Reconstructive Dispersers and Hitting Set Generators[*]

Christopher Umans[†]
Department of Computer Science
California Institute of Technology
Pasadena, CA 91125

November 28, 2006

Abstract

We give a generic construction of an optimal hitting set generator (HSG) from any good "reconstructive" disperser. Past constructions of optimal HSGs have been based on such disperser constructions, but have had to modify the construction in a complicated way to meet the stringent efficiency requirements of HSGs. The construction in this paper uses existing disperser constructions with the "easiest" parameter setting in a black-box fashion to give new constructions of optimal HSGs without any additional complications.

Our results show that a straightforward composition of the Nisan-Wigderson pseudorandom generator that is similar to the composition in works by Impagliazzo, Shaltiel and Wigderson in fact yields optimal HSGs (in contrast to the "near-optimal" HSGs constructed in those works). Our results also give optimal HSGs that do not use any form of hardness amplification or implicit list-decoding – like Trevisan's extractor, the only ingredients are combinatorial designs and any good list-decodable error-correcting code.

[*]A preliminary version of this paper appeared in RANDOM 2005.

[†]email: umans@cs.caltech.edu. Supported by NSF CCF-0346991, BSF 2004329, a Sloan Research Fellowship, and an Okawa Foundation research grant.

1 Introduction

Derandomization is a very active area within complexity theory, whose goal is to prove the existence of generic and efficient deterministic simulations of probabilistic procedures. This general endeavor makes sense in many different settings, as there are several meaningful choices for exact definitions of "efficient" and "probabilistic procedure". This paper focuses on the derandomization of probabilistic polynomial-time; the ultimate goal in complexity terms is to prove BPP $=$ P.

A very natural route to proving BPP $=$ P is to construct a *pseudorandom generator* (PRG), which is a deterministic procedure that stretches a short, truly random string (the "seed") into a long "pseudorandom" string that is indistinguishable from a truly random string by polynomial-time procedures. The output of a PRG can thus be substituted for true randomness to obtain efficient simulations of BPP, by enumerating over all seeds. However, the existence of a uniform family of PRGs useful for derandomizing BPP implies circuit lower bounds that seem well beyond our current abilities to prove (and more recently, it has been shown that BPP $=$ P itself implies circuit lower bounds [KI04]). Thus, in the absence of circuit lower bounds, the goal is to construct PRGs *under a hardness assumption*, and then we can hope for a family of constructions that represent a "best-possible" tradeoff between the hardness assumption required and the deterministic simulation implied by the PRG. This general "hardness vs. randomness" paradigm began with [BM84, Yao82, NW94], and continued with a number of papers working toward a best-possible quantitative tradeoff [IW97, STV01, ISW99, ISW00, SU05, Uma03]. See the survey by Kabanets [Kab02] for a more complete history and an account of the current state of derandomization research.

If one wishes to derandomize one-sided-error probabilistic decision procedures, RP, then the natural associated derandomization object is a *hitting set generator* (HSG) (see Definition 2.4). Surprisingly, it was shown in [ACR98] (and refined in [ACRT99, BF99, GVW00]) that HSGs suffice to derandomize BPP, even though they are only "intended" for one-sided error. Optimal HSGs were first constructed in [SU05], while optimal PRGs were first constructed in [Uma03]; here "optimal" means that, up to a polynomial, the constructions cannot be improved without implying stronger hardness assumptions than were used to construct them in the first place (see [ISW03] for a more detailed justification of the term "optimal" in this context).

In this paper we construct optimal HSGs, which, as noted above, suffice for optimal "hardness vs. randomness" tradeoffs for BPP as well as RP . Our construction is cast as a completely generic procedure for converting an object we call a "reconstructive disperser" into a HSG. The construction is arguably simpler than previous constructions, and its modular description exposes certain useful features shared by known constructions of reconstructive dispersers. Conceptually, it is interesting to view this work as another example of the surprising connection between information-theoretic objects (e.g., dispersers) and computational objects (e.g., HSGs) articulated by Trevisan [Tre02]. Our construction can also be appreciated for a number of more technical reasons, which we outline now.

The two main objects we work with are hitting set generators (HSGs) — the one-sided variant of PRGs, and *dispersers* — the one-sided variant of randomness extractors.

Informally, a HSG construction takes an n-bit truth table of a hard function f and converts it into a collection of poly(n) shorter m-bit strings, with the property that every small circuit D that accepts at least 1/2 of its inputs, also accepts one of these m-bit strings. The proof that a construction is indeed a HSG typically gives an efficient way to convert a small circuit D on which the construction fails to meet the definition into a small circuit computing f, thus contradicting the hardness of f.

Informally, a disperser takes an n-bit string x sampled from a weak random source with sufficient min-entropy and converts it into a collection of poly(n) shorter m-bit strings, with the property that every circuit D that accepts at least 1/2 of its inputs, also accepts one of these m-bit strings. Trevisan's insight [Tre02] is that a HSG construction whose proof uses D in a black-box fashion *is* a disperser, for the following reason: if there is a circuit D on which the construction fails to meet the disperser definition, then we have a small circuit *relative to* D that describes input x, and it cannot be that every string in a source with sufficiently high min-entropy has such a short description.

Thus we can produce a formal statement to the effect that "every black-box HSG construction yields a disperser with similar parameters." In this paper we consider the reverse question, namely: "under what conditions does a disperser construction yield a HSG construction?"

We will limit ourselves to so-called "reconstructive" dispersers which means, roughly, that the associated proof has the same outline as the one sketched above, and that the conversion in the proof is efficient. At first glance this may seem to be such a strong constraint that the question becomes uninteresting. However, there is an important issue related to the precise meaning of "efficient." It turns out that there are (at least) two possible notions of "efficient;" one is satisfied naturally in several disperser constructions, and the other – which is the one that is actually required for the construction to be a HSG – is far more stringent. The distinction between the two is analogous to the distinction between an error-correcting code being efficiently decodable in the usual sense, and being efficiently *implicitly* (or "locally") decodable in the sense of [STV01].

To be precise, consider a function $E : \{0,1\}^n \times \{0,1\}^t \rightarrow \{0,1\}^m$ (where $m < n$ may be as small as poly $\log n$) and a circuit $D : \{0,1\}^m \rightarrow \{0,1\}$ relative to which E fails to be a disperser; i.e. D accepts at least half of its inputs, but for every x in the weak random source, $E(x, \cdot)$ fails to hit an accepting input of D. If E is equipped with a "reconstructive" proof then the proof should give an efficient way to reconstruct x from D and short advice (the advice may depend on x and auxiliary randomness used in the proof). The two notions of reconstructivity for E that we discuss are:

Explicit Reconstructivity. Given oracle access to circuit D, and advice $A(x, w)$, compute in time poly(n) the string x with non-negligible probability over the choice of w.

Implicit Reconstructivity. Given oracle access to circuit D, and advice $A(x, w)$, and an index i, compute in time poly$(m, \log n)$ the i-th bit of x with non-negligible probability over the choice of w.

Explicit reconstructivity is naturally satisfied by relatively simple disperser constructions[1] [Tre02, TSZS01, SU05], and these construction possess two additional useful features as well (as observed in [ISW99, ISW00, TSUZ01]): (1) w has length $O(\log n)$ and (2) A is computable in time poly(n).

In contrast, obtaining *implicit reconstructivity* has always required significant extra effort: in [SU05], one needs to use multiple copies of the original disperser with multiple "strides" and to piece them together in a complex manner; in [Uma03], similar ideas are used, together with an "augmented" low-degree extension; and in [ISW99, ISW00], the construction of [Tre02] is modified by repeated composition, but at the price of a super-polynomial degradation in the output length.

In this paper, we show that the weaker notion of explicit reconstructivity (together with the two additional properties satisfied by known constructions) is in fact sufficient to construct *optimal* HSGs, thus avoiding the complications of past work.

Our result is shown by analyzing a composition of reconstructive dispersers, similar to the one used in [ISW99, ISW00]. Our composition has the advantage of being simpler (we believe), and it is presented in a modular and generic manner, making it easy to see that it works for any reconstructive disperser. Most importantly, it produces optimal HSGs for all hardnesses, starting only with reconstructive dispersers for a particular "easy" setting of parameters.

Our results also shed light on the following issues:

[1]Of course these are all actually extractor constructions, but an extractor is a disperser.

