. We will think of M as a matrix with column-vectors $a_1, \ldots, a_d$, and of b as a column-vector. Let $S = \mathbb{F}_q^d$ be the set of all d-dimensional vectors over $\mathbb{F}_q$; we think of elements in S as column-vectors.

It is easy to argue the following.

LEMMA 2.3. *For random M and b as above, the random variables $Ms + b$, for $s \in S$, are pairwise independent and uniform over U.*

Proof. The proof is a straightforward generalization (to m dimensions) of the standard fact that, for random field elements $a, b \in \mathbb{F}_q$, the variables $ai + b$, for $i \in \mathbb{F}_q$, are pairwise independent and uniform over $\mathbb{F}_q$. The uniformity is obvious. For pairwise independence, consider any $i, i' \in \mathbb{F}_q$, with $i \neq i'$, and any $c, c' \in \mathbb{F}_q$, and observe that the system of linear equations $ai + b = c$ and $ai' + b = c'$ (in the variables a and b) has a unique solution over $\mathbb{F}_q$. ∎

Let S' be any fixed subset of nonzero vectors of $\mathbb{F}_q^d$ such that every two $s, s' \in S$ are linearly independent; it is easy to show that one can choose $t = (q^d - 1)/(q - 1)$

such pairwise independent vectors.

We have the following lemma.

LEMMA 2.4. *For a random M as above, the random variables Ms, for $s \in S'$, are pairwise independent and uniform over U.*

Proof. The proof follows by generalizing the simple fact that, for random $a, b \in \mathbb{F}_q$, the variables $ai + bj$, for all nonzero pairwise linearly independent vectors $(i, j) \in \mathbb{F}_q^2$, are pairwise independent and uniform over $\mathbb{F}_q$. The uniformity is obvious. The pairwise independence follows since, for any two linearly independent (i, j) and (i', j') in $\mathbb{F}_q^2$, and any $c, c' \in \mathbb{F}_q$, the system of linear equations $ai + bj = c$ and $ai' + bj' = c'$ has a unique solution in a, b. □

We will view the vectors in M as the basis of a random linear subspace. The probability that M has rank less than d is at most $O(q^d/q^m)$, which will be negligible for our choice of parameters. Thus, with probability very close to 1, we have that a random M has full rank. So, essentially, the random variables of Lemma 2.3 correspond to all points in a randomly chosen d-dimensional affine subspace, whereas the random variables of Lemma 2.4 correspond to a subset of points in a random d-dimensional linear subspace.

Let $\mathcal{E} = \mathcal{E}(M)$ be the event that the random matrix M has full rank. We have the following.

CLAIM 2.5. *For any random event X,*

$$\mathbf{Pr}[X] - negl \leqslant \mathbf{Pr}[X \mid \mathcal{E}] \leqslant (1 + negl) \cdot \mathbf{Pr}[X],$$

where the probabilities are over the random choice of M and b, and $negl = O(q^d/q^m)$.

Proof. The proof follows by the definition of conditional probability and the bound on the probability of $\mathcal{E}$. □

We will interpret this claim as basically saying that a sequence of all q^d elements of a randomly chosen d-dimensional affine subspace of U are pairwise independent and uniform over U, and that the sequence of elements of a randomly chosen d-dimensional linear subspace (corresponding to points $s \in S'$) are also 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 *biregular* if the degrees of vertices in L are the same, and if 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 2.6 (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 2.7 ($\mathcal{S}$-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 2.8 ($\mathcal{T}$-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.

DEFINITION 2.9 ($\lambda(\cdot)$-sampler). *Let $G = G(L, R)$ be any biregular bipartite graph. For a function $\lambda : [0,1] \to [0,1]$, we say that G is a* $\lambda(\cdot)$-sampler *if for every $0 \leqslant \mu \leqslant 1$ and 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* [BGG93] or *approximating disperser* [Zuc97].

In Definition 2.9 above, the sampling property must hold for *every* value μ, where the fraction of "bad" vertices decreases as a function $\lambda(\mu)$. While the graphs we consider in this paper are shown to be samplers according to this general definition, a weaker sampling property is sufficient for most of our arguments. This weaker property is formalized in the following definition.

DEFINITION 2.10 ((β, λ)-sampler). *For a biregular bipartite graph $G = G(L, R)$ and parameters $0 \leqslant \beta, \lambda \leqslant 1$, we call G a* (β, λ)-sampler *if, for every value μ, where $\beta \leqslant \mu \leqslant 1$ and 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 $\lambda \cdot |R|$ vertices $r \in R$, where*

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

Note that the difference from Definition 2.9 of a $\lambda(\cdot)$-sampler is that a (β, λ)-sampler needs to work only for functions F with an average $\mu \geqslant \beta$ and the fraction of "bad" vertices (where there is a significant deviation from the expectation μ) is at most some fixed value λ which is independent of μ. Obviously, if G is a $\lambda(\cdot)$-sampler (according to Definition 2.9), where the function $\lambda(\cdot)$ is monotone nonincreasing, then, for every $0 \leqslant \beta \leqslant 1$ and $\lambda = \lambda(\beta)$, G is also a (β, λ)-sampler (according to Definition 2.10).

For the analysis of intersection codes $Code(\mathcal{S}, \mathcal{T})$ based on families $\mathcal{S}$ and $\mathcal{T}$, we will need the corresponding $\mathcal{S}$-graphs and $\mathcal{T}$-graphs to 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 2.11. *For both Independent and Subspaces families $(\mathcal{S}, \mathcal{T})$, the $\mathcal{S}$-graph H is a $\lambda(\cdot)$-sampler, where*

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

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 2.5), and then we apply Chebyshev's bound of Lemma 2.2. □

LEMMA 2.12. *For both Independent and Subspaces families* $(\mathcal{S}, \mathcal{T})$*, the* $\mathcal{T}$*-graph* G *is a* $\lambda(\cdot)$*-sampler, where*

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

Proof. For Independent, we use the Hoeffding bound of Lemma 2.1.

For Subspaces, we use pairwise independence and the Chebyshev bound. Fix an affine subspace A of dimension r. Suppose A is $V + v$, for some r-dimensional linear subspace V of $U = \mathbb{F}_q^m$ and a vector $v \in U$. Let $V^\dagger$ be any $(m-r)$-dimensional linear subspace of U that is complementary to V in the sense that V and $V^\dagger$ are linearly independent (and so $U = V + V^\dagger$); such a subspace is not unique, but any choice of $V^\dagger$ will do for our purposes. To choose a random $d = 2r$-dimensional affine subspace B containing A, we choose a random r-dimensional subspace W of $V^\dagger$, and we define our affine $2r$-dimensional subspace $B = A+W$. It is easy to see that (no matter which subspace $V^\dagger$ we use) the resulting distribution over subspaces B is uniform over all $2r$-dimensional affine subspaces containing A.

Note that all of $U \setminus A$ can be represented as the disjoint union of cosets $A + u$, over all distinct nonzero vectors u in the complementary subspace $V^\dagger$. A function $F : (U \setminus A) \to [0,1]$ with the expectation β yields $[0,1]$-valued functions F_u, where F_u is the restriction of F to the coset $A + u$ for every nonzero vector $u \in V^\dagger$. Let β_u denote the average value of F_u over the points in $A + u$. Clearly, the average of β_u's is exactly β.

