

Dispersing Hash Functions*

Rasmus Pagh[†]

May 23, 2008

Abstract

We define a family of functions F from a domain U to a range R to be *dispersing* if for every set $S \subseteq U$ of a certain size and random $h \in F$, the expected value of $|S| - |h[S]|$ is not much larger than the expectation if h had been chosen at random from the set of all functions from U to R .

We give near-optimal upper and lower bounds on the size of dispersing families and present several applications where using such a family can reduce the use of random bits compared to previous randomized algorithms. A close relationship between dispersing families and extractors is exhibited. This relationship provides good explicit constructions of dispersing hash functions for some parameters, but in general the explicit construction is left open.

1 Introduction

Universal families of hash functions [1] are widely used in various areas of computer science (data structures, derandomization, cryptography). In this paper we consider families of hash functions from a finite universe U to a finite range R . Suppose that F is a universal family (see definition in Section 4.1). Then for every subset $S \subseteq U$, if we pick a function h uniformly at random from F , the expected value of $|S| - |h[S]|$ is not much larger than the expectation if h had been chosen from the set of all functions from U to R . We call a family F where this is the case *dispersing*. Another way of putting the dispersion property is that the family is good at *distinguishing* the elements of every set S : on average the functions map the elements of S to many different values. For comparison, a universal family is good at distinguishing all *pairs* of elements in U (few functions map a pair to the same value).

While universality implies dispersion, we show in Section 3 that there exist dispersing families that are much smaller than every universal family with the same dispersion property. In other words, dispersion is a property strictly weaker than universality. While our first

*A preliminary version appeared in the proceedings of RANDOM 2000.

[†]IT University of Copenhagen, Rued Langgaards Vej 7, 2300 København S, Denmark. Work done in part while at BRICS, University of Aarhus.

upper bound is nonconstructive, Section 4 explores explicit constructions of small dispersing families. In particular, we exhibit a strong connection to the construction of *extractors*.

Small families of functions with random properties are important for derandomization (removing or reducing the use of random bits in algorithms). It is hard to characterize the situations in which a dispersing family could be used instead of a universal one. Indeed, the derandomization examples given in Section 5 use dispersing families in a somewhat different way than one would use a universal family. We also give an example from the literature where replacing a universal family with a dispersing one immediately gives an improved result.

1.1 Related work

The dispersion property of universal families was shown and first used in [7]. It has since found application in several papers [6, 11, 16]. Mehlhorn [12] has given tight bounds (up to a constant factor) on the number of bits needed to represent universal hash functions, i.e., determined the size of such families up to a polynomial.

Another way of stating the dispersion property of a hash function family F is that for random $h \in F$, $\mathbf{E}|h[S]|$ should be “large”. The definition of a *disperser* is similar to this in that one requires $\mathbf{E}|\bigcup_{h \in F} h[S]|$ to be “large”. However, in the usual treatment of dispersers, the range R has size $|S|$ or smaller (whereas we will be interested in $|R| \geq |S|$), and “large” means greater than $(1 - \epsilon)|R|$, for some choice of the parameter ϵ (while we can only hope for some fraction of $|S|$). Nisan’s survey [15] gives a good introduction to dispersers. It also covers the stronger notion of extractors, where the requirement is near-uniformity of the random variable $h(x)$, for uniformly and independently chosen $h \in F$ and $x \in S$. As we will see in Section 4, extractors are closely related to dispersing hash function families. Construction of extractors and dispersers has been intensely researched in recent years. We refer to [20] and the references therein for an overview. The extractors considered in this paper are *seeded extractors*, and not the kind of *deterministic* extractors considered in some recent works on randomness extraction. Also, we need extractors of the type that extract all the entropy from the source.

1.2 Notation

In the following, S denotes a subset of $U = \{1, \dots, u\}$, $|S| = n$, and we consider functions from U to $R = \{1, \dots, r\}$ where $r \geq n > 1$ and $u \geq 2r$. The set of all functions from U to R is denoted by $(U \rightarrow R)$, and $\binom{U}{n}$ denotes the set of subsets of U of size n . The number of *overflow elements* for S under h is $C(S, h) = n - |h[S]|$ (this is the number of elements that could not be placed if we put the maximum number of elements of S in an array according to h -value). A uniform random choice is denoted by $\in_{\mathbf{R}}$, and is always independent of other events. The base of “log” is 2, and “ln” denotes the natural logarithm.