- The Nisan-Wigderson pseudorandom generator (PRG) [NW94] does not achieve an optimal quantitative hardness vs. randomness tradeoff. Two works [ISW99, ISW00] addressed this deficiency by composing the PRG with itself in a clever way. The results came close to an optimal construction, but fell short. Subsequently optimal tradeoffs were achieved by employing different, algebraic constructions [SU05, Uma03]. Thus, one might have guessed that there was some inherent loss associated with the composition-based approach, or that combinatorial-design-based constructions such as [NW94] were incapable of achieving the optimal result. Our results show that in the end, composing the combinatorial-design-based Nisan-Wigderson PRG with itself *can* be made to work to obtain HSGs, by using a somewhat different composition than those used previously. We note, however, that while we obtain optimal HSGs (improving [ISW99] and matching [SU05]), we do not know how to achieve optimal PRGs (which would improve [ISW00] and match [Uma03]) using the present techniques.
- Until this paper, known constructions of optimal HSGs [SU05, Uma03] have made crucial use of implicit list-decoding (cf. [STV01]) of Reed-Muller codes. Implicit (or "local") list-decoding of Reed-Muller codes also underlies the hardness amplification results that were a component of earlier (non-optimal) constructions [ISW99, ISW00, STV01]. One may wonder whether some form of implicit list-decoding is in fact *necessary* to construct HSGs. Our results show that it is not, and indeed we can construct optimal HSGs using only combinatorial designs and any good list-decodable code (which, not accidentally, are also the two ingredients needed for Trevisan's extractors).

We remark that "reconstructivity" of various disperser and extractor constructions has emerged as a crucial property of these constructions in a number of applications: error-correcting codes [TSZ04, Gur04], data structures [TS02], and complexity theory [BLvM05]. This suggests that it is worthwhile to formalize and study notions of reconstructivity as we do in this paper.

Outline. In Section 2 we define HSGs and reconstructive dispersers. In Section 3 we give informal intuition for how our composition works, and in Section 4 we state and prove the formal result. Section 5 applies the composition theorem to obtain optimal HSGs. Section 6 outlines three known constructions of reconstructive dispersers, in our terminology. Finally in Section 7 we discuss some open problems raised by this work.

2 Preliminaries

We use $[n]$ as shorthand for the set $\{1, 2, 3 \ldots n\}$, and $x[i]$ to denote the i-th symbol in string x. We will also need to refer to "b-bit blocks" of a string x, by which we mean that that the bits of x should be partitioned into contiguous segments of b bits (the last segment may have fewer than b bits and can be padded arbitrarily). The block length b will always be clearly stated (or clear from context), and in such settings we will denote the i-th block of x by x_i.

2.1 Error-correcting codes

Our use of error-correcting codes is almost entirely in Section 6, so the reader may wish to skip this subsection on a first reading.

Definition 2.1. *An* error-correcting code *is a function* $C : \mathbb{F}_q^k \rightarrow \mathbb{F}_q^n$. *The code is* binary *if* $q = 2$. *The* minimum distance *of a code* C *is* $\min_{x \neq y} \Delta(C(x), C(y))$, *where* Δ *is the Hamming distance function. The* relative distance *is the minimum distance divided by* n. *A code* C *is* efficiently encodable *if* C *is computable* *in poly$(n, \log q)$ time. A code C is* efficiently list-decodable to radius r *if, given a received word $w \in \mathbb{F}_q^n$, one can produce the set $S_w = \{x : \Delta(w, C(x)) \leq r\}$ in poly$(|S_w|, n, \log q)$ time.*

The Johnson Bound gives an upper bound on $|S_w|$ in terms of the minimum distance of the code: if a binary code C has relative distance at least $1/2 - \delta^2$, then the number of codewords in any ball of relative radius $1/2 - \delta$ is at most $O(1/\delta^2)$.

We will mainly use codes based on polynomials. The Reed-Muller code with total degree h and dimension ℓ has as codewords the multivariate polynomials $p : \mathbb{F}_q^\ell \rightarrow \mathbb{F}_q$ of total degree h, evaluated at every point in their domain (and so the codewords have length q^ℓ). We index the symbols of Reed-Muller codewords naturally by elements of $\mathbb{F}_q^\ell$. By the Schwartz-Zippel Lemma, the relative distance of such codes is at least $1 - h/q$. The special case of $h = 1$ yields the Hadamard code, and the special case of $\ell = 1$ yields Reed-Solomon codes. We will use the following result regarding list-decoding of Reed-Solomon codes:

Lemma 2.2 ([Sud97]). *Given a received word $w \in \mathbb{F}_q^q$, one can produce in poly(q) time the set of degree k polynomials $p : \mathbb{F}_q \rightarrow \mathbb{F}_q$ having agreement t with w, provided $t \geq \sqrt{2kq}$. Moreover the number of such polynomials is at most $2q/t$.*

2.2 Hitting set generators and ordinary dispersers

We use U_m for the random variable uniformly distributed on $\{0,1\}^m$. If X is a set, we sometimes also use X to denote the random variable that is uniform over X.

Definition 2.3. *Let Z be a random variable distributed on $\{0,1\}^m$. We say that a function $D : \{0,1\}^m \rightarrow \{0,1\}$ "ϵ-catches" Z if*

$$|\Pr[D(U_m) = 1] - \Pr[D(Z) = 1]| > \epsilon.$$

In the special case that $\Pr[D(Z) = 1] = 0$, we say that D "ϵ-avoids Z".

In this paper we will always be in the aforementioned special case, since we are discussing one-sided objects (HSGs and dispersers). Replacing "ϵ-avoids" with "ϵ-catches" everywhere in the paper yields the two-sided version of all of the definitions, theorems and proofs, with the exception of one place in the proof of Theorem 4.1 where we use the one-sidedness critically.

Definition 2.4. *An* ϵ-HSG *is a function*

$$H : \{0,1\}^t \rightarrow \{0,1\}^m$$

such that no size m circuit $D : \{0,1\}^m \rightarrow \{0,1\}$ ϵ-avoids $H(U_t)$.

This means that every size m circuit D that accepts more than an ϵ fraction of its inputs, also accepts $H(y)$ for some y. We will construct HSGs from hard functions: given an n-bit string x that is the truth table of a function requiring circuits of size k, we will use it to construct a HSG H_x. In discussing such a construction, we take n as the "main" size parameter, and measure the other parameters as functions of n. It has become customary to refer to families of ϵ-HSGs that for every $k = k(n)$ have parameters $t \leq O(\log n)$ and $m \geq k^\delta$ for some constant $\delta > 0$ as *optimal*, because they give rise to hardness vs. randomness tradeoffs that are optimal up to a polynomial. See the discussion of this point in [ISW03].

For reference, we give the standard definition of a (k, ϵ)-disperser before stating the more complicated definition of a reconstructive disperser that we will need for this paper.

Definition 2.5. *A* (k, ϵ)-disperser *is a function*

$$E : \{0,1\}^n \times \{0,1\}^t \rightarrow \{0,1\}^m$$

such that for all subsets $X \subseteq \{0,1\}^n$ with $|X| > 2^k$, no circuit $D : \{0,1\}^m \rightarrow \{0,1\}$ ϵ-avoids $E(X, U_t)$.

parameter	interpretation
n	length of source string
t	disperser seed length
m	disperser output length
d	advice function seed length
a	advice length
b	block length
ϵ	disperser error
δ	reconstruction procedure success probability

Table 1: The eight parameters of a reconstructive disperser.

2.3 Reconstructive dispersers

We now define the central object, which we call a reconstructive disperser. It has more parameters than one would prefer, but keeping track of all of these parameters will make the composition much easier to state. For reference, in Table 1, we list the parameters of a reconstructive disperser together with their meanings.

Definition 2.6. *A* $(n, t, m, d, a, b, \epsilon, \delta)$-reconstructive disperser *is a triple of functions:*

- *the "disperser" function* $E : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$
- *the "advice" function* $A : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^a$
- *the randomized*[2] *oracle "reconstruction" procedure* $R : \{0,1\}^a \times [n/b] \to \{0,1\}^b$

that satisfy the following property: for every $D : \{0,1\}^m \to \{0,1\}$ *and* $x \in \{0,1\}^n$ *for which* D ϵ*-avoids* $E(x, U_t)$*, we have*

$$\forall i \in [n/b] \quad \Pr_w[R^D(A(x,w), i) = x_i] \geq \delta. \tag{1}$$

Here x_i *refers to the* i*-th* b*-bit block in* x*. When* $b = n$ *we drop the second argument to* R*.*

Note that in this definition there is no reference to the parameter "k" that occurs in Definition 2.5. The idea is that if (E, A, R) is a reconstructive disperser, then E must be a (k, ϵ)-disperser for k slightly larger than a, because relative to D, many strings x in the source X have descriptions of size approximately a (via R). Similarly, $E(x, \cdot)$ is an ϵ-HSG when x is the truth table of a function that does not have size k circuits, for k slightly larger than a plus the running time of R. This is because relative to D, and given a bits of advice, R can be used to compute any specified bit of x. See Theorems 2.8 and 2.9 for formal statements of these assertions. We remark that both proofs work by amplifying the success probability of the reconstruction procedure R (by repetition), which produces a new procedure that with positive probability succeeds on *all* b-bit blocks of x simultaneously, in contrast to the definition above which demands only that for each block, the procedure succeeds with probability δ. The former requirement (that the reconstruction procedure succeed on all blocks simultaneously) may seem more natural, but the weaker requirement in Definition 2.6 will be more convenient to work with, and can be converted with relative ease to the stronger one, as in the proofs of Theorems 2.8 and 2.9 (or in a randomness-efficient manner via Lemma 2.10).