If we pick t nonzero vectors $u_1, \ldots, u_t \in V^\dagger$ independently at random, we would obtain by the Chernoff–Hoeffding bound that the average $(1/t)\sum_{i=1}^t \beta_{u_i}$ is very likely to be close to β. Similarly, if these t vectors were chosen pairwise independently, we could argue the concentration around the expectation β by Chebyshev's bound. The intuition is that vectors in a random r-dimensional subspace W are essentially pairwise independent, and hence we can argue that our random affine subspace B is likely to be a good sample for estimating the average of F.

More precisely, let $w_1, \ldots, w_t \in \mathbb{F}_q^r$ be any fixed collection of $t = (q^r - 1)/(q - 1)$ nonzero vectors such that every two of them are linearly independent. By Claim 2.5, in a random W the t corresponding vectors of W are, essentially, pairwise independent and uniform over $V^\dagger$. Let us denote by ω_i, $1 \leqslant i \leqslant t$, the element of W corresponding to w_i (i.e., ω_i is a linear combination of the basis vectors of W with scalar coefficients being the r field elements of w_i).

For each field element $i \in \mathbb{F}_q$, define $B_i = \cup_{j=1}^t (A + i \cdot \omega_j)$. Note that $B = \cup_{i \in \mathbb{F}_q} B_i$. Fix any nonzero $i \in \mathbb{F}_q$. For a random W, the vectors $i \cdot \omega_1, \ldots, i \cdot \omega_t$ are pairwise independent. By Chebyshev's bound of Lemma 2.2, the probability that $(1/|B_i|) \cdot \sum_{x \in B_i} F(x)$ is less than $\beta/2$ or more than $3\beta/2$ is at most $4/(\beta t)$. By the union bound, the probability that at least one of the B_i's deviates from the expectation is at most $4(q-1)/(\beta t)$. Thus, with probability at least $1 - 4/(\beta q^{r-2}) = 1 - 4q^2/(\beta s)$, a random affine subspace B containing A is a good sample for estimating the expectation of F. Since $s = \sqrt{k}$ for Subspaces, we get the desired lemma. □

Note that for each $\lambda(\cdot)$-sampler in Lemmas 2.11 and 2.12 above, the function $\lambda(\cdot)$ is monotone nonincreasing. Hence, for every $0 \leqslant \beta \leqslant 1$ and $\lambda = \lambda(\beta)$, all of these samplers are also (β, λ)-samplers.

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 2.13. *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 biregular, 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 $W' \subseteq W$ 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 biregular 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 biregular, 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 biregular). However, we prove that this is approximately true when G is a sampler.

LEMMA 2.14. *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

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

On the other hand, $\sum_{y \in W}\big(D \cdot \mathbf{Exp}_{x \in N(y)}[\xi(x)]\big)$ 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

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

Finally, after comparing the bounds in (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 $\mathit{Code}(\mathcal{S}, \mathcal{T})$ be the intersection code defined for these families. Fix a function $f : U \to R$. Let C' be a circuit that ϵ-computes $\mathit{Code}(f)$. We will give a randomized decoding algorithm for constructing 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 $\mathit{Code}(\mathcal{S}, \mathcal{T})$ 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 $\mathit{Cons} \subseteq N(A)$ 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 $C_{A,v}$: "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 $B' \notin \mathit{Cons}$, until the first $B' \in \mathit{Cons}$ is obtained. For this $B' \in \mathit{Cons}$, output the value $C'(B')|_x$. Produce the default (error) output if no $B' \in \mathit{Cons}$ 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 biregular), the $\mathcal{S}$-graph H is a $(\mu, \delta\epsilon'^2/(256\mu))$-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 1.6, which we restate below for convenience.

THEOREM 3.4 (Theorem 1.6 restated). *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.*

Proof of Theorem 1.6. For Independent, we get by Lemmas 2.11 and 2.12 that both H and G are $\lambda(\cdot)$-samplers with $\lambda(\mu) \leqslant e^{-\Omega(\mu k)}$. 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).[5] Choosing a sufficiently large constant d, we can ensure that $\epsilon^{cd'} < 2^{-c}/256$, and so $c\delta e^{-\Omega(c\delta k)} \leqslant c\delta\epsilon'^2 2^{-c}/256 \leqslant \delta\epsilon'^2/256$. 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 Theorem 3.2.

For Subspaces, Lemma 2.11 gives us that H is a $\lambda(\cdot)$-sampler with $\lambda(\mu) = 4/(\mu\sqrt{k})$. Hence, $\mu \cdot \lambda(\mu) \leqslant 4/\sqrt{k}$. The latter is at most $\delta\epsilon'^2/256$ for $\delta \geqslant 1024/\epsilon'^2\sqrt{k}$. Lemma 2.12 says that the graph G is a $(\delta/16, \epsilon'/2)$-sampler for $\delta \geqslant 128q^2/(\epsilon'\sqrt{k})$. Thus, to satisfy the conditions of Theorem 3.2, we can set $\delta \leqslant 1024q^2/(\epsilon'^2\sqrt{k})$, which is $O(1/(\epsilon'^2 k^{1/4}))$ for $q \leqslant k^{1/8}$.

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

Outline of the proof of Theorem 3.2. The proof of Theorem 3.2 will follow from two technical lemmas proved in the following subsections. First, in section 3.1, we define certain conditions on our auxiliary graphs (inclusion, $\mathcal{S}$-, and $\mathcal{T}$-graphs) and an edge (A, B) of the inclusion graph, and we show (in Lemma 3.8) that these conditions are sufficient for the circuit $C_{A,v}$ described above to satisfy the conclusion of Theorem 3.2. Then, in section 3.2, we prove (in Lemma 3.11) that a random choice of an edge (A, B) (as made by our decoding algorithm) will satisfy with probability $\Omega(\epsilon)$ the required sufficient conditions. The two lemmas imply Theorem 3.2. The formal proof of Theorem 3.2 (taking care of all the parameters) is given at the end of section 3.2.

3.1. Why $C_{A,v}$ works. 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 the following:

- $C'(B)$ is correct, which means that $v = f(A)$; 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 $x \in B' \setminus A$.

[5] In fact, if a circuit C' ϵ-computes f^k for $\epsilon \geqslant 0.9$, then for $k > 1/\delta$ there is a *single* algorithm that $(1-\delta)$-computes f: "Given input x, sample $O(\log 1/\delta)$ random k-sets B containing x, and output the majority answer of $C'(B)|_x$." For the analysis, it suffices to show that for each except $\delta/2$ fraction of inputs x, there are at least 2/3 sets B containing x such that $C'(B) = f^k(B)$, which is easy to argue.

As explained above, our proof of Theorem 3.2 will consist of showing that these conditions suffice, and that relatively many such edges (A, B) exist. In this subsection, we show that these conditions suffice.