2 The family of all functions

As preparation for the results on dispersing families, this section contains some results on the distribution of $C(S, h)$ for $h \in_{\mathbf{R}} (U \rightarrow R)$. The probability that $i \notin h[S]$ is $(1 - \frac{1}{r})^n$ for $i \in \{1, \dots, r\}$. Thus the expected size of $R \setminus h[S]$ is $(1 - \frac{1}{r})^n r$, and the expected size of $h[S]$ is:

$$\mu := r \left(1 - \left(1 - \frac{1}{r}\right)^n\right) = r \left(\binom{n}{1}/r - \binom{n}{2}/r^2 + \dots - (-1)^n \binom{n}{n}/r^n\right) = n - \Theta\left(\frac{n^2}{r}\right). \quad (1)$$

Hence the expected value of $C(S, h)$ is:

$$\lambda := n - \mu = \sum_{i=2}^n (-1)^i \binom{n}{i} r^{1-i} \in \left[\frac{n^2}{4r}, \frac{n^2}{2r}\right]. \quad (2)$$

We now turn to giving tail bounds. Let $S = \{s_1, \dots, s_n\}$ and let $B_{i,k}$ be the indicator random variable that is 1 if $h(s_k) = i$. These random variables are known to be *negatively associated* [3]. Note that $C(S, h) = \sum_i B_i$, where $B_i := \max(0, |\{k \mid h(s_k) = i\}| - 1)$. Since the B_i are non-decreasing functions of disjoint subsets of the $B_{i,k}$, the B_i are also negatively associated. As shown in [3] this implies that standard Chernoff bounds can be applied on the sum $\sum_i B_i$. We use the following two bounds (see e.g. [14, Theorem 4.1 and 4.2]):

$$\Pr[C(S, h) \leq c\lambda] \leq \exp(-(1-c)^2\lambda/2), \quad \text{for } c \leq 1. \quad (3)$$

$$\Pr[C(S, h) \geq c\lambda] \leq \left(\frac{e^{c-1}}{c^c}\right)^\lambda, \quad \text{for } c \geq 1. \quad (4)$$

Analogously, for a sequence $h_1, \dots, h_b \in_{\mathbf{R}} (U \rightarrow R)$, one can derive the following estimate, for $c \geq 1$:

$$\Pr\left[\frac{1}{b} \sum_{i=1}^b C(S, h_i) \geq c\lambda\right] \leq \left(\frac{e^{c-1}}{c^c}\right)^{\lambda b}. \quad (5)$$

3 Dispersing families

In this section we show upper and lower bounds on the minimal size of dispersing families, formally defined as follows:

Definition 1 *A family of functions $F \subseteq (U \rightarrow R)$, where $|U| = u$ and $|R| = r$, is (c, n, r, u) -dispersing if for every $S \subseteq U$, $|S| = n$, the expected value of $C(S, h)$ for $h \in_{\mathbf{R}} F$ is at most $c\lambda$, where λ is given by (2). When parameters n , r and u follow from the context, we will use the term c -dispersing.*

By definition of λ , the family $(U \rightarrow R)$ is 1-dispersing. Thus, the parameter c is a measure of how well a random function from a family disperses compared to a function chosen at random from $(U \rightarrow R)$.

We now give a simple nonconstructive argument (using the probabilistic method) that a small family of c -dispersing functions exists for $c \geq 1 + \epsilon$, where $\epsilon > 0$ is a constant. Let $k(c) = \ln(c^c/e^{c-1}) = \Theta(c \log c)$. The family can be constructed by picking $h_1, \dots, h_b \in_{\mathbf{R}} (U \rightarrow R)$, where $b > \frac{n \ln(ue/n)}{k(c)\lambda} = O\left(\frac{r \log(u/n)}{n c \log c}\right)$. We now argue that with nonzero probability this gives a family with the desired property, namely $\frac{1}{b} \sum_{i=1}^b C(S, h_i) \leq c \lambda$ for every $S \in \binom{U}{n}$. By inequality (5),

$$\Pr \left[\frac{1}{b} \sum_{i=1}^b C(S, h_i) > c \lambda \right] \leq \left(\frac{e^{c-1}}{c^c} \right)^{\lambda b} < (ue/n)^{-n} .$$

Since there are fewer than $(ue/n)^n$ sets in U of size n (see, e.g., [10, Section 1.1]), the probability of failure for at least one set is less than 1, as desired.