[2]When we refer to a "randomized" function, we mean a function f that takes an extra argument which is thought of as random bits. We refrain from explicitly writing the second argument; however whenever f occurs within a probability, we understand that the probability space includes the randomness of f.

The parameter b interpolates between the two types of reconstructivity. When $b = n$, the reconstruction procedure outputs all of x. A reconstruction procedure of this type running in time poly(n) is *explicit* and is often implied by disperser constructions whose proofs use the so-called "reconstruction proof paradigm."

To obtain an optimal HSG, we need b to be small and the running time of the reconstruction procedure to be poly$(m, \log n)$ (where $m < n$ may be as small as poly $\log n$). Such a reconstruction procedure is necessarily *implicit*, and it satisfies a much more stringent efficiency requirement that typically does *not* follow from disperser constructions without modification. Note however that it is trivial to decrease b by a multiplicative factor without changing the running time of R (and we will use this fact). The challenge is to decrease b while decreasing the running time of R simultaneously.

There are three quite clean constructions [Tre02, TSZS01, SU05] of reconstructive dispersers known for a certain "easy" setting of the parameters. By "easy," we mean here that the advice length is at most n^γ for some constant $\gamma < 1$, while the disperser output length is n^β for some constant $\beta > 0$. The original goal in these three works was to construct extractors (as compared to our target object, a HSG), and then it becomes important to minimize $\gamma - \beta$. It is the pursuit of this objective that made the original constructions and their parameter choices somewhat delicate; without this constraint, the constructions are fairly straightforward — see Section 6.

Theorem 2.7 ([Tre02, TSZS01, SU05]). *There exist constants $1 > \gamma > \beta > 0$ such that for every n and $\epsilon \geq n^{-\beta}$ there is a*

$$(n, t = O(\log n), m = n^\beta, d = O(\log n), a = n^\gamma, b = n, \epsilon, \delta = 2/3)$$

reconstructive disperser (E, A, R) with the running time of E, A, R at most poly(n).

For the proof, see Section 6, which outlines the three constructions in detail.

Note that these constructions are *not* sufficient to produce HSGs directly, because the running time of the reconstruction procedure R is by itself far greater than the trivial upper bound on the circuit complexity of a function whose truth table has size n, so we cannot get the required contradiction.

However, when the running time of R is much smaller than the input length n, reconstructive dispersers *are* hitting set generators when their input is fixed to be a hard function.

Theorem 2.8. *Let (E, A, R) be a*

$$(n, t, m, d, a, b, \epsilon, \delta = 2/3)$$

reconstructive disperser, for which R runs in time T. Let $x \in \{0,1\}^n$ be the truth table of a function that cannot be computed by circuits of size k. There is a universal constant c for which: if $k > c(\log n)(Tm + a)$, then $H_x(\cdot) = E(x, \cdot)$ is an ϵ-HSG.

Proof. If $E(x, \cdot)$ is not the claimed HSG, then there is a size m circuit $D : \{0,1\}^m \to \{0,1\}$ that ϵ-avoids $E(x, U_t)$. By the definition of reconstructive dispersers we have:

$$\forall i \in [n/b] \quad \Pr_w[R^D(A(x,w), i) = x_i] \geq 2/3.$$

We repeat the reconstruction $\Theta(\log n)$ times with independent random w's and take the majority outcome. For a given i, the probability that this fails to produce x_i is less than $1/n$ by Chernoff bounds. By a union bound, the probability that we fail on any i is strictly less than one. Thus we can fix the random bits used in this procedure so that we correctly produce x_i for all i. We hardwire $A(x, w)$ for the chosen w's. The resulting circuit has size $c(\log n)(Tm + a)$ which contradicts the hardness of x. □

Essentially the same argument shows that reconstructive dispersers are indeed a special case of ordinary dispersers (a more efficient conversion is possible if an error-correcting code is applied to the source, but that is not important for this paper):

Theorem 2.9. *Let (E, A, R) be a*

$$(n, t, m, d, a, b, \epsilon, \delta = 2/3)$$

reconstructive disperser. Then there is a universal constant c for which: E is a $(k = ac \log n, \epsilon)$-disperser.

Proof. Fix a function $D : \{0,1\}^m \to \{0,1\}$. Let x be a string for which D ϵ-avoids $E(x, U_t)$. Following the proof of Theorem 2.8, we consider the randomized procedure that runs the reconstruction $\Theta(\log n)$ times with independent random seeds and takes the majority outcome. As argued in the proof of Theorem 2.8, there is some choice of random w's (and associated $A(x, w)$'s) for which the procedure correctly produces x_i for all i. These $A(x, w)$'s comprise a description of x, of length $ac \log n$ for some universal constant c.

We conclude that at most $K = 2^{ac \log n}$ strings x have the property that D ϵ-avoids $E(x, U_t)$. So, if $X \subseteq \{0,1\}^n$ satisfies $|X| > K$, then D does not ϵ-avoid $E(X, U_t)$, and this holds for all D, so E is the claimed disperser.

□

2.4 Amplifying the success probability of the reconstruction procedure

The proofs of Theorems 2.8 and 2.9 work by amplifying the success probability of the reconstruction procedure by repetition. In the rest of the paper we will need a similar amplification, but one that is more randomness-efficient. The next lemma achieves this using pairwise independence:

Lemma 2.10 (increasing δ). *Suppose (E, A, R) is a*

$$(n, t, m, d, a, b = n, \epsilon, \delta)$$

reconstructive extractor. For every $\alpha > 0$, we can convert (E, A, R) into a

$$(n, t, m, d' = O(d + \log(pn)), a' = O(pa), b = n, \epsilon, \delta' = 1 - \alpha)$$

reconstructive extractor (E, A', R'), where $p = \delta^{-1}\alpha^{-1}$. The running time of A' is at most poly(p, n) times the running time of A, and the running time of R' is at most poly(p, n) times the running time of R.

Proof. Set $r = 2p$, and let $Y_1, Y_2, \ldots, Y_r$ be pairwise independent random variables over $\{0,1\}^d$. We will use an explicit construction of such a space that can be sampled using $u = O(d + \log r)$ random bits[3], and we denote by $Y_i(w)$ the value of random variable Y_i when sampling from the space with w as the random bits. Let $C : \{0,1\}^n \to \mathbb{F}_q^q$ be a Reed-Solomon code (which we will use as a hash function family) with $q \geq 2rn/\alpha$.

We define $A' : \{0,1\}^n \times \{0,1\}^{d'} \to \{0,1\}^{a'}$ as follows:

$$A'(x; w \in \{0,1\}^u, j \in \mathbb{F}_q) = (A(x, Y_1(w)), A(x, Y_2(w)), \cdots, A(x, Y_r(w)), C(x)_j, j).$$

In other words, A' simply concatenates the output of r different runs of A using pairwise independent seeds, and appends a hash of x.

Our new randomized oracle procedure $R' : \{0,1\}^{a'} \to \{0,1\}^n$ is defined as follows:

[3]One construction with these parameters is as follows. Let $\mathbb{F}$ be the field with 2^d elements, and choose $\ell = 1 + \lceil (\log(r+1))/d \rceil$. Pick $a \in \mathbb{F}^\ell$ at random, and define $Y_i = \sum_j a_j b_j^{(i)}$, where the $b^{(i)}$ are distinct vectors in $\mathbb{F}^\ell$ whose first coordinate is 1.

- On input $(z_1, z_2, \ldots, z_r, y, j)$, run $R^D(z_1), R^D(z_2), \ldots, R^D(z_r)$
- Select any ℓ for which $C(R^D(z_\ell))_j = y$, and output $R^D(z_\ell)$. If there is no such ℓ, output anything.

In other words, R' runs the original reconstruction procedure R on each of the r concatenated advice strings, and uses the hash of x to select which of the reconstructions to output.