First we sketch an informal argument explaining why the conditions above should be sufficient. Suppose we have an edge (A, B) satisfying the conditions above. The relative abundance of consistent sets B' for A can be used to show that our decoding algorithm is unlikely to time-out (i.e., there will be very few inputs x where random sampling fails to produce a B' containing A and x such that $C'(B')$ is consistent with A). Next, the condition that consistent sets B' are usually mostly correct implies that, conditioned on our algorithm sampling a consistent set B', the value $C'(B')|_x$ is likely to be correct; intuitively, for a random input x and a random consistent set B' containing x, this x is random within B', and so $C'(B')|_x$ is likely correct.

Next we give a formal argument. We need the following definitions.

DEFINITION 3.5 (correct edges). *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 $\epsilon' = \epsilon/2$.

DEFINITION 3.6 (good edges). *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.*

DEFINITION 3.7 (excellent edges). *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 other 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.

The following is the main result of this subsection.

LEMMA 3.8 ("excellence implies correctness"). *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 $\lambda \leqslant \epsilon'/2$, $\alpha \leqslant \beta/4$, 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 3.8, 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. Second, 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 3.9. *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 $\mathbf{Pr}_{x \in U \setminus A, y \in N(x)}[x \in F' \ \& \ y \in \mathit{Cons}] < \beta\mu = \beta(\epsilon' - \lambda)/2$.

On the other hand, we know that *Cons* has density at least ϵ' (by the definition of goodness of (A, B)). By Lemma 2.13, 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 3.10. *Suppose (A, B) is α-excellent, and the $\mathcal{T}$-graph G is a (β, λ)-sampler. Further suppose that $\alpha \leqslant \beta/4$ 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 $G' = G(U \setminus A, \mathit{Cons})$ 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 $x \in \mathit{Err}(B')$, and we 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 $\alpha D|\mathit{Cons}|$, by the definition of α-excellence.

By Lemma 2.14, we conclude that $\mathbf{Pr}_{x \in U \setminus A, B' \in N_{G'}(x)}[x \in \mathit{Err}(B')] \leqslant \max\{2\alpha/(1 - \lambda/\rho), \beta\}$. By assumption, $1 - \lambda/\rho \geqslant 1 - \lambda/\epsilon' \geqslant 1/2$, and thus $2\alpha/(1 - \lambda/\epsilon') \leqslant 4\alpha \leqslant \beta$. ∎

Now we can finish the proof of Lemma 3.8.

Proof of Lemma 3.8. Lemma 3.9 implies that 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 3.10, 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 biregular and if the $\mathcal{S}$-graph H is a sampler, then a random edge (A, B) of I will be excellent with probability $\Omega(\epsilon)$. That is, a random choice is likely to satisfy the sufficient conditions for the circuit $C_{A,v}$ being correct.

For intuition, remember that a random edge is correct with probability at least ϵ, by the assumption on the oracle C'. By a (careful) averaging argument, one can show that for $\epsilon/2$ fraction of edges (A, B), it is the case that both (A, B) is correct, and there are at least $\epsilon/2$ fraction of correct edges (A, B'). That is, a random edge (A, B) is good with probability at least $\epsilon/2$.

For excellence, we use the symmetry in our choices of (A, B'): we can either choose a random $k/2$-set A first and then a random k-set B' containing A, or choose a random k-set B' first and then a random $k/2$-subset A inside B'. Let us consider a random good edge (A, B) and a random consistent set B'. Suppose that $C'(B')$ has many errors. By symmetry, even though (A, B) was chosen first and then we chose B', we can think of B' chosen first and then A chosen randomly inside B' (and B being a random set containing A). If B' has many inputs where $C'(B')$ is wrong, then its random subset A is also very likely to contain some of these inputs. But then it

cannot be that (A, B) is both correct and consistent with B'.

Now we give a formal argument. The following is the main result of this subsection.

LEMMA 3.11 ("excellence is abundant"). *Suppose the inclusion graph I is biregular and the $\mathcal{S}$-graph H is a $\nu(\cdot)$-sampler.*[6] *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 3.12. *A random edge (A, B) of a biregular 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. ∎

Now we can prove Lemma 3.11.

Proof of Lemma 3.11. To show that a good 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

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

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

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

[6] Here we need only 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 $\nu(\mu)$.

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

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

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, $\mathbf{Pr}_{A \in \mathcal{S}, B \in N(A)}[(A, B)$ is α-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 3.12, the first probability in the difference above is at least ϵ', and, by what we showed earlier, the second probability is at most $\epsilon'/2$. Therefore, the lemma follows. □

Proof of Theorem 3.2. As explained in the proof outline earlier, we will use Lemmas 3.8 and 3.11. We set $\beta = \delta/16$, $\lambda = \epsilon'/2$, $\alpha = \beta/4 = \delta/64$, and $\nu(\mu) = \alpha\epsilon'^2/(4\mu) = \delta\epsilon'^2/(256\mu)$. This choice of parameters satisfies the assumptions of both lemmas, and the proof of Theorem 3.2 easily follows from the lemmas. □

4. Extensions.

4.1. Approximate version of the uniform direct product theorem.

In this section, we prove Theorem 1.8, which we restate below.

THEOREM 4.1 (Theorem 1.8 restated). *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:* $\delta = O(1/(\epsilon^2 k^{1/4}))$.

The proof is along the same lines as that of Theorem 1.6 given in the previous section. We just need to make the following modifications in our definitions. 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 thus could be somewhat inconsistent. We need to redefine consistency to allow a small number of contradictory values, and then show that any very incorrect $C'(B')$ will have too many inconsistent values with high probability.

Recall that for $B \in \mathcal{T}$, $Err(B)$ is the set of those $x \in B$, where $C'(B)|_x \neq f(x)$, and $err(B) = |Err(B)/|B|$. We say that a set $B \in \mathcal{T}$ is δ'-*correct* if $err(B) \leqslant \delta'$ (i.e., $C'(B)$ and $f^k(B)$ disagree on at most δ' fraction of elements of B). An edge (A, B) of the inclusion graph I is called δ'-*correct* if B is δ'-correct and $|A \cap Err(B)| \leqslant 2\delta'|A|$.

For this section, we set $\epsilon' = \epsilon/4$. Call an edge (A, B) of I *good* if it is δ'-correct and at least ϵ' fraction of all neighbors B' of A are δ'-correct.

The definition of consistency changes as follows. Two neighbors B, B' of A are called *consistent* if $C'(B)|_A$ and $C'(B')|_A$ disagree on at most $4\delta'$ fraction of elements in A. Note that for any two δ'-correct edges (A, B) and (A, B'), we have that B and B' are consistent. As before, for a given edge (A, B), we denote by *Cons* the set of all B' that are consistent with B. Finally, the definition of an excellent edge is as before: An edge (A, B) is α-*excellent* if it is good, and moreover, $\mathbf{Exp}_{B' \in Cons}[err(B')] \leqslant \alpha$.

Next we need to verify that with these modifications in the definitions, all lemmas of the previous section go through. It is straightforward to check that all lemmas in section 3.1 continue to hold (with the same proofs) with respect to these new definitions.

For the lemmas of section 3.2, we need to argue that a random edge (A, B) is excellent with probability $\Omega(\epsilon)$. For this, we need an analogue of Lemma 3.12.