We now show a lower bound on the size of a c -dispersing family for $n > 1$. Take some such family $F = \{h_1, \dots, h_b\}$. We construct U_1, \dots, U_k , with $U_k \subseteq U_{k-1} \subseteq \dots \subseteq U_1 \subseteq U_0 = U$, such that $|U_i| \geq u (\lceil n/2 \rceil / r)^i$ and $|h_i[U_k]| \leq \lceil n/2 \rceil$ for $i \leq k$. The set U_i is constructed from U_{i-1} using the pigeonhole principle to pick a subset U_i with $|h_i[U_i]| \leq \lceil n/2 \rceil$ of size at least $|U_{i-1}| \lceil n/2 \rceil / r$. (U_i is simply the union of the elements in the $\lceil n/2 \rceil$ most loaded ‘‘buckets’’ when mapping U_{i-1} with h_i .) Setting $k = \lfloor \log(u/n) / \log(r/\lceil n/2 \rceil) \rfloor$ we have $|U_k| \geq n$ and can take $S \subseteq U_k$ of size n . Observe that $k \geq \log(u/n) / (2 \log(2r/n)) \geq 1$. Since F is c -dispersing we must have $\sum_i C(S, h_i) \leq b c \lambda$. On the other hand, $C(S, h_i) \geq n - \lceil n/2 \rceil = \lfloor n/2 \rfloor$ for $i \leq k$ by construction, so $\sum_i C(S, h_i) \geq k \lfloor n/2 \rfloor \geq kn/3$ (using that $n > 1$ and n is integer). Therefore we must have:

$$b \geq \frac{kn/3}{c \lambda} \geq \frac{r \log(u/n)}{3 n c \log(2r/n)} .$$

We summarize the bounds as follows:

Theorem 2 *For $r \geq n > 1$, $u \geq 2r$ and $c > 1 + \epsilon$, for constant $\epsilon > 0$, a minimal size (c, n, r, u) -dispersing family F satisfies:*

$$\frac{r \log(u/n)}{3 n c \log(2r/n)} \leq |F| = O\left(\frac{r \log(u/n)}{n c \log c}\right) .$$

The gap between the bounds is $O\left(\frac{\log(2r/n)}{\log c}\right)$. This is a tight bound in many cases: The parameter c ranges from $1 + \epsilon$ to $O(r/n)$, and for $c = (r/n)^{\Omega(1)}$ the bounds differ by a constant factor.

A random function h from a $(\frac{\delta n}{\lambda}, n, r, u)$ -dispersing family has expected size of $h[S]$ at least $(1 - \delta)n$. This makes the following version of Theorem 2 convenient.

Corollary 3 *For $r \geq n > 1$, $u \geq 2r$ and $\delta > (1 + \epsilon)\lambda/n$, for constant $\epsilon > 0$, a minimal size $(\frac{\delta n}{\lambda}, n, r, u)$ -dispersing family F satisfies:*

$$\frac{\log(u/n)}{8 \delta \log(2r/n)} \leq |F| = O\left(\frac{\log(u/n)}{\delta \log(2\delta r/n)}\right) .$$

4 Explicit constructions

This section concerns the explicit construction of dispersing families, concentrating on $O(1)$ -dispersing families. By explicit families $F_{c,n,r,u}$ we mean that there is a Turing machine that, given parameters c, n, r and u , the number of some function f in (an arbitrary ordering of) $F_{c,n,r,u}$, and $x \in U$, computes $f(x)$ in time $(\log |F_{c,n,r,u}| + \log(u + c))^{O(1)}$. The general goal, only reached here for some parameters, would be explicit families of sizes polynomially related to the bounds of Theorem 2. Picking a random function from such a family uses a number of random bits that is within a constant factor of optimal, i.e., the *sample complexity* is optimal.

4.1 Universal families

Definition 4 *A family $F \subseteq (U \rightarrow R)$ is c -universal if for all $x, y \in U$, $x \neq y$ and $h \in_R F$, $\Pr[h(x) = h(y)] \leq c/r$. It is strongly c -universal if for all $a, b \in R$, $\Pr[h(x) = a, h(y) = b] \leq c/r^2$.*

One strongly $(1 + \epsilon)$ -universal (and thus $(1 + \epsilon)$ -universal) family for $0 < \epsilon \leq 1$ is:

$$F_{\text{su}} = \{x \mapsto ((tx + s) \bmod p) \bmod r \mid m/2 \leq p \leq m, p \text{ prime}, 0 \leq s, t < p\}$$

where $m = 8r^2 \log(u)/\epsilon$. The universality proof is given in appendix A. Note that the family has size $(r \log(u)/\epsilon)^{O(1)}$, and that, given parameters s, t and p , we can compute a function value in polynomial time. To get universal families with parameter $c \geq 2$, we note that if we take a subset of size $2|F|/c$ of a 2-universal family F , then this subset is a c -universal family.