We need to prove that this works. Fix $D : \{0,1\}^m \to \{0,1\}$ and $x \in \{0,1\}^n$ for which D ϵ-avoids $E(x, U_t)$. We want to show:

$$\Pr_{w \in \{0,1\}^u, j \in \mathbb{F}_q}[{R'}^D(A'(x; w, j)) = x] \geq 1 - \alpha.$$

We first argue that the probability over $w \in \{0,1\}^u$ that x fails to appear among

$$R^D(A(x, Y_1(w))), R^D(A(x, Y_2(w))), \ldots, R^D(A(x, Y_r(w)))$$

is small. The expected number of occurrences of x in this list is $2/\alpha$. By Chebyshev, the probability that the number of occurrences is 0 is at most $\alpha/2$.

Next, we argue that if x is in the list, then the probability of outputting a string that is not x is small. For every $x' \neq x$ in the list, $\Pr_{j \in \mathbb{F}_q}[C(x')_j = C(x)_j] \leq n/q$, since the Reed-Solomon code C has relative agreement at most n/q. By a union bound,

$$\Pr_{j \in \mathbb{F}_q}[\exists x' \neq x \;\; C(x')_j = C(x)_j] \leq rn/q \leq \alpha/2.$$

We conclude that $\Pr_{w \in \{0,1\}^u, j \in \mathbb{F}_q}[{R'}^D(A'(x; w, j)) = x] \geq 1 - \alpha$, as required. $\square$

3 Intuition for the composition

From Theorem 2.8 we can see that a reconstructive disperser *is* a HSG, which would be *optimal* if the advice length a and the running time of the reconstruction procedure R can be made to be only polynomially large in the output length m. But starting with the constructions of Theorem 2.7, we only know how to achieve that for large $m = n^{\Omega(1)}$. Rather than try to preserve these parameters while making m smaller and smaller (which is a natural approach), our goal in the composition will be to maintain the absolute parameters of advice length $a = \text{poly}(m) = \text{poly}(n)$ and reconstruction procedure running time $T = \text{poly}(m) = \text{poly}(n)$, while making the *input length* larger and larger — and to emphasize that the input length of the composed object is large we will rename it N. Specifically, our goal will be to obtain (after several compositions) a reconstructive disperser with input length $N \gg n$, advice length poly(n), and R's running time poly(n), while maintaining an output length of at least $n^{\Omega(1)}$. Note that to achieve this running time for R, we *must* achieve implicit reconstructivity in the final object.

We now outline the composition at a high level. In this discussion we focus almost entirely on the advice and reconstruction functions. Our starting point is the "simple" constructions of Theorem 2.7. Note that $b = n$ in those constructions so we drop the second argument of the reconstruction procedure R, and then it simply maps $a = n^\gamma$ bits to n bits.

We make two unreasonable simplifying assumptions for the purpose of this exposition, and later describe how to remove them. First, we assume that that $d = 0$, i.e., the advice function A just maps n bits to $a = n^\gamma$ bits. Second, we assume that the reconstruction procedure R can function properly *without* access to the oracle D, and thus it just maps a bits to n bits. It is then useful to think of the advice function A as a procedure that "compresses" an arbitrary n-bit string x into n^γ bits and the reconstruction procedure R as a procedure that "decompresses" the n^γ-bit string back to the original n-bit string x.

So we have the ability to compress from n bits down to n^γ bits. Suppose we want to be able to compress from $N \gg n$ down to n^γ bits. A natural thing to do is to divide the N-bit string into N/n substrings of size n, and use A to compress each of the substrings. The resulting string has length $(N/n)n^\gamma$; we can repeat the process $O(\log N/\log n)$ times until we get down to length n^γ. To decompress, we use R to reverse the process. A nice side-effect of this scheme is that when we know *which* n-bit substring of the original N bit string we wish to recover, we only need to invoke R *once* at each level (as opposed to, e.g., N/n times at the bottom level, if we insist on recovering the entire original N bit string). In the formalism of Definition 2.6, this means that we can take $b = n$, and have R only recover the specified b-bit block of the input string.

The above description specifies an advice function that maps an N-bit string x down to n^γ bits, and a reconstruction procedure that recovers any specified n-bit substring of x from those n^γ bits. A crucial observation is that the new reconstruction procedure has $b = n \ll N$, and it runs in time poly$(n, \log N/\log n) \ll N$, since it invokes R once for each of the $O(\log N/\log n)$ levels. Thus we have achieved our goals related to the advice function and reconstruction procedure : we have advice length poly(n), and a reconstruction procedure running in time poly(n). We have yet to describe the disperser function of our composed object, which we are aiming to have output $n^{\Omega(1)}$-bit strings. We now describe the disperser function, in the process of showing how to remove the two simplifying assumptions.

Our first simplifying assumption was that the advice function seed length $d = 0$, when in fact it must be $O(\log n)$. When $d = O(\log n)$ the "compression" function A produces $2^d =$ poly(n) *candidate* compressed versions of x, with the property that with high probability over a choice of candidates, the "decompression" procedure R succeeds. In the actual construction, we will pick an independent seed of length $d = O(\log n)$ for each level which determines which candidate we work with at that level (for a total seed length of $d' = O(\log N)$). The overall compression function thus produces $2^{d'} =$ poly(N) candidates. By making the probability that R fails at each individual level small enough (i.e., less than $1/(\#$ of levels$)$), we ensure that the overall decompression (which invokes the reconstruction procedure R once at each level) works with constant probability when run with a random candidate at each level, which is what is required.

Our second simplifying assumption was to ignore the fact that R needs oracle access to a function D with certain properties to succeed. To deal with this, we simply run the disperser E on every n-bit string that we "compress" (and hence may need the capability to decompress) at any level in the entire composition. We define the overall disperser E' so that its outputs are the union of the outputs of each invocation of E. Recalling Definition 2.6, we see that the reconstruction procedure R' for the composed object is only required to work relative to D that ϵ-avoids the output of E'. Such a D also ϵ-avoids the output of each invocation of E we have performed[4]. Our overall reconstruction procedure R' has oracle access to D, and therefore, each invocation of the "decompression" function R has oracle access to D (which ϵ-avoids the output of the associated E) — and this is exactly what is required to enable the individual "decompressions" to succeed at all levels.

Both the disperser function E' and the advice function A' of the composed object have asymptotically optimal seed lengths of $O(\log N)$. This is because at each level we need a fresh $O(\log n)$ bit seed for E and A coming from Theorem 2.7, and there are $O(\log N/\log n)$ levels.

The formal analysis of a single level of this composition is the content of Theorem 4.1. Corollary 4.2 and Theorem 5.1 apply this composition $O(\log N/\log n)$ times to the reconstructive dispersers of Theorem 2.7 to obtain the final result.

[4]This is the single place where we rely critically on the fact that we are dealing with "ϵ-avoids," rather than "ϵ-catches."

Comparison with [ISW99, ISW00]. As noted, our composition is similar to the one used in [ISW99, ISW00], which can also be understood in the language of reconstructive dispersers. Adopting the intuition of "compression" and "decompression" from above, one can view [ISW99, ISW00] as defining the advice function of the composed object to be the original advice function run repeatedly: i.e., in the first application of the advice function, the original string x is compressed from n bits down to n^γ bits, in the second application of the advice function to this new *shorter* string, it is compressed further to $(n^\gamma)^\gamma$ bits, and so on. (In contrast, in this paper, we only ever apply the advice function to strings of length n). The reconstruction procedure can be used to "decompress" by applying it repeatedly to the final advice string, but in order for the overall object to achieve implicit reconstructivity, one must demand that the original reconstructive disperser possess implicit reconstructivity from the start. In addition, when things are organized this way, each invocation of the reconstruction procedure associated with a lower level in the composition triggers recursive invocations of the reconstruction procedures for higher levels. The result is a superpolynomial degradation of the running time of the overall reconstruction procedure if the number of levels is super-constant.

So one might say that the overall strategy in [ISW99, ISW00] was to take an initial construction possessing implicit reconstructivity [NW94, STV01], whose only failing was that the advice length was much larger than poly(m), and repeatedly compress until the advice length became poly(m). In contrast, our strategy in this paper is to take an initial construction with the proper relationship between output length, advice length, and reconstruction procedure running time, but which only works for large $m = n^{\Omega(1)}$ (Theorem 2.7) and compose it to obtain a new construction with a much larger *input length* N. This method only requires explicit reconstructivity to start (implicit reconstructivity is a natural side-effect of the composition), and avoids the superpolynomial loss associated with the recursive invocation of the reconstruction procedure in [ISW99, ISW00]. It should be pointed out, however, that [ISW00] produces a two-sided object (a PRG) via composition, while our composition seems to only work to produce a one-sided object (a HSG).