LEMMA 4.2. *Suppose the inclusion graph I is biregular, and the $\mathcal{S}$-graph H is a $(\delta', 1/2)$-sampler. Then a random edge (A, B) of the inclusion graph I is good with probability at least ϵ'.*

Proof. We choose a random edge (A, B) of I by choosing a random $B \in \mathcal{T}$ first, and then choosing a random $A \in N(B)$. By the assumption on the circuit C', the probability that a random $B \in T$ is δ'-correct is at least ϵ. For every fixed δ'-correct set B, the sampling property of the $\mathcal{S}$-graph implies that $\mathbf{Pr}_{A \in N(B)}[|A \cap Err(B)| > 2\delta'|A|] \leqslant 1/2$. It follows that a random edge (A, B) is δ'-correct with probability at least $\epsilon/2$.

Similarly to the proof of Lemma 3.12, let $P(A)$ be the event that $\mathbf{Pr}_{B' \in N(A)}[(A, B')$ is δ'-correct$] < \epsilon/4$. We get that

$$\mathbf{Pr}_{A \in \mathcal{S}, B \in N(A)}[((A, B) \text{ is } \delta'\text{-correct}) \ \&\ P(A)] < \epsilon/4.$$

Finally, the probability that (A, B) is good is equal to the probability that it is δ'-correct, minus the probability that it is δ'-correct and the event $P(A)$ happens. The former is $\epsilon/2$, and the latter is less than $\epsilon/4$. Thus (A, B) is good with probability at least $\epsilon/4$, as required. □

We have the following analogue of Lemma 3.11.

LEMMA 4.3. *Suppose the inclusion graph I is biregular, and the $\mathcal{S}$-graph H is a $\nu(\cdot)$-sampler. Assume that $1 \geqslant \alpha \geqslant 24\delta'$ is such that for every $1 \geqslant \mu > \alpha/2$, $\mu \cdot \nu(\mu) < \alpha\epsilon'^2/4$. Then a random edge (A, B) of I is α-excellent with probability at least $\epsilon'/2$.*

Proof sketch. Compared with the proof of Lemma 3.11, the only change is in the argument to upperbound $\mathbf{Pr}[E(A, B, B', x)]$. This is modified as follows. Condition on any set $B' \in \mathcal{T}$ that is μ-incorrect for $\mu > \alpha/2$. By the sampling property of the $\mathcal{S}$-graph, the probability that a random neighbor $A \in N(B')$ has less than $\mu/2$ fraction of elements from $Err(B')$ is at most $\nu(\mu)$. Consider any fixed A that has more than $\mu/2$ fraction of elements from $Err(B')$. For any neighbor B of A such that B is consistent with B', we have that A contains more than $(\mu/2 - 4\delta')|A|$ elements from $Err(B)$, which is more than $2\delta'|A|$ for $\mu > \alpha/2 \geqslant 12\delta'$, and so the edge (A, B) is not δ'-correct.

This implies that the conditional probability $\mathbf{Pr}[E(A,B,B',x) \mid B'] \leqslant \mu \cdot \nu(\mu)$. The rest of the proof is exactly the same as that of Lemma 3.11. □

With the lemmas above, we get the proof of Theorem 1.8 in the same way as the proof of Theorem 1.6 for $\delta' \geqslant \Omega(\delta)$.

4.2. Derandomized direct product theorems. Here we will prove Theorem 1.9, restated next.

THEOREM 4.4 (Theorem 1.9 restated). *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 $\mathsf{BPTIME}(t)//\log t$, then the function h defined from f as described above is $(1-\epsilon)$-hard for $\mathsf{BPTIME}(t^{1/c})//(1/c)\log t$. The input size of h is $O(n)$.*

Proof. For $K = \mathrm{poly}(1/\epsilon)$ and $k = O(\log 1/\epsilon)$, let $\mathcal{K}$ denote the collection of all k-subsets of $\{1,\dots,K\}$. We need to analyze the function $h : \mathcal{T} \times \mathcal{K} \to \{0,1\}^k$ mapping $(T, i_1, \dots, i_k)$ to $g(T)|_{i_1,\dots,i_k}$, 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 $\{1,\dots,K\}$ 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., [STV01]). 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 T's.

By Theorem 1.6, 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 $g(T)$.

For each $T \in \mathcal{T}$, let us order the strings returned by our approximate list-decoding algorithm on input $C'(T,\cdot)$. Define a list of ℓ circuits $C''_1, \dots, C''_\ell$ for $g(T)$, where $C''_i(T)$ outputs the ith K-bit string on the list corresponding to T. By averaging, there is some $1 \leqslant i \leqslant \ell$ such that $C''_i(T)$ will $(1-\delta')$-agree with $g(T)$ for at least $1/\ell$ fraction of inputs $T \in \mathcal{T}_{good}$, which is at least $\Omega(\epsilon^2)$ fraction of all inputs T to g. Let us call such a circuit C''_i approximately good for g.

By Theorem 1.8, the Subspaces intersection code is (δ,δ')-approximately $(\Omega(\epsilon^2), O(1/\epsilon^2))$–list-decodable. Thus, for each of our ℓ circuits $C''_1, \dots, C''_\ell$, we efficiently get $O(1/\epsilon^2)$ new circuits such that, if C''_i is an approximately good circuit for g, then the list of circuits obtained from that C''_i will have a circuit $(1-\delta)$-computing f. Overall, we efficiently construct a list of $O(\ell/\epsilon^2) = O(1/\epsilon^3)$ circuits for f, one which will $(1-\delta)$-compute f. Hence, f is not δ-hard for $\mathsf{BPTIME}(t)//\log t$, which is a contradiction. □

4.3. Hardness condensing. In this subsection, we reinterpret the results of the previous section to give a version of hardness condensing for the semiuniform model, proving Theorem 1.10, which we restate below.

THEOREM 4.5 (Theorem 1.10 restated). *For an absolute constant $c > 0$, 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 $f|_B$.

Think of the sets B as being exponentially large but succinctly representable (as in the subspace construction for large values of $k = q^d$). Before, we assumed to have an oracle C' such that $C'(B)$ is supposed to output the tuple of values of f on all (or most) elements in B. Now we will think of C' as an algorithm that, on input B, outputs a circuit V_B such that $V_B(x) = f(x)$ on all (or most) $x \in B$. That is, instead of being given the k-tuple of values explicitly, we are now given these values implicitly, via a circuit which presumably computes the function $f|_B$.

We can still carry out the steps of our decoding algorithm in this new setting. For example, to check the (approximate) consistency of C' on two sets B and B' containing some common subset A, we simply use random sampling to estimate the fraction of elements $x \in A$, where $V_B(x) \neq V_{B'}(x)$. Thus, if f is hard, we can conclude that for almost all B, the restricted function $f|_B$ is about as hard as f. Since f_B is defined on fewer inputs than f, but has about the same average-case hardness as f (for most B), this can be viewed as a randomized procedure for hardness condensing, with respect to the semiuniform model of computation.