We now establish that universal families are dispersing, slightly generalizing an observation in [7]:

Proposition 5 *A c -universal family is $2c$ -dispersing.*

Proof. Let F be a c -universal family, and take $S \in \binom{U}{n}$. For $h \in_R F$ consider $K(S, h) = |\{\{x, y\} \in \binom{S}{2} \mid h(x) = h(y)\}|$. Since $C(S, h) \leq K(S, h)$ we just need to bound the expected value of $K(S, h)$. By c -universality this is at most $\binom{n}{2} c/r$, and by (2) we have the bound $\binom{n}{2} c/r < cn^2/(2r) \leq 2c\lambda$. \square

Mehlhorn [12] has shown that a c -universal family can have size no smaller than $r(\lceil \log u / \log r \rceil - 1)/c$. This is $\Omega(n \log c / \log r)$ times larger than the upper bound of Theorem 2. Hence, dispersion is a property strictly weaker than universality.

For $r \geq n^{1+\Omega(1)}$ the lower bound of Theorem 2 is $r^{\Omega(1)} \log(u)/c$. When $c = O(1)$ this is polynomially related to the size of F_{su} . Thus we have the following:

Corollary 6 *There exists an explicit $O(1)$ -dispersing family of optimal sample complexity for $r = n^{1+\Omega(1)}$.*

4.2 Dispersing families from extractors

This section addresses the construction of $O(1)$ -dispersing families for $r = n^{1+o(1)}$, where universal families do not have optimal sample complexity (except for very large universes). We give an explicit construction of an $O(1)$ -dispersing family from an *extractor* (see the definition below). Plugging in an explicit optimal extractor would yield an explicit $O(1)$ -dispersing family with optimal sample complexity (except perhaps for very small universes). Without loss of generality we may consider only the case where r is a power of two. This is because an $O(1)$ -dispersing family with range $[2^{\lceil \log r \rceil}]$ is also an $O(1)$ -dispersing family with range $[r]$.

Definition 7 A random variable X with values in a finite set T is ϵ -close to uniform if

$$\sum_{i \in T} |\Pr[X = i] - 1/|T|| \leq 2\epsilon .$$

It has min-entropy κ if

$$\max_{i \in T} \Pr[X = i] \leq 2^{-\kappa} .$$

Note that every random variable with min-entropy κ automatically also has minentropy κ' for all $\kappa' < \kappa$.

Definition 8 A function $E : U \times \{0, 1\}^s \rightarrow \{0, 1\}^t$ is an (n, ϵ) -extractor if for every random variable X with values in U and min-entropy $\log n$, the distribution of $E(X, y)$ for $y \in_{\mathbb{R}} \{0, 1\}^s$ is ϵ -close to uniform over $\{0, 1\}^t$. The parameter s is called the seed length.

Nonconstructive arguments show that for $t \leq \log n$ and $\epsilon > 0$ there exist (n, ϵ) -extractors with $s = O(\log(\log(u)/\epsilon))$. As mentioned in the introduction, much research effort is currently directed towards explicit construction of such functions. We will make use of the following folklore fact [13], which gives an alternative characterization of extractors:

Lemma 9 A function $E : U \times \{0, 1\}^s \rightarrow \{0, 1\}^t$ is an (n, ϵ) -extractor if and only if for every $S \subseteq U$ with $|S| = n$, the distribution of $E(x, y)$ for $x \in_{\mathbb{R}} S$ and $y \in_{\mathbb{R}} \{0, 1\}^s$ is ϵ -close to uniform over $\{0, 1\}^t$.

Proof. (Sketch.) The “only if” direction is obvious since the distribution of x has the required min-entropy. For the other direction we use the fact that every source X of min-entropy $\log n$ is a *convex combination* of sources that are uniformly distributed on n points in U . This can be seen by induction on the number of points x for which $\Pr[X = x] \notin \{0, 1/n\}$.

In turn, this implies that $E(X, y)$ is a convex combination of distributions that are ϵ -close to uniform over $\{0, 1\}^t$. Specifically, $E(X, y)$ itself is ϵ -close to uniform. \square

Theorem 10 Let $u, n, r,$ and t be integers such that $1 < n \leq r \leq u/2, r \geq 2^t,$ and r is a power of 2. We define $U = \{1, \dots, u\}$ and $R = \{1, \dots, r\}$. For some $\epsilon > 0$ suppose that $E : U \times \{0, 1\}^s \rightarrow \{0, 1\}^t$ is an $(\lfloor n/2 \rfloor, \epsilon)$ -extractor, where $\epsilon = O(n/r), F' \subseteq (U \rightarrow \{0, 1\}^s)$ is strongly $(1 + \epsilon)$ -universal, and $F'' \subseteq (U \rightarrow \{0, 1\}^{\log(r)-t})$ is 2-universal. Then the family

$$F_1 = \{x \mapsto E(x, f'(x)) \circ f''(x) \mid f' \in F', f'' \in F''\} \subseteq (U \rightarrow R)$$

where \circ denotes bit string concatenation, is $O(1)$ -dispersing.