4 Analysis of the composition

Our main composition operation for reconstructive dispersers increases n. More specifically, it roughly squares n, while multiplying each of the two seed lengths (t and d) by a constant. The final object inherits the advice length a from the first reconstructive disperser, and the block length b from the second.

Theorem 4.1 (composition of reconstructive dispersers). *Suppose* (E_1, A_1, R_1) *is a*

$$(n_1, t_1, m, d_1, a_1, b_1, \epsilon, \delta_1)$$

reconstructive disperser and (E_2, A_2, R_2) *is a*

$$(n_2, t_2, m, d_2, a_2, b_2, \epsilon, \delta_2)$$

reconstructive disperser, with $a_2 = b_1$. *Set* $r = n_1/b_1$. *Then* (E, A, R) *defined as:*

$$\begin{aligned} E(x_1, \ldots, x_r; w_2, y, p \in [r+1]) &= \begin{cases} E_1(A_2(x_1, w_2) \circ \cdots \circ A_2(x_r, w_2), y) & \text{if } p = r+1 \\ E_2(x_p, y) & \text{otherwise} \end{cases} \\ A(x_1, \ldots, x_r; w_1, w_2) &= A_1(A_2(x_1, w_2) \circ \cdots \circ A_2(x_r, w_2), w_1) \\ R^D(z; i_1 \in [r], i_2 \in [n_2/b_2]) &= R_2^D(R_1^D(z, i_1), i_2) \end{aligned}$$

is a

$$\left(n = \frac{n_1 n_2}{a_2}, t = \max(t_1, t_2) + d_2 + \log(r+1), m, d = d_1 + d_2, a = a_1, b = b_2, \epsilon, \delta = \delta_1 + \delta_2 - 1\right)$$

reconstructive disperser.

A few words of explanation are in order. The input for the composed object is $x = x_1, x_2, \ldots, x_r$, where the x_i coincide with the blocks R will need to output. The function A is very simple: it just concatenates the output of A_2 run on each of $x_1, x_2, \ldots, x_r$, and runs A_1 on the concatenated string. The function R is similarly simple; it reverses the process: it takes the output of A and first uses R_1 to extract the advice associated with x_{i_1}, and then uses R_2 to actually recover (the i_2-th block of) x_{i_1}. Finally E uses part of its seed, p, to decide either to run E_2 on input x_i for some i, or to run E_1 on the concatenated advice string on which A_1 is run.

Proof. (of Theorem 4.1) Fix D and $x = x_1, x_2, \ldots, x_r$ for which D ϵ-avoids $E(x_1, \ldots, x_r; U_t)$. Also fix $i = i_1 i_2$, where $i_1 \in [r]$ and $i_2 \in [n_2/b_2]$.

From the fact that D ϵ-avoids $E(x_1, \ldots, x_r; U_t)$, we know that

$$\begin{array}{rl} \forall w_2 & D \ \ \epsilon\text{-avoids } E_1(A_2(x_1, w_2) \circ \cdots \circ A_2(x_r, w_2), U_{t_1}) \qquad (2) \\ \forall p \in [r] & D \ \ \epsilon\text{-avoids } E_2(x_p, U_{t_2}) \qquad (3) \end{array}$$

From (2) and the definition of reconstructive dispersers, we get that for all w_2:

$$\Pr_{w_1}[R_1^D(A_1(A_2(x_1, w_2) \circ \cdots \circ A_2(x_r, w_2), w_1), i_1) = A_2(x_{i_1}, w_2)] \geq \delta_1. \qquad (4)$$

From (3) and the definition of reconstructive dispersers, we get that:

$$\Pr_{w_2}[R_2^D(A_2(x_{i_1}, w_2), i_2) = (x_{i_1})_{i_2}] \geq \delta_2. \qquad (5)$$

The probability over a random choice of w_1 and w_2 that both events occur is at least $\delta_1 + \delta_2 - 1$. If both events occur, then:

$$\begin{aligned} R^D(A(x_1, \ldots, x_r; w_1, w_2); i_1, i_2) &= R_2^D(R_1^D(A(x_1, \ldots, x_r; w_1, w_2), i_1), i_2) && (6) \\ &= R_2^D(R_1^D(A_1(A_2(x_1, w_2) \circ \cdots \circ A_2(x_r, w_2), w_1), i_1), i_2) && (7) \\ &= R_2^D(A_2(x_{i_1}, w_2), i_2) = (x_{i_1})_{i_2}. && (8) \end{aligned}$$

where (6) just applies the definition of R, (7) applies the definition of A, and (8) follows from our assumption that the events in (4) and (5) both occur.

We conclude that

$$\Pr_{w_1, w_2}[R^D(A(x; w_1, w_2), i) = x_i] \geq \delta_1 + \delta_2 - 1,$$

which is what was to be shown. □

We now apply Theorem 4.1 repeatedly. We start with a reconstructive disperser (E, A, R) and compose it with itself. In subsequent repetitions, we apply Theorem 4.1, taking the reconstructive disperser from the previous repetition as (E_1, A_1, R_1), and the original (E, A, R) as the second reconstructive disperser (E_2, A_2, R_2).

Corollary 4.2. *Fix N, n, ϵ, and constant $\gamma < 1$. Let (E, A, R) be a*

$$(n, t_1 = O(\log n), m, d_1 = O(\log n), a = n^\gamma, b = a, \epsilon, \delta_1 = 1 - (1/\log N))$$

reconstructive disperser with the running time of E, A, R at most n^{c_1} for a universal constant c_1. Then (E', A', R') obtained by $\left(\frac{\log N}{(1-\gamma)\log n}\right)$ applications of Theorem 4.1 is a

$$(N, t = O(\log N), m, d = O(\log N), a = n^\gamma, b = a, \epsilon, \delta = 1 - o(1))$$

reconstructive disperser with the running time of E' and A' at most N^c, and the running time of R' at most n^c for a universal constant c.

Proof. We claim that after i compositions of (E, A, R) with itself, we obtain a

$$\left(n_i = n^{i-(i-1)\gamma}, t_i = t_1 + (i-1)(d_1 + \log(n^{1-\gamma}+1)), m, d_i = id_1, a, b, \epsilon, \delta_i = i(\delta_1 - 1) + 1\right)$$

reconstructive disperser (E_i, A_i, R_i). This clearly holds for $i = 1$.

To see that it holds for arbitrary i, consider composing $(E_{i-1}, A_{i-1}, R_{i-1})$ with (E, A, R). By Theorem 4.1 and our inductive assumption, we get (E_i, A_i, R_i) with parameters:

$$\begin{aligned}
n_i &= n_{i-1}n/a = n^{i-1-(i-2)\gamma}n/n^\gamma = n^{i-(i-1)\gamma} \\
t_i &= \max(t_{i-1}, t_1) + d_1 + \log(n^{1-\gamma}+1) \\
&= [t_1 + (i-1)(d_1 + \log(n^{1-\gamma}+1))] + d_1 + \log(n^{1-\gamma}+1) \\
&= t_1 + i(d_1 + \log(n^{1-\gamma}+1)) \\
d_i &= d_{i-1} + d_1 = id_1 \\
\delta_i &= \delta_{i-1} + \delta_1 - 1 = (i-1)(\delta_1 - 1) + \delta_1 = i(\delta_1 - 1) + 1
\end{aligned}$$

as claimed. Note that $n_i > n^{i(1-\gamma)}$. Thus when

$$i = \frac{\log N}{(1-\gamma)\log n},$$

we have $n_i \geq N$, and also $d_i = O(\log N)$ and $t_i = O(\log N)$. Also note that for sufficiently large n

$$\delta_i = 1 - i(1/\log N) = 1 - \frac{1}{(1-\gamma)\log n} = 1 - o(1).$$

Finally, let $T(E_i), T(A_i), T(R_i)$ denote the running times of the functions E_i, A_i, R_i respectively. From the specification of the composition in Theorem 4.1, we see that

$$\begin{aligned}
T(E_i) &\leq \max(T(E_{i-1}) + n_{i-1}T(A), T(E)) \\
T(A_i) &\leq T(A_{i-1}) + n_{i-1}T(A) \\
T(R_i) &\leq T(R_{i-1}) + T(R),
\end{aligned}$$

and then it is easy to verify by induction that $T(E_i), T(A_i) \leq in^{i+c_1}$ and $T(R_i) \leq in^{c_1}$. Plugging in $i = O(\log N/\log n)$ gives the claimed running times. □

5 Optimal hitting set generators

Now, we can use any one of the constructions of Theorem 2.7 in Corollary 4.2 to obtain an optimal HSG for arbitrary hardness. Specifically, our HSG is built from the N-bit truth table of a function that is hard for circuits of size k, has a seed length of $O(\log N)$, and outputs $k^{\Omega(1)}$ bits, while running in time poly(N). As usual, we assume $N < 2^{k^\eta}$ for any constant η; as otherwise we could just output the seed.