To get a precise statement of this idea, we extend the definition of semiuniform time of [TV02] to *families* of functions. Consider a family of functions $f_h(x) = F(h,x)$, where $h \in \mathcal{H}$ and $x \in U = \{0,1\}^n$. Call the family f_h *$(1-\epsilon)$-hard to δ-compute in semiuniform time t* if, for any time $t(|h|+n)$ probabilistic algorithm $A(h,r)$ which produces a circuit $C_{h,r}$ on n bit inputs x, we have $\mathbf{Pr}_{h,r}[C_{h,r}\ \delta\text{-computes } f_h] \leq \epsilon$.

Assume $\mathcal{S}$ and $\mathcal{T}$ meet the conditions of Theorem 3.2, and that furthermore, we can describe $B \in \mathcal{T}$ and $A \in \mathcal{S}$ as strings of length n_1, and sample uniformly from either, using at most n_2 random bits, in time polynomial in $n_1 + n_2$. For x of length n_2, let $f_B(x)$ be f applied to the random element of B obtained by using string x in the sampling algorithm. (For example, in the case that B is a d-dimensional affine subspace of $(\mathbb{F}_q)^m$, we can represent B by its basis and skew vectors $b_1, \ldots, b_d, v$, with $n_1 = (d+1)m \log q$ bits. Then with $n_2 = d \log q$ bits, we can sample from B by picking random $\alpha_1, \ldots, \alpha_d$ and letting $y = \alpha_1 b_1 + \cdots + \alpha_d b_d + v$. Then $f_{b_1,\ldots,b_d,v}(\alpha_1, \ldots, \alpha_d) = f(\alpha_1 b_1 + \cdots + \alpha_d b_d + v)$.)

Then by altering the previous proof of Theorem 1.8 as specified above, we have the following theorem.

THEOREM 4.6. *Let $\mathcal{S}$, $\mathcal{T}$, δ, ϵ meet the conditions of Theorem* 3.2 *and be efficiently describable and sampleable as above. There is a constant c so that if f is δ-hard for $\mathsf{BPTIME}(t(n))//\log t(n)$, and $\epsilon > t(n)^{-1/c}$, then the family f_B is $(1-\epsilon)$-hard to $(1-\Omega(\delta))$-compute in semiuniform time $t(n)^{1/c}$.*

The only difference is that the algorithm C', on set B, generates a circuit V_B rather than values v. The advice becomes (A, V_B), and when we generate B' with $A \cup x \subseteq B'$, we use the algorithm to compute the circuit $V_{B'}$, and then we estimate consistency by randomly sampling $O((\log 1/\epsilon)/\delta^2)$ elements $a \in A$ and seeing for how many $V_B(a) \neq V_{B'}(a)$.

Theorem 1.10 is equivalent to the following corollary of Theorem 4.6.

COROLLARY 4.7. *Let $\mathcal{T}$ be the family of affine subspaces of dimension d of $\mathbb{F}_q^m$, where $d \geq 8$. For some absolute constant c, if f is δ-hard for $\mathsf{BPTIME}(t(n))//\log t(n)$, then the family $f|_B$ for $B \in \mathcal{T}$ is $(1-\epsilon)$-hard to $(1-\Omega(\delta))$-compute in semiuniform time $t(n)^{1/c}$ for $\epsilon = \max\{q^{-d/16}, t(n)^{-1/c}\}$. Moreover, each $f|_B$ is equivalent to a function on $d \log q$ bit inputs.*

Finally, we observe that Corollary 4.7 can be used to prove the following deran-

domized hardness amplification result.

THEOREM 4.8. *Let $\delta > 0, 2^{\sqrt{n}+1} \geq q \geq 2^{\sqrt{n}}$, and let $\mathcal{T}$ be the family of affine subspaces of dimension $d = 8$ of $\mathbb{F}_q^m$, let $k(n) = O(\sqrt{n}/\delta)$, and let $t(n) \leq 2^{\sqrt{n}}$. For some absolute constant c, if f is δ-hard for $\mathsf{BPTIME}(t(n))//\log t(n)$, then the function $g(B, y_1, \ldots, y_k) = (f|_B)^k(y_1, \ldots, y_k)$ for $B \in \mathcal{T}$ and $y_1, \ldots, y_k \in B$, is $1 - t(n)^{-1/c}$ hard for $\mathsf{BPTIME}(t(n)^{1/c})//(1/c)\log t(n)$. Moreover, g is equivalent to a function on $O(n)$ bits.*

Proof. Assume we have an algorithm that with probability $\epsilon > t(n)^{1/c} > e^{-k(n)\delta/c}$ produces a circuit that ϵ-computes $g = (f|_B)^k$ in time $t'(n) = t(n)^{1/c}$. Then for each of the $\epsilon/2$ B's where the conditional probability of success for the circuit is at least $\epsilon/2$, we can use the list decoder for our Independent code to get a circuit $1 - \Omega(\delta)$ computing f_B in time $t'(n)/\text{poly}(\epsilon)$. In other words, the family $f|_B$ has a semiuniform algorithm that $1 - \Omega(\delta)$-computes it with probability $\text{poly}(\epsilon)$. By Theorem 4.6, f has a semiuniform time $t'(n)/\text{poly}(\epsilon)$ algorithm that $(1 - \delta)$-computes f with $\text{poly}(\epsilon)$ success, a contradiction to the assumed hardness. ∎

5. k-XOR code. Here we prove Theorem 1.11, restated below.

THEOREM 5.1 (Theorem 1.11 restated). *The k-XOR code is efficiently, locally, δ-approximately $(1/2 + \epsilon, O(1/\epsilon^2))$–list-decodable for $\delta = O((\log 1/\epsilon)/k)$.*

As a warm-up, we first list-decode a code which is a concatenation of our Independent code and the standard Hadamard code. The proof of Theorem 1.11 will follow along similar lines.

Let Ind_k be Independent k-wise direct product code. Let Had_k be the Hadamard code on messages of size k; i.e., for every message $msg \in \{0,1\}^k$, the encoding $Had_k(msg)$ is a function mapping a string $r \in \{0,1\}^k$ to the inner product $\langle msg, r\rangle$ over the binary field $\mathbb{F}_2$. Define $Code_k$ to be the concatenation of Ind_k and Had_k, i.e., $Code_k(f)$ is a function mapping $(x_1, \ldots, x_k; r)$ to $\sum_{i=1}^k f(x_i) \cdot r_i \mod 2$ for $x_i \in \{0,1\}^n$ and $r \in \{0,1\}^k$.

We will list-decode this code using the algorithm of [GL89] for the Hadamard code and our algorithm $\mathcal{A}$ for the Independent code. First we state the result of Goldreich and Levin.

THEOREM 5.2 (see [GL89]). *There is a probabilistic algorithm A with the following property. Let $h \in \{0,1\}^k$ be any string, and let $B : \{0,1\}^k \to \{0,1\}$ be any predicate such that $|Pr_{r\in\{0,1\}^n}[B(r) = \langle h, r\rangle] - 1/2| \geqslant \gamma$ for some $\gamma > 0$. Then, given oracle access to B and given γ, the algorithm A runs in time poly$(k, 1/\gamma)$ and outputs a list of size $l = O(1/\gamma^2)$ such that with high probability the string h is on this list.*

Using the Goldreich–Levin algorithm of Theorem 5.2, we will show the following.