Proof. Throughout this proof we let $S \in \binom{U}{n}, x \in_R S,$ and $z \in_R \{0, 1\}^s.$ By the extractor property, the distribution of $E(x, z)$ is ϵ -close to uniform. We can therefore identify a set $B \subseteq \{0, 1\}^t$ of “bad” points, such that $|B| \leq \epsilon 2^t$ and for $i \in \{0, 1\}^t \setminus B$ we have:

$$\Pr[E(x, z) = i] \leq 2^{-t+1} . \quad (6)$$

Now let $f' \in_R F', f'' \in_R F'',$ and define the function $h(y) = E(y, f'(y)) \circ f''(y)$ (note that h is a random function from F_1). We wish to argue that the distribution of $E(x, f'(x))$ is γ -close to uniform for $\gamma = O(\epsilon).$ This is because for every fixed value of $x, x = \bar{x},$ the distribution of $f'(\bar{x})$ is ϵ -close to uniform, and therefore $E(\bar{x}, f'(\bar{x}))$ is ϵ -close to the distribution $E(\bar{x}, z).$ We can bound the value of $C(S, h)$ by the sum of two terms: 1) The number of elements in S mapping to $B,$ and 2) The number of pair-wise collisions *outside* of $B.$ Formally:

$$\begin{aligned} \mathbf{E}[C(S, h)] &\leq \mathbf{E}[|\{x \in S \mid E(x, f'(x)) \in B\}|] + \\ &\mathbf{E}[|\{\{x_1, x_2\} \in \binom{S}{2} \mid h(x_1) = h(x_2) \wedge E(x_1, f'(x_1)) \notin B \wedge E(x_2, f'(x_2)) \notin B\}|] . \end{aligned} \quad (7)$$

The first term is:

$$n \Pr[E(x, f'(x)) \in B] \leq (\gamma + \epsilon) n = O(n^2/r) . \quad (8)$$

For $\{x_1, x_2\} \in_R \binom{S}{2},$ the second term is:

$$\begin{aligned} &\binom{n}{2} \Pr[h(x_1) = h(x_2) \wedge E(x_1, f'(x_1)) \notin B \wedge E(x_2, f'(x_2)) \notin B] \\ &= \binom{n}{2} \sum_{i \in \{0, 1\}^t \setminus B} \Pr[E(x_1, f'(x_1)) = E(x_2, f'(x_2)) = i \wedge f''(x_1) = f''(x_2)] \\ &\leq \binom{n}{2} 2^{-\log(r)+t+1} \sum_{i \in \{0, 1\}^t \setminus B} \Pr[E(x_1, f'(x_1)) = E(x_2, f'(x_2)) = i] . \end{aligned} \quad (9)$$

To bound $\Pr[E(x_1, f'(x_1)) = E(x_2, f'(x_2)) = i]$ for $i \in \{0, 1\}^t \setminus B,$ we note that the random choice $\{x_1, x_2\} \in_R \binom{S}{2}$ can be thought of in the following way: First choose $S_1 \in_R \binom{S}{\lfloor n/2 \rfloor}$ and then choose $x_1 \in_R S_1, x_2 \in_R S \setminus S_1.$ By symmetry, this procedure yields the same distribution. Let $z_1, z_2 \in_R \{0, 1\}^s.$ For every choice of S_1 we choose x_1 and x_2 independently from disjoint sets of size at least $\lfloor n/2 \rfloor.$ For fixed S_1 and $S_2,$ since f' is strongly $(1 + \epsilon)$ -universal, the tuple $(x_1, x_2, f'(x_1), f'(x_2))$ is close to uniform on $S_1 \times S_2 \times \{0, 1\}^s \times \{0, 1\}^s$ in the sense that

the probability of each particular value is at most $1 + \epsilon$ times higher than in the uniform distribution. Using that E is an $(\lfloor n/2 \rfloor, \epsilon)$ -extractor this means:

$$\begin{aligned}
& \Pr[E(x_1, f'(x_1)) = E(x_2, f'(x_2)) = i] \\
& \leq (1 + \epsilon) \Pr[E(x_1, z_1) = E(x_2, z_2) = i] \\
& = (1 + \epsilon) \Pr[E(x_1, z_1) = i] \Pr[E(x_2, z_2) = i] \\
& \leq (1 + \epsilon) (2^{-t+1})^2 .
\end{aligned} \tag{10}$$