Theorem 5.1. *Let x be the N-bit truth table of a function that cannot be computed by circuits of size k, and set $n = k^{1/c}$ for a sufficiently large constant c. Let (E, A, R) be a*

$$(n, t = O(\log n), m = n^\beta, d = O(\log n), a = n^\gamma, b = n, \epsilon, \delta = 2/3)$$

reconstructive disperser from Theorem 2.7.

If (E', A', R') is the reconstructive disperser obtained by applying Lemma 2.10 with $\alpha = 1/\log N$ and then applying Corollary 4.2, then $H_x(\cdot) = E'(x, \cdot)$ is an ϵ-HSG against circuits of size $m = k^{\beta/c}$.

Proof. After applying Lemma 2.10 with $\alpha = 1/\log N$, we have a

$$\left(n, t = O(\log n), m = n^\beta, d' = O(\log n), a' = (\log N)n^\gamma, b = n, \epsilon, \delta' = 1 - \frac{1}{\log N}\right)$$

reconstructive disperser.

By the assumption on N stated before the theorem, we have that $a' \leq n^{\gamma'}$ for some other constant $\gamma' < 1$. As noted following Definition 2.6, we can decrease b from n to a' trivially. Then, after applying Corollary 4.2 we obtain a

$$(N, O(\log N), m, O(\log N), n^{\gamma'}, a', \epsilon, 2/3)$$

reconstructive disperser (E', A', R'), with the running time of E' and A' at most poly(N) and the running time of R' at most n^{c_1}.

To satisfy Theorem 2.8 we need $c_2 \log N(n^{c_1} n^\beta + n^{\gamma'}) < k$, where c_1 and c_2 are the universal constants from Corollary 4.2 and Theorem 2.8, respectively. We choose c large enough to ensure that this holds. Theorem 2.8 then states that $E'(x, \cdot)$ is an ϵ-HSG against circuits of size $m = n^\beta = k^{\beta/c}$. □

As noted in the introduction, if we use Trevisan's extractor [Tre02] as our starting object, then the entire construction requires only two ingredients: (1) any good list-decodable error-correcting code and (2) combinatorial designs (see Subsection 6.2). In particular there is *no* hardness amplification or implicit list-decoding hidden in the construction, precisely because we are able to work with a starting object that only has *explicit reconstructivity* rather than implicit reconstructivity (the latter type of reconstructivity typically has required implicit list-decoding in some form or another).

6 Specific constructions

As stated in Theorem 2.7, there are three relatively straightforward constructions of reconstructive dispersers that are sufficient for our purposes. Here we describe these constructions in our language. The construction in Subsection 6.2 is intended to be a complete proof of Theorem 2.7. The presentations of the following two constructions are detailed sketches; the reader should consult the original papers [TSZS01, SU05] to obtain full proofs.

6.1 Yao's Lemma

We will make repeated use of a variant of Yao's Lemma (the "moreover" part of the lemma is often not stated as it was not important in past applications, but it is crucial here):

Lemma 6.1 ([Yao82]). *Let Z be a random variable distributed on $\{0, 1\}^m$ and let $D : \{0, 1\}^m \to \{0, 1\}$ be a function that ϵ-avoids Z. Then exists an $i \in [m]$, and a function $P : \{0, 1\}^{i-1} \to \{0, 1\}$ for which*

$$\Pr[P(Z_{1,\ldots,i-1}) = Z_i] \geq \frac{1}{2} + \frac{\epsilon}{2m}.$$

Moreover, there is a uniform randomized procedure to produce P from D, which succeeds with probability at least $\epsilon/(8m^2)$.

Proof. It is by now standard (see, e.g., the proof in [Tre02]) that there exists $i \in [m]$, and bits $b_0, b_1 \in \{0,1\}$, for which

$$\Pr_{c_{i+1},\ldots,c_m}[b_0 \oplus D(Z_{1\ldots i-1}, b_1, c_{i+1}, \ldots, c_m) = Z_i] \geq \frac{1}{2} + \frac{\epsilon}{m}.$$

To obtain P from D, pick b_0, b_1, i, and $c_{i+1}, \ldots, c_m$ uniformly at random, and define $P(z_1, \ldots, z_{i-1}) = b_0 \oplus D(z_1, \ldots, z_{i-1}, b_1, c_{i+1}, \ldots, c_m)$. The probability b_0, b_1, i are correct is $1/(4m)$ and an averaging argument shows that with probability at least $\epsilon/(2m)$ the randomly chosen c's yield the required P. $\square$

6.2 The Trevisan reconstructive disperser

In this subsection, we describe Trevisan's extractor [Tre02] (based on [NW94]) in our language, showing that it is a reconstructive disperser satisfying Theorem 2.7. Fix $n, m = n^\beta, \epsilon \geq m^{-1}$, for a constant $\beta < 1$ to be specified later. There are two ingredients to the construction:

- A binary error-correcting code $C : \{0,1\}^{\bar{k}=n} \to \{0,1\}^{\bar{n}}$ with $\bar{n} = \text{poly}(\bar{k})$, and relative distance at least $1/2 - (\epsilon/(4m))^2$. We require that C has efficient encoding and efficient list-decoding from relative radius $1/2 - \epsilon/(4m)$. Such a code is easy to obtain, e.g., by concatenating a Reed-Solomon code with a binary Hadamard code, using brute-force list-decoding of the inner code and Lemma 2.2 to list-decode the outer code [GS00].
- A combinatorial design: subsets $S_1, S_2, \ldots, S_m$ of a universe $[t]$, for which $|S_i| = \log \bar{n}$ for all i, and $|S_i \cap S_j| \leq \beta \log n$. From [NW94] we have that such designs exist, for $t = O(\log \bar{n}/\beta^2)$, and that they can be constructed deterministically in time $O(2^t m) = \text{poly}(n)$.

The reconstructive disperser has three parts:

- The disperser function $E : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$. Given input x and seed $y \in \{0,1\}^t$, the i-th output bit is $C(x)[y_{|S_i}]$ where $y_{|S_i}$ denotes the restriction of $y \in \{0,1\}^t$ to the bit positions in the set S_i.
- The advice function $A : \{0,1\}^n \times \{0,1\}^{d=t} \to \{0,1\}^a$. We are given an input x and a seed $w \in \{0,1\}^d$. We output the following bits for every $i \in [m]$: for each $j < i$, restrict w to the bit positions indexed by S_j, alter the bit positions indexed by $S_j \cap S_i$ in all $2^{|S_j \cap S_i|}$ possible ways, and for each, use the resulting $(\log \bar{n})$-bit string to index into $C(x)$, outputting that bit of $C(x)$. For each i, this produces at most $m(i-1)$ output bits. Finally, we output the seed w for a total of at most $a = m^3$ output bits.
- The reconstruction function $R : \{0,1\}^a \to \{0,1\}^n$, which works as follows. We are given access to D that ϵ-avoids $Z = E(x, U_t)$. By Lemma 6.1 we can convert D into a next-bit predictor P with probability $\epsilon/(8m^2)$. That is,

  $$\Pr_w[P(E(x,w)_1, \ldots, E(x,w)_{i-1}) = E(x,w)_i] \geq \frac{1}{2} + \frac{\epsilon}{2m}.$$

  Substituting the definition of E:

  $$\Pr_w\left[P(C(x)[w_{|S_1}] \ldots C(x)[w_{|S_{i-1}}]) = C(x)[w_{|S_i}]\right] \geq \frac{1}{2} + \frac{\epsilon}{2m}.$$

Writing w as (w', w''), where w' is the portion of w outside the positions in S_i and w'' is the portion of w inside S_i, we have:

$$\Pr_{w',w''}\left[P(C(x)[(w',w'')_{|S_1}]\dots C(x)[(w',w'')_{|S_{i-1}}]) = C(x)[(w',w'')_{|S_i}]\right] \geq \frac{1}{2} + \frac{\epsilon}{2m}.$$