THEOREM 5.3. *The code $Code_k$ is efficiently, locally, δ-approximately $(1/2 + \epsilon, O(1/\epsilon^2))$–list-decodable for $\delta = O(\log 1/\epsilon/k)$.*

Proof. Let C' be the circuit which $(1/2 + \epsilon)$-computes $Code_k(f)$. For a given k subset $\bar{x} = (x_1, \ldots, x_k)$, define $\gamma_{\bar{x}} = \mathbf{Pr}_r[\langle f^k(\bar{x}), r\rangle = C'(\bar{x}; r)] - 1/2$. Clearly, we have $\mathbf{Exp}_{\bar{x}}[\gamma_{\bar{x}}] \geqslant \epsilon$ (since C' $(1/2 + \epsilon)$-computes $Code_k(f)$).

For a given $\bar{x} = (x_1, \ldots, x_k)$, we set $h = f^k(\bar{x})$ and $B(r) = C'(\bar{x}; r)$ and run the Goldreich–Levin algorithm with $\gamma = \epsilon/2$. For every $\bar{x}$ with $|\gamma_{\bar{x}}| \geqslant \epsilon/2$, the Goldreich–Levin algorithm will return a list $h_1, \ldots, h_l$ of size $l = O(1/\epsilon^2)$ that, with high probability, contains h.

For each h_i on the list, define $\gamma_{\bar{x},i} = Pr_r[\langle h_i, r\rangle = C'(\bar{x}; r)] - 1/2$. By random sampling, we can efficiently estimate each $\gamma_{\bar{x},i}$ to within a constant factor, with high probability. Let $\tilde{\gamma}_{\bar{x},i}$ denote the corresponding approximation. We will choose string h_i with probability proportionate to $(\tilde{\gamma}_{\bar{x},i})^2$, i.e., with probability $(\tilde{\gamma}_{\bar{x},i})^2 / \sum_{j=1}^l (\tilde{\gamma}_{\bar{x},j})^2$.

For the analysis, first observe that $2\gamma_{\bar{x},i}$ is the discrete Fourier coefficient at h_i of the Boolean function $C'(\bar{x},\cdot)$. By Parseval's identity, we have $\sum_{j=1}^{l} 4\cdot\gamma_{\bar{x},i}^2 \leqslant 1$. Assuming that we have constant-factor approximations of all $\gamma_{\bar{x},i}$'s and that h was on the list, we conclude that the described algorithm outputs h with probability $\Omega(\gamma_{\bar{x}}^2)$. Since the assumed two events happen with high probability, we get that the probability of producing h is at least $\alpha\cdot\gamma_{\bar{x}}^2$ for some absolute constant $\alpha>0$.

Denote by X the set of all inputs $\bar{x}$, and by G the set of those $\bar{x}$ where $|\gamma_{\bar{x}}| \geqslant \epsilon/2$. The probability (over a random $\bar{x}$ and internal randomness) that the described algorithm outputs the correct string $f^k(\bar{x})$ is

$$(1/|X|)\sum_{\bar{x}\in G}\alpha\cdot\gamma_{\bar{x}}^2 \geqslant (1/|X|)\left(\sum_{\bar{x}\in X}\alpha\cdot\gamma_{\bar{x}}^2 - \sum_{\bar{x}\in X\setminus G}\alpha\cdot\gamma_{\bar{x}}^2\right).$$

The first term is α times $\mathbf{Exp}_{\bar{x}}[\gamma_{\bar{x}}^2] \geqslant (\mathbf{Exp}_{\bar{x}}[\gamma_{\bar{x}}])^2 \geqslant \epsilon^2$, by Cauchy–Schwarz and the lower bound $\mathbf{Exp}_{\bar{x}}[\gamma_{\bar{x}}] \geqslant \epsilon$. The second term is at most $\alpha\cdot\epsilon^2/4$ by the definition of G. So the overall success probability of the described algorithm at computing f^k is at least $\Omega(\epsilon^2)$.

Finally, we apply Theorem 1.6 to the described algorithm for f^k, concluding that the code $Code_k$ is efficiently, locally, δ-approximately $(1/2+\epsilon, O(1/\epsilon^2))$–list-decodable for $\delta = O((\log 1/\epsilon)/k)$. □

To prove Theorem 1.11, we will show how to list-decode the code obtained by concatenating Ind_{2k} with the truncated Hadamard code $Had_{2k,k}$, where the given $2k$-bit message msg is encoded by the string of inner products $\langle msg, r\rangle \bmod 2$, over all $2k$-bit strings r of Hamming weight exactly k. More precisely, we consider the following code $Code(x_1,\ldots,x_{2k};r) = \sum_{i=1}^{2k} f(x_i)r_i \mod 2$, where $r\in\{0,1\}^{2k}$ have Hamming weight exactly k.

First we observe that given a circuit C which $(1/2+\epsilon)$-computes the k-XOR encoding of f, the following circuit C' will $(1/2+\epsilon)$-compute the encoding $Code$ defined above: "Given $(x_1,\ldots,x_{2k};r)$ for $x_i\in\{0,1\}^n$ and $r\in\{0,1\}^{2k}$ of Hamming weight k, let $y_1,\ldots,y_k$ be the subset of $(x_1,\ldots,x_{2k})$ corresponding to the k positions i, where $r_i=1$. Output the value $C(y_1,\ldots,y_k)$."

Indeed, for uniformly random $2k$-subsets $(x_1,\ldots,x_{2k})$ and a random string $r\in\{0,1\}^{2k}$ conditioned on having Hamming weight exactly k, our circuit C' runs the circuit C on a uniformly random k-subset $(y_1,\ldots,y_k)$, and hence outputs the value $\oplus_{i=1}^k f(y_i) = Code_{2k}(f)(x_1,\ldots,x_{2k};r)$ with probability at least $1/2+\epsilon$.

We can also get a circuit C'' that $(1/2+\Omega(\epsilon/\sqrt{k}))$-computes the code $Code_k$ defined earlier: "Given $(x_1,\ldots,x_{2k};r)$ for $x_i\in\{0,1\}^n$ and $r\in\{0,1\}^{2k}$, output a random bit if the Hamming weight of r is not k. Otherwise, let $y_1,\ldots,y_k$ be the subset of $(x_1,\ldots,x_{2k})$ corresponding to the k positions i, where $r_i=1$. Output the value $C(y_1,\ldots,y_k)$." For the analysis, simply observe that a random $2k$-bit string will have Hamming weight k with probability $\Omega(1/\sqrt{k})$. Conditioned on r being of weight k, we get a correct answer with probability $1/2+\epsilon$; otherwise, we are correct with probability $1/2$.

Applying Theorem 5.3 to the circuit C'' will yield a list of $O(k/\epsilon^2)$ circuits, one of which $(1-\delta)$-computes f.

To get the optimal $O(1/\epsilon^2)$ list size, we will *approximately* list-decode the inner, truncated Hadamard code in $Code$. The idea is as follows. We will mimic the proof of Theorem 5.3 to argue that with probability $\Omega(\epsilon^2)$ over random $2k$-tuples $(x_1,\ldots,x_{2k})$ and internal randomness, one can produce a $2k$-tuple of bits $(b_1,\ldots,b_{2k})$ such that for

all but $O(\delta)$ fraction of indices $i \in [2k]$, we have $f(x_i) = b_i$. Running the approximate list-decoder of Theorem 1.8, we then get a list of $O(1/\epsilon^2)$ algorithms, one of which $(1-\delta)$-computes f.