Plugging this into (9) we obtain:

$$\begin{aligned}
& \binom{n}{2} \Pr[h(x_1) = h(x_2) \wedge E(x_1, f'(x_1)) \notin B \wedge E(x_2, f'(x_2)) \notin B] \\
& \leq n^2 2^{-\log(r)+t} 2^t (1 + \epsilon) (2^{-t+1})^2 \\
& = 4(1 + \epsilon) n^2 / r = O(n^2 / r) .
\end{aligned} \tag{11}$$

Hence, the expected value of $C(S, h)$ is $O(n^2 / r)$, as desired. \square

Corollary 11 *Let u, n, r , and t be integers such that $1 < n \leq r \leq u/2$, $r \geq 2^t$, and r is a power of 2. We define $U = \{1, \dots, u\}$ and $R = \{1, \dots, r\}$. Given an explicit $(\lfloor n/2 \rfloor, \epsilon)$ -extractor $E : U \times \{0, 1\}^s \rightarrow \{0, 1\}^t$ with $\epsilon = O(n/r)$ and $t = \log n - O(1)$, there is an explicit $O(1)$ -dispersing family $F_1 \subseteq (U \rightarrow R)$ with sample complexity $O(\log(r \log(u)/n) + s)$.*

Proof. We use the construction of Section 4.1 for the universal families in the construction above. The number of bits needed to sample $f' \in_R F'$ is $O(\log(2^s \log(u)/\epsilon)) = O(s + \log(r \log(u)/n))$. The number of bits needed to sample $f'' \in F''$ is $O(\log(2^{\log(r)-t} \log u)) = O(\log(r \log(u)/n))$. \square

Of course, Theorem 10 and Corollary 11 are trivial in cases where the $O(1)$ -parameter of the dispersing family is bigger than $n/\lambda = O(r/n)$. In these cases we can get a nontrivial family by using Corollary 11 to obtain an $O(1)$ -dispersing family with range $\{1, \dots, r'\}$, where r'/r is a suitably large constant power of 2. To get the family F with range R , simply cut away $\log(r'/r)$ bits of the output. This only decreases sizes of images of sets by a constant factor, which means that for $h \in_R F$ and $S \in \binom{U}{n}$ the expected size of $h[S]$ is still $\Omega(n)$.

4.2.1 Explicit extractors

First, we remark that we need extractors of the type that extracts *all* the entropy from the source, i.e., where $2^t \approx n$. (Much work has been done on extractors that extract less entropy, $2^t \ll n$.) The best explicit extractor in current literature with the parameters required by Corollary 11 (output of $\log n - O(1)$ bits and error n/r) has seed length [18]:

$$s = O\left(\min\left(\log^2(\log(u)r/n) \log \log n, \log^2(\log u) \log(n/r) \log \log n\right)\right) .$$

This should be compared to the minimum seed length of an $O(1)$ -dispersing family, which is roughly $O(\log \log u + \log(r/n))$. For constant ϵ (applicable when $r = O(n)$), the best

known seed length is $O(\log \log u + (\log \log n)^{2+o(1)})$ [21]. In particular, when $\log \log u \geq (\log \log n)^{2+\Omega(1)}$ this gives an $O(1)$ -dispersing family of functions with $r = O(n)$ that has size $(\log u)^{O(1)}$, which is polynomial in the lower bound of Theorem 2.

4.3 Extractors from dispersing families

This section points out that nontrivial $O(1)$ -dispersing families with range $\{0, 1\}^{\log n}$ are also (n, ϵ) -extractors for a constant $\epsilon < 1$.

Proposition 12 *Let U be a set, and let n, t , and s be integers, $t = \log n - O(1)$. Suppose that a family of functions $\{h_z\}_{z \in \{0,1\}^s} \subseteq (U \rightarrow \{0,1\}^t)$ is such that for every $S \in \binom{U}{n}$, and for $z \in_{\mathbb{R}} \{0,1\}^s$, we have $\mathbf{E}[|h_z[S]|] = \Omega(n)$. Then $E(x, z) := h_z(x)$ is an $(n, 1 - \Omega(1))$ -extractor.*

Proof. Let $S \in \binom{U}{n}$. For $x \in_{\mathbb{R}} S$ and $z \in_{\mathbb{R}} \{0,1\}^s$, let $B \subseteq \{0,1\}^t$ consist of the values $i \in \{0,1\}^t$ for which $\Pr[h_z(x) = i] < 2^{-t}$. We have:

$$\sum_{i \in B} (2^{-t} - \Pr[h_z(x) = i]) = 1 - \sum_{i \in \{0,1\}^t} \min\{\Pr[h_z(x) = i], 2^{-t}\} \leq 1 - \mathbf{E}[|h_z[S]|]/2^t = 1 - \Omega(1) .$$

This implies that the distribution of $E(x, z)$ is $(1 - \Omega(1))$ -close to uniform. By Lemma 9 this implies that E is an $(n, 1 - \Omega(1))$ -extractor. \square

It should be noted that there is an efficient explicit way of converting an extractor with nontrivial constant error into an extractor with almost any smaller error [17]. Unfortunately, this conversion slightly weakens other parameters, so the problem of constructing optimal extractors with small error cannot be said to be quite the same as that of constructing optimal dispersing families.