By an averaging argument, for at least an $\epsilon/(4m)$ fraction of the w', we have:

$$\Pr_{w''}\left[P(C(x)[(w',w'')_{|S_1}]\dots C(x)[(w',w'')_{|S_{i-1}}]) = C(x)[(w',w'')_{|S_i}]\right] \quad \geq \quad \frac{1}{2} + \frac{\epsilon}{4m}.$$

Note that $(w', w'')_{|S_i}$ is just w'', and so this can be rewritten as:

$$\Pr_{w''}\left[P(C(x)[(w',w'')_{|S_1}]\dots C(x)[(w',w'')_{|S_{i-1}}]) = C(x)[w'']\right] \quad \geq \quad \frac{1}{2} + \frac{\epsilon}{4m}. \tag{9}$$

Our reconstruction function is given $A(x, w)$, which contains the bits of $C(x)$ required to evaluate P as above, for all w''. In particular, $A(x, w)$ contains $C(x)[(w', w'')_{|S_j}]$ for all $j < i$. Whenever (9) holds for some w', the evaluations

$$r_{w''} = P(C(x)[(w',w'')_{|S_1}]\dots C(x)[(w',w'')_{|S_{i-1}}])$$

form a string r having $1/2 + \epsilon/(4m)$ relative agreement with $C(x)$. We apply list-decoding to obtain a list of $L = O((4m/\epsilon)^2)$ strings (this upper bound on the list size is guaranteed by the Johnson Bound), one of which is x (when (9) holds). We output a random one of those strings. Altogether

$$\Pr_w[R^D(A(x,w)) = x] \geq \frac{\epsilon}{8m^2} \cdot \frac{\epsilon}{4m} \cdot \frac{1}{L},$$

where the first term on the right-hand-side is the probability that we produce a good next-bit predictor P, the second term is the probability that $w = (w', w'')$ is good (i.e. that (9) holds for w'), and the third term is the probability that we select x from the list of size L. Note that the right-hand-side is at least m^{-c_1} for some universal constant c_1.

Finally, we apply Lemma 2.10 with $\alpha = 1/3$ and $\delta = m^{-c_1}$ to obtain a

$$(n, t = O(\log n), m = n^\beta, d = O(\log n), O(m^{c_1+3}), b = n, \epsilon, 2/3)$$

reconstructive disperser. Choosing $\beta < 1/(c_1 + 3)$ satisfies the statement of the theorem.

6.3 The Ta-Shma-Zuckerman-Safra reconstructive disperser

In this subsection, we describe a quite different, algebraic construction [TSZS01] in our language. Since we are only aiming to satisfy Theorem 2.7, we can choose parameters more liberally than in the original presentation of [TSZS01]. Fix $n, m = n^\beta, \epsilon \geq m^{-1}$, for a constant β to be specified later. Set $q = n^4$. We will use the following error-correcting codes:

- A bivariate (i.e., $\ell = 2$) Reed-Muller code $C_1 : \mathbb{F}_q^{\bar{k}=n/\log q} \to (\mathbb{F}_q)^{q^2}$ with total degree $h = 2\sqrt{n}$.
- A binary Hadamard code $C_2 : \{0,1\}^{\log q} \to \{0,1\}^q$.

The reconstructive disperser has three parts:

- The disperser function $E : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$. Given input x and a seed consisting of $y_1 \in F_q^2$ and $y_2 \in [q]$, the i-th output bit is

$$C_2(C_1(x)[y_1 + (i,0)])[y_2].$$

- The advice function $A : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^a$. We are given an input x and a seed $w \in \{0,1\}^d$. We view w as specifying a random line $L_w : \mathbb{F}_q \to \mathbb{F}_q^2$, together with a random element $p_w \in F_q$ (so $d = 5 \log q = O(\log n)$). We output $C_1(x)$ restricted to $L_w + (i,0)$ for $i = 1, \ldots, (m-1)$ (such a restriction is a degree h polynomial and thus can be specified by $h+1$ elements of $\mathbb{F}_q$), as well as $C_1(x)[L_w(p_w) + (i,0)]$ for $i = m, m+1, \ldots, m+h$. Finally, we output the seed w, for a total output length of $a = (h+1)m \log q + d$.

- The reconstruction function $R : \{0,1\}^a \to \{0,1\}^n$, which works as follows. We are given access to D that ϵ-avoids $Z = E(x, U_t)$, and $A(x,w)$ for a randomly chosen w. By Lemma 6.1 we can convert D into a next-bit predictor P with probability $\epsilon/(8m^2)$, and then by standard techniques (see, e.g, [SU05]), P can be converted to a next-element predictor $P' : \mathbb{F}_q^{i-1} \to \mathbb{F}_q$ for which:

$$\Pr_{y_1}\left[P'(C_1(x)[y_1 + (1,0)], \ldots, C_1(x)[y_1 + (i-1,0)]) = C_1(x)[y_1 + (i,0)]\right] = \rho \geq \Omega(\epsilon/m)^2.$$

Note that $A(x,w)$ contains the portion of $C_1(x)$ required to evaluate P' to obtain "predictions" for $C_1(x)$ restricted to the line $L_w + (m,0)$. We expect ρq correct predictions along this line, and because the points on line $L_w + (m,0)$ are pairwise independent, with probability at least $1 - O(1/(\rho q))$ we obtain at least $\rho q/2$ correct predictions. Efficient list-decoding from these data points is possible as long as $\rho q/2 \geq \sqrt{2hq}$ via Lemma 2.2, and this inequality holds by our choice of q and h. We thus obtain a list of $4/\rho$ candidates for $C_1(x)$ restricted to $L_w + (m,0)$. Now, $A(x,w)$ also contains the value of $C_1(x)$ at a random point p_w along this line. With probability at least $1 - O(\rho^{-2}h/q)$, the correct candidate in the list is the only one agreeing with this random value.

Having learned $C_1(x)$ restricted to line $L_w + (m,0)$, we have recovered the portion of $C_1(x)$ required to repeat the process, to learn $C_1(x)$ restricted to $L_w + (m+1,0)$. After h repetitions we have learned enough of $C_1(x)$ to interpolate and recover x. The probability we succeed on all h repetitions is at least $1 - O(h^2\rho^{-2}/q)$ by a union bound, and this probability is at least $1/2$ by our choice of q and h.

Altogether,

$$\Pr_w[R^D(A(x,w)) = x] \geq \frac{\epsilon}{8m^2} \cdot \frac{1}{2}$$

where the first term on the right-hand-side is the probability that we produce a good next-bit predictor, and the second term is the probability over w that the procedure above outputs x. Note that the right-hand-side is at least m^{-c_1} for some universal constant c_1.

As in the previous subsection, we apply Lemma 2.10 with $\alpha = 1/3$ and $\delta = m^{-c_1}$ to obtain a

$$(n, t = O(\log n), m = n^\beta, d = O(\log n), O(m^{c_1}a), b = n, \epsilon, 2/3)$$

reconstructive disperser. Since $a = O(mn^{1/2} \log n)$, we can choose $\beta < (1/2)/(c_1 + 1)$ to satisfy the statement of the theorem.

6.4 The Shaltiel-Umans reconstructive disperser

In this subsection, we describe the construction of [SU05] in our language. In the present context, this construction is quite similar to the one in the previous subsection – the difference between the two constructions is crucial when viewing them as extractor constructions, but not important for the purpose of obtaining optimal HSGs via the results in this paper.

As above, we are only aiming to satisfy Theorem 2.7, so we choose parameters more liberally than in the presentation in [SU05]. Fix $n, m = n^{\beta}, \epsilon \geq m^{-1}$, for a constant β to be specified later. We also have a parameter ℓ, which will be specified later. Set $h = \ell n^{1/\ell}$ and $q = n^{10}$. We will use the following error-correcting codes:

- A ℓ-variate Reed-Muller code $C_1 : \mathbb{F}_q^{\bar{k}=n/\log q} \to \mathbb{F}_q^{q^\ell}$, with total degree h.
- A binary Hadamard code $C_2 : \{0,1\}^{\log q} \to \{0,1\}^q$.