We need the following.

LEMMA 5.4. *Let $a = (a_1, \ldots, a_{2k})$ be any $2k$-bit string, and let B be a function mapping $2k$-bit strings r of Hamming weight k to $\{0,1\}$ such that $\mathbf{Pr}_r[\langle a, r\rangle = B(r)] = 1/2 + \eta$ for some unknown η. Suppose we are given $\gamma > 0$ such that $|\eta| \geqslant \gamma$. Then there is an algorithm that, given γ and oracle access to B, runs in time $\mathrm{poly}(k, 1/\gamma)$ and, with probability $\Omega(\eta^2)$, outputs a $2k$-bit string that agrees with a in all but at most δ' fraction of positions for $\delta' = O((\log 1/\gamma)/k)$.*

Proof. Given B, we define the following randomized algorithm B' mapping $2k$-bit strings to $\{0,1\}$: "On a given r, if the Hamming weight of r is k, output $B(r)$; otherwise, output a random bit."

It is easy to see that $B'(r)$ will agree with the Hadamard encoding $Had_{2k}(a)$ at r for at least $1/2 + \Omega(\eta/\sqrt{k})$ fraction of $2k$-bit strings r. Running the Goldreich–Levin list-decoding algorithm on this B' with the agreement parameter $\Omega(\gamma/\sqrt{k})$, we get with high probability a list of at most $\ell = O(k/\gamma^2)$ strings $h_1, \ldots, h_\ell$, which contains our string a. Next we describe an algorithm for producing a string approximately equal to a, with probability $\Omega(\eta^2)$.

For each $i \in [\ell]$, define $\eta_i = \mathbf{Pr}_r[\langle h_i, r\rangle = B(r)] - 1/2$, where the probability is over $2k$-bit strings r of Hamming weight k. By random sampling, we can estimate each η_i to within a constant factor, with high probability. Let $\tilde{\eta}_i$ denote the respective approximations.

Let us order $|\tilde{\eta}_i|$'s from largest to smallest, and let us discard all those strings h_i, where $|\tilde{\eta}_i| < \gamma/2$. For the remaining strings, assume w.l.o.g. that $|\tilde{\eta}_1| \geqslant \cdots \geqslant |\tilde{\eta}_{\ell'}|$. We partition the strings' h_i's into groups as follows: Let B_1 be the set of all strings of Hamming distance at most δ from h_1; we call h_1 a leader of cluster B_1. Remove all strings B_1 from our list. Let h_j be the first remaining string (according to the order on $\tilde{\eta}_i$'s). Define B_2 to be the set of all remaining strings of Hamming distance at most δ from h_j; here h_j is a leader of B_2. Remove B_2 from the list, and continue until all strings are partitioned into disjoint clusters $B_1, B_2, \ldots, B_t$. For simplicity of notation, assume that the leaders of these clusters are $h_1, h_2, \ldots, h_t$.

Finally, output a leader h_i with probability $\tilde{\eta}_i^2 / \sum_{j=1}^t \tilde{\eta}_j^2$.

For the analysis, we will need the following claim.

CLAIM 5.5. $\sum_{i=1}^t \eta_i^2 \leqslant 1/2$.

Proof. The idea of the proof is the following. The truncated Hadamard code $Had_{2k,k}$ maps any two far-apart messages msg_1 and msg_2 to the codewords $code_1$ and $code_2$ that are almost Hamming distance $1/2$ apart. Switching from the $\{0,1\}$ alphabet to the $\{1,-1\}$ alphabet, the previous statement means that the normalized inner product $\mathbf{Exp}_r[code_1(r) \cdot code_2(r)]$ of the vectors $code_1$ and $code_2$ is close to 0, where the expectation is over $2k$-bit strings of weight k.

Thus the encodings $y_1, \ldots, y_t$ of the leaders $h_1, \ldots, h_t$, respectively, are pairwise almost orthogonal. It is also easy to see that $2\eta_i = \mathbf{Exp}_r[y_i(r) \cdot B(r)]$, and so η_i's are the projections of the vector B onto vector y_i. If the y_i's were pairwise orthogonal, we would get that $B = \sum_{i=1}^t (2\eta_i) \cdot y_i + B^\perp$, where $B^\perp$ is orthogonal to every y_i for $i \in [t]$. Taking the normalized inner product of B with itself, we would get $\mathbf{Exp}_r[(B(r))^2] = \sum_{i=1}^t (2\eta_i)^2 + \mathbf{Exp}_r[(B^\perp(r))^2]$. Since $(B(r))^2 = 1$ for every r, we conclude that $\sum_{i=1}^t (2\eta_i)^2 \leqslant 1$.

In reality, the vectors' y_i's are pairwise almost orthogonal, and so the calculations

will be slightly more complicated but will follow the same idea. For notational convenience, denote $\alpha_i = 2\eta_i$. Let us write the vector $B = (\sum_{i=1}^{t} \alpha_i \cdot y_i) + (B - \sum_{i=1}^{t} \alpha_i \cdot y_i)$. Also for notational convenience in the rest of the proof, let us denote by $\langle B, y_i \rangle$ the normalized inner product $\mathbf{Exp}_r[B(r) \cdot y_i(r)]$. We have

$$1 = \langle B, B \rangle = \sum_{i,j} \alpha_i \alpha_j \cdot \langle y_i, y_j \rangle + 2 \left\langle \sum_{i=1}^{t} \alpha_i \cdot y_i, B - \sum_{i=1}^{t} \alpha_i \cdot y_i \right\rangle + \left\langle B - \sum_{i=1}^{t} \alpha_i \cdot y_i, B - \sum_{i=1}^{t} \alpha_i \cdot y_i \right\rangle.$$

The last term on the right-hand side is nonnegative, and after dropping it we get the following:

$$1 \geqslant 2 \sum_i \alpha_i^2 - \sum_{i,j} \alpha_i \alpha_j \cdot \langle y_i, y_j \rangle = \sum_i \alpha_i^2 - \sum_{i \neq j} \alpha_i \alpha_j \cdot \langle y_i, y_j \rangle.$$

Hence, $\sum_i \alpha_i^2 \leqslant 1 + \sum_{i \neq j} \alpha_i \alpha_j \cdot \langle y_i, y_j \rangle$. Since $|\alpha_i| \leqslant 1$ for all i, the latter is at most $1 + t^2 \cdot \max_{i \neq j} \{\langle y_i, y_j \rangle\}$.

To finish the proof, we need to upperbound t and $\max_{i \neq j} \{\langle y_i, y_j \rangle\}$. We start with the latter. Consider any two $2k$-bit messages msg_1 and msg_2 that differ in at least δ' fraction of positions. Then the normalized inner product of their respective encodings (in the $\{1, -1\}$ alphabet) will be $\mathbf{Exp}_r[(-1)^{\langle msg_1 \oplus msg_2, r \rangle}]$, where r ranges over all $2k$-bit strings of Hamming weight k. Using the Chernoff–Hoeffding bounds, this expectation can be upperbounded by $e^{-\Omega(\delta' k)}$.