5 Applications

The model of computation used for our applications is a unit-cost RAM with word size w . We assume that the RAM supports dispersing families, i.e., given the parameters of a dispersing family, a “function number” i and an input x , it can compute $f_i(x)$ in constant time, where f_i is the i th function from a dispersing family with the given parameters and optimal sample complexity. (Unfortunately, the constructions the previous section cannot be used to explicitly construct dispersing families with the required parameters.) The RAM also has an instruction to generate random numbers.

5.1 Relational joins

Suppose we have two lists of at most n machine words in total, each of which has no duplicate elements. We consider the problem of identifying all elements that occur in both lists. This task underlies *join* operations, a fundamental primitive in relational databases. Other

applications are determining whether two sets are identical or whether one is a subset of the other.

Comparison-based algorithms for these problems require $\Omega(n \log n)$ time, and performing a relational join is easily reduced to sorting. The currently best deterministic linear-space sorting algorithm runs in time $O(n \log \log n)$ [9]. Using randomization and universal hashing, the time for relational joins can be improved to $O(n)$, expected, using linear space. The number of random bits used for this is $\Omega(\log n + \log w)$. We now show how to decrease the number of random bits to $O(\log w)$, and still solve the problem in expected $O(n)$ time and linear space, using dispersing hash functions.

Pick a function h at random from an $(n/\log n, n, n^2, 2^w)$ -dispersing family. According to Theorem 2 there is such a family of size $O(w)$, which means that the sample complexity is $O(\log w)$ bits. The algorithm goes as follows. First compute the function values under h for all elements in the two lists, and sort the elements into buckets according to their hash function values. This can be done in time $O(n)$ using radix sort. All duplicates in buckets with at most two elements can be identified by a linear time scan. We then list the elements in buckets of size more than two, and sort them using an $O(n \log n)$ time sorting algorithm. As a last step, we go through the sorted list and report all duplicates.

The correctness of the algorithm is straightforward. It remains to be argued that the expected running time is $O(n)$. Let S_1 and S_2 be the two sets of elements. For the sake of analysis let S' be a set of $n - |S_1 \cup S_2|$ elements, disjoint from S_1 and S_2 . By definition of dispersing families, the expected size of $h[S_1 \cup S_2 \cup S']$ is $n - O(n/\log n)$. In particular, the number of elements in the set $S_1 \cup S_2$ that share their function value with another element in $S_1 \cup S_2$ is $O(n/\log n)$, expected. Since the elements in the two lists are distinct, this means that the expected number of elements in buckets of size more than two is $O(n/\log n)$, meaning that the sorting step takes $O(n)$ expected time.

5.2 Element distinctness

We now consider the element distinctness problem for a list of n machine words. Again, in a comparison-based model this problem requires time $\Omega(n \log n)$, and the problem can be reduced to sorting. As before, with universal hashing the problem can be solved in expected linear time and linear space. We show how to decrease the number of random bits to $O(\log w)$, using dispersing hash functions. The method is similar to the one in Section 5.1, but we describe it here for completeness.

Again, pick h at random from an $(n/\log n, n, n^2, 2^w)$ -dispersing family of size $O(w)$. Arguing as above, the expected size of $h[S]$ is $|S| - O(n/\log n)$, where S is the set of machine words considered. Words involved in a collision are put into a balanced binary search tree, each in time $O(\log n)$. Since no more than $2(|S| - |h[S]|)$ elements can be inserted before a duplicate (if any) is found, this takes time $O((|S| - |h[S]|) \log n)$, which is expected $O(n)$.

Remark. It would be interesting if the more general problem of determining the number of distinct elements in a list of machine words (the set cardinality problem), could be solved in a similar way. Possibly, a slightly stronger property than dispersion is needed.

5.3 Static dictionary construction

The next problem considered is that of constructing a static dictionary, i.e., a data structure for storing a set $S \subseteq U = \{0, 1\}^w$, $|S| = n$, allowing constant time lookup of elements (plus any associated information) and using $O(n)$ words of memory. The best known deterministic algorithm runs in time $O(n \log n)$ [8]. In the case $u = n^{O(1)}$ there exists a construction algorithm running in time $O(n \log \log n)$ [19].