The reconstructive disperser has three parts:

- The disperser function $E : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$. Let $\alpha : \mathbb{F}_q^\ell \to \mathbb{F}_q^\ell$ be a generator of the multiplicative group of $\mathbb{F}_{q^\ell}$. Given input x and a seed consisting of $y_1 \in F_q^\ell$ and $y_2 \in [q]$, the i-th output bit is
$$C_2(C_1(x)[\alpha^i y_1])[y_2].$$
- The advice function $A : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^a$. We are given an input x and a seed $w \in \{0,1\}^d$. We view w as specifying two random degree $c = \Theta(\ell)$ curves $L_w^1 : \mathbb{F}_q \to \mathbb{F}_q^\ell$ and $L_w^2 : \mathbb{F}_q \to \mathbb{F}_q^\ell$, interleaved in the manner described in [SU05] (so $d = 2(c+1)\ell \log q = O(\ell^2 \log n)$). We output $C(x)$ restricted to $L_w^1 \circ \alpha^i$ and $L_w^2 \circ \alpha^i$ for $i = 1, 2, \ldots, (m-1)$; each restriction can be specified by $ch+1$ elements of $\mathbb{F}_q$. We also output the seed w, for a total output length of $a = 2(ch+1)m \log q + d$.
- The reconstruction function $R : \{0,1\}^a \to \{0,1\}^n$, which works as follows. We are given access to D that ϵ-avoids $Z = E(x, U_t)$, and $A(x, w)$ for a randomly chosen w. As in the previous subsection, we use Lemma 6.1 to convert D into a next-bit predictor P with probability $\epsilon/(8m^2)$, and then into a next-element predictor $P' : \mathbb{F}_q^{i-1} \to \mathbb{F}_q$ for which:
$$\Pr_{y_1}\left[P'(C_1(x)[\alpha^1 y_1], \ldots, C_1(x)[\alpha^{i-1} y_1]) = C_1(x)[\alpha^i y_1]\right] = \rho \geq \Omega(\epsilon/m)^2.$$

  As in the previous subsection, $A(x, w)$ contains the portion of $C_1(x)$ required to evaluate P' to obtain "predictions" for $C_1(x)$ restricted to the curves $L_w^1 \circ \alpha^m$ and $L_w^2 \circ \alpha^m$. The analysis in [SU05] shows that by list-decoding via Lemma 2.2 and pruning the lists according to the intersections of these two curves with portions of $C_1(x)$ contained in $A(x, w)$, we can learn the restriction of $C_1(x)$ to these two curves with probability at least $1 - 1/(4q^\ell)$.

  This sets us up to repeat the process to learn the restrictions of $C_1(x)$ to the curves $L_w^1 \circ \alpha^{m+1}$ and $L_w^2 \circ \alpha^{m+1}$, and then curves $L_w^1 \circ \alpha^{m+2}$ and $L_w^2 \circ \alpha^{m+2}$, and so on, until we have learned all of $C_1(x)$, from which we can recover x. The probability that we succeed on all steps in this process is at least $1/2$ by a union bound.

  The analysis in [SU05] further shows that when $m = n^{\Omega(1)}$ (as it is here), we may choose ℓ to be a constant. This has the important consequence of ensuring that the advice seed length $d = O(\log n)$, and that the number of steps in the reconstruction procedure is $q^\ell = \text{poly}(n)$. As in the previous subsection, we end up with

$$\Pr_w[R^D(A(x,w)) = x] \geq \frac{\epsilon}{8m^2} \cdot \frac{1}{2}$$

where the right-hand-side is at least m^{-c_1} for some universal constant c_1. We finally apply Lemma 2.10 with $\alpha = 1/3$ and $\delta = m^{-c_1}$ to obtain a

$$(n, t = O(\log n), m = n^\beta, d = O(\log n), O(m^{c_1}a), b = n, \epsilon, 2/3)$$

reconstructive disperser. Since $a = O(mn^{1/\ell}\log n)$, we can choose $\beta < (1-1/\ell)/(c_1+1)$ to satisfy the statement of the theorem.

7 Open problems

We mention briefly two interesting open problems related to this work.

First, is it possible to extend these results to two-sided objects, by giving a similar composition for reconstructive *extractors*? Because implicit reconstructivity of extractors is closely related to efficient *implicit* list-decodability, it is possible that such a result would give a new generic construction of implicitly list-decodable codes (in the sense of [STV01]) from *any* good list-decodable codes.

Second, is it possible to extend our result to the non-deterministic setting? Here there is an important technical issue of *resiliency* of HSGs discussed in [GSTS03]; obtaining a resilient HSG construction would lead to so-called "low-end" uniform hardness vs. randomness tradeoffs for the class AM. One possible route to constructing low-end resilient HSGs against nondeterministic circuits is to construct high-end resilient HSGs (typically an easier task) that possess the features needed to apply the composition in this paper – namely an associated advice function $A(x,w)$ computable in polynomial time, with w having length $O(\log n)$. In the currently known resilient construction [MV05], w has length much larger than $O(\log n)$.

Acknowledgements. We thank Ronen Shaltiel for his comments on an early draft of this paper, and the anonymous referees for numerous helpful suggestions.

References

[ACR98] A. E. Andreev, Andrea E. F. Clementi, and J. D. P. Rolim. A new general derandomization method. *Journal of the ACM*, 45(1):179–213, January 1998.

[ACRT99] A. E. Andreev, A. E. F. Clementi, J. D. P. Rolim, and L. Trevisan. Weak random sources, hitting sets, and BPP simulations. *SIAM Journal on Computing*, 28(6), 1999.

[BF99] H. Buhrman and L. Fortnow. One-sided versus two-sided error in probabilistic computation. In *Theoretical aspects of computer science, 16th annual symposium*, 1999.

[BLvM05] H. Buhrman, T. Lee, and D. van Melkebeek. Language compression and pseudorandom generators. *Computational Complexity*, 14(3):228–255, December 2005.

[BM84] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13(4):850–864, 1984.

[GS00] V. Guruswami and M. Sudan. List decoding algorithms for certain concatenated codes. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, 2000.

[GSTS03] D. Gutfreund, R. Shaltiel, and A. Ta-Shma. Uniform hardness vs. randomness tradeoffs for Arthur-Merlin games. *Computational Complexity*, 12(3–4):85–130, 2003.

[Gur04] V. Guruswami. Better extractors for better codes? In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pages 436–444, 2004.

[GVW00] O. Goldreich, S. Vadhan, and A. Wigderson. Simplified derandomization of BPP using a hitting set generator. Technical Report TR00-004, Electronic Colloquium on Computational Complexity, January 2000.

[ISW99] R. Impagliazzo, R. Shaltiel, and A. Wigderson. Near-optimal conversion of hardness into pseudo-randomness. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pages 181–190, 1999.

[ISW00] R. Impagliazzo, R. Shaltiel, and A. Wigderson. Extractors and pseudo-random generators with optimal seed-length. In *Proceedings of the Thirty-second Annual ACM Symposium on the Theory of Computing*, pages 21–23, May 2000.

[ISW03] R. Impagliazzo, R. Shaltiel, and A. Wigderson. Reducing the seed length in the Nisan-Wigderson generator. Full version of [ISW00] and [ISW99]. To appear in Combinatoria. Manuscript, 2003.

[IW97] R. Impagliazzo and A. Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 220–229, 1997.

[Kab02] V. Kabanets. Derandomization: a brief overview. *Bulletin of the European Association for Theoretical Computer Science*, 76:88–103, 2002.

[KI04] V. Kabanets and R. Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004.

[MV05] P. B. Miltersen and N. V. Vinodchandran. Derandomizing Arthur-Merlin games using hitting sets. *Computational Complexity*, 14(3):256–279, December 2005.

[NW94] N. Nisan and A. Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, 1994.

[STV01] M. Sudan, L. Trevisan, and S. Vadhan. Pseudorandom generators without the XOR lemma. *JCSS: Journal of Computer and System Sciences*, 62, 2001.

[SU05] R. Shaltiel and C. Umans. Simple extractors for all min-entropies and a new pseudorandom generator. *Journal of the ACM*, 52(2):172–216, 2005.

[Sud97] M. Sudan. Decoding of Reed Solomon codes beyond the error-correction bound. *Journal of Complexity*, 13, 1997.

[Tre02] L. Trevisan. Extractors and pseudorandom generators. *Journal of the ACM*, 48(4):860–879, 2002.

[TS02] A. Ta-Shma. Storing information with extractors. *Inf. Process. Lett.*, 83(5):267–274, 2002.

[TSUZ01] A. Ta-Shma, C. Umans, and D. Zuckerman. Loss-less condensers, unbalanced expanders, and extractors. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, pages 143–152, 2001.

[TSZ04] A. Ta-Shma and D. Zuckerman. Extractor codes. *IEEE Transactions on Information Theory*, 50(12):3015–3025, December 2004.

[TSZS01] A. Ta-Shma, D. Zuckerman, and S. Safra. Extractors from Reed-Muller codes. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science*, 2001.

[Uma03] C. Umans. Pseudo-random generators for all hardnesses. *Journal of Computer and System Sciences*, 67:419–440, 2003.

[Yao82] A. C. Yao. Theory and applications of trapdoor functions. In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, pages 80–91, 1982.