The bound on t can be obtained by the Johnson bound: if $y_1, \ldots, y_t$ have pairwise inner products at most $e^{-\Omega(\delta' k)}$ in absolute value, and each $|\langle y_i, B \rangle| \geqslant \gamma$ for $i \in [t]$, then $t \leqslant 1/(\gamma^2 - e^{-\Omega(\delta' k)})$ (see, e.g., [IJK06]). For $\delta' = d(\log 1/\gamma)/k$ for a large enough constant d, we get that $t^2 \cdot \max_{i \neq j} \{\langle y_i, y_j \rangle\} \leqslant 1$. The claim follows. ∎

Suppose that our string a was put in a cluster with a leader h_i. This means that the (approximation of the) agreement of the truncated Hadamard encoding of h_i with $B(r)$ is at least as big as that of a with $B(r)$ (in absolute values), and that a and h_i are at most δ' Hamming distance apart. We get by the claim above that h_i is output with probability $\Omega(\eta^2)$, as required. ∎

Proof of Theorem 1.11. Let *Code* be the concatenation of Ind_{2k} and the truncated Hadamard $Had_{2k,k}$. As explained earlier, from a circuit C $(1/2 + \epsilon)$-computing $f^{\oplus k}$, we can get C' that $(1/2 + \epsilon)$-computes *Code*. For each $2k$-subset $\bar{x} = (x_1, \ldots, x_{2k})$, let $\epsilon_{\bar{x}} = \mathbf{Pr}_r[\langle f^k(\bar{x}), r \rangle = C'(\bar{x}, r)] - 1/2$. Clearly, $\mathbf{Exp}_{\bar{x}}[\epsilon_{\bar{x}}] \geqslant \epsilon$.

For a given $\bar{x}$, let $a = f^{2k}(\bar{x})$, and let $B(r) = C'(\bar{x}, r)$. Run the algorithm of Lemma 5.4 on this $B(r)$, with the parameter $\gamma = \epsilon/2$. If $|\epsilon_{\bar{x}}| \geqslant \gamma$, then we will get with probability $\Omega(\epsilon_{\bar{x}}^2)$ a $2k$-bit string that agrees with a in all but at most δ' positions for $\delta' = O((\log 1/\epsilon)/k)$.

As in the proof of Theorem 5.3 above, we then obtain a randomized algorithm for Ind_{2k} that with probability at least $\Omega(\epsilon^2)$ (where the probability is over $\bar{x}$ and the internal randomness of the algorithm) outputs a string that is at most δ' distance away from $f^{2k}(\bar{x})$. Running the approximate list-decoding algorithm for Ind_{2k} from Theorem 1.8, we obtain a list of $O(1/\epsilon^2)$ circuits, one of which $(1 - \delta)$-computes f for $\delta \leqslant O(\delta')$. ∎

6. 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 (in uniform randomized AC^0) and has a simple analysis. We also defined a natural generalization of direct product codes and intersection codes for families of subsets $(\mathcal{S}, \mathcal{T})$, and we 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 1.8), ideally achieving a uniform version of the "Chernoff-type" direct product theorem in the spirit of [IJK09]. Finally, it is interesting to see whether the ideas from our new list-decoding algorithm can help in improving the known uniform hardness amplification results for NP of [Tre05].

REFERENCES

[AS03] S. Arora and M. Sudan, *Improved low-degree testing and its applications*, Combinatorica, 23 (2003), pp. 365–426.

[BGG93] M. Bellare, O. Goldreich, and S. Goldwasser, *Randomness in interactive proofs*, Comput. Complexity, 3 (1993), pp. 319–354.

[BOS06] J. Buresh-Oppenheim and R. Santhanam, *Making hard problems harder*, in Proceedings of the Twenty-First Annual IEEE Conference on Computational Complexity, Prague, 2006, pp. 73–87.

[DR06] I. Dinur and O. Reingold, *Assignment testers: Towards a combinatorial proof of the PCP theorem*, SIAM J. Comput., 36 (2006), pp. 975–1024.

[GL89] O. Goldreich and L. A. Levin, *A hard-core predicate for all one-way functions*, in Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing, Seattle, 1989, pp. 25–32.

[GNW95] O. Goldreich, N. Nisan, and A. Wigderson, *On Yao's XOR-Lemma*, Technical report TR95-050, Electronic Colloquium on Computational Complexity, Potsdam, Germany, 1995.

[GS00] O. Goldreich and S. Safra, *A combinatorial consistency lemma with application to proving the PCP theorem*, SIAM J. Comput., 29 (2000), pp. 1132–1154.

[GGH+07] S. Goldwasser, D. Gutfreund, A. Healy, T. Kaufman, and G. N. Rothblum, *Verifying and decoding in constant depth*, in Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing, San Diego, 2007, pp. 440–449.

[Hoe63] W. Hoeffding, *Probability inequalities for sums of bounded random variables*, J. Amer. Statist. Assoc., 58 (1963), pp. 13–30.

[Imp95] R. Impagliazzo, *Hard-core distributions for somewhat hard problems*, in Proceedings of the Thirty-Sixth Annual IEEE Symposium on Foundations of Computer Science, Milwaukee, WI, 1995, pp. 538–545.

[Imp02] R. Impagliazzo, *Hardness as randomness: A survey of universal derandomization*, in Proceedings of the International Congress of Mathematicians, Vol. 3, Higher Ed. Press, Beijing, 2002, pp. 659–672.

[IJK06] 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, Berkeley, CA, 2006, pp. 187–196.

[IJK09] R. Impagliazzo, R. Jaiswal, and V. Kabanets, *Chernoff-type direct product theorems*, J. Cryptology, 22 (2009), pp. 75–92.

[IKW09] R. Impagliazzo, V. Kabanets, and A. Wigderson, *New direct-product testers and* 2*-query PCPs*, in Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing, Bethesda, MD, 2009, pp. 131–140.

[IW97] 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, El Paso, TX, 1997, pp. 220–229.

[Lev87] L. A. Levin, *One-way functions and pseudorandom generators*, Combinatorica, 7 (1987), pp. 357–363.

[RS97] 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, El Paso, TX, 1997, pp. 475–484.

[STV01] M. Sudan, L. Trevisan, and S. Vadhan, *Pseudorandom generators without the XOR lemma*, J. Comput. System Sci., 62 (2001), pp. 236–266.

[Tre05] L. Trevisan, *On uniform amplification of hardness in NP*, in Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing, Baltimore, MD, 2005, pp. 31–38.

[TV02] L. Trevisan and S. Vadhan, *Pseudorandomness and average-case complexity via uniform reductions*, in Proceedings of the Seventeenth Annual IEEE Conference on Computational Complexity, Montreal, 2002, pp. 103–112.

[Yao82] A. C. Yao, *Theory and applications of trapdoor functions*, in Proceedings of the Twenty-Third Annual IEEE Symposium on Foundations of Computer Science, Chicago, 1982, pp. 80–91.

[Zuc97] D. Zuckerman, *Randomness-optimal oblivious sampling*, Random Structures Algorithms, 11 (1997), pp. 345–367.