Randomized algorithms running in time $O(n)$ can be made to use as few as $O(\log n + \log w)$ random bits [2]. Here we will show how to achieve another trade-off, namely expected time $O(n \log \log n)$ using $O(\log w)$ random bits. It suffices to reduce the general case to the case $u = n^{O(1)}$, where the efficient deterministic algorithm [19] can be applied.

Picking random functions from an $(n/\log n, n, n^2, 2^w)$ -dispersing family of size $O(w)$, we can first reduce our problem to two subproblems: one within a universe of size n^2 and one with $O(n/\log n)$ colliding elements. The set of $O(n/\log n)$ elements can be handled in time $O(n)$ using the deterministic $O(n \log n)$ algorithm. The set within a universe of size n^2 is handled by the deterministic $O(n \log \log n)$ time algorithm.

Note that when randomly picking a hash function, there is a positive constant probability of finding one reducing the problem to two subproblems with the desired parameters. Thus, the expected number of attempts before a suitable function is found is constant. This means that the expected number of random bits needed to find a suitable reduction function is $O(\log w)$.

We sum up our derandomization results as follows:

Theorem 13 *On a unit cost RAM supporting dispersing families we can, using $O(\log w)$ random bits, expected, and space $O(n)$:*

- *Perform relational join in expected time $O(n)$.*
- *Solve the element distinctness problem in expected time $O(n)$.*
- *Construct a static dictionary (supporting constant time queries) in expected time $O(n \log \log n)$.*

5.4 An implicit dictionary

As a final application, we consider the implicit dictionary scheme of Fiat et al. [6]. In an implicit dictionary, the elements of S are placed in an array of n words. A result of Yao states that without extra memory, a lookup requires $\log n$ table lookups in the worst case [22]. The question considered in [6] is how little extra memory is needed to enable constant worst case lookup time. The information stored outside the table in the construction of [6] is the description of (essentially) a universal hash function, occupying $O(\log n + \log w)$ bits. However, the only requirement on the function is that it is, say, $(2, n, O(n), 2^w)$ -dispersing, so we can reduce the extra memory to $O(\log w)$ bits. (This result was also shown in a follow-up paper [5], using an entirely different nonconstructive argument.)

6 Open problems

The most obvious open problem is to find explicit dispersing families of close to minimal size for a wider range of parameters. As shown in Section 4.2, explicit $O(1)$ -dispersing families with optimal sample complexity will follow if optimal extractors are constructed, and such families are themselves extractors with nontrivial error. Secondly, the issue of constructing explicit c -dispersing families with parameter c close to r/n is interesting in view of the derandomization applications given in Section 5.

We note that our applications would even require construction of efficient circuits for dispersing families, to justify the assumption of a RAM model in which function evaluation takes constant time. While current state-of-the-art seems far from achieving this, there are practical function families that are plausible candidates for being optimal dispersing families. One such example is a family of functions of the form $x \mapsto x \bmod p_i$, where p_i is taken from a set of $O(w)$ primes of size $\Theta(r)$.

Acknowledgments: The author would like to thank Martin Dietzfelbinger, Johan Kjeldgaard-Pedersen, Peter Bro Miltersen, and Mikkel Thorup for useful help and feedback during this work. Also thanks to two anonymous reviewers for helping to greatly improve the presentation.

A Universality proof

This appendix gives a proof of strong $(1 + \epsilon)$ -universality, for arbitrarily small $\epsilon > 0$, of a small explicit family of functions. To the knowledge of the author, such a proof never appeared explicitly in the literature. However, the proof is along the lines of the universality proof in [7].

Theorem 14 *Given integers $r > 1$ and $u \geq 2r$, let $U = \{1, \dots, u\}$ and $R = \{1, \dots, r\}$. For $0 < \epsilon \leq 1$ and $m \geq 8r^2 \log(u)/\epsilon > 45$ the following family of functions from U to R is strongly $(1 + \epsilon)$ -universal:*

$$F_{\text{su}} = \{x \mapsto ((tx + s) \bmod p) \bmod r \mid m/2 \leq p \leq m, p \text{ prime}, 0 \leq s, t < p\} .$$

Proof. For all $a, b \in R$, $x, y \in U$ ($x \neq y$), and for $h \in_{\mathbb{R}} F_{\text{su}}$, we must show that $\Pr[h(x) = a \wedge h(y) = b] \leq (1 + \epsilon)/r^2$. So pick a random prime p in the range $m/2$ to m and $s, t \in_{\mathbb{R}} \{0, \dots, p - 1\}$.

We need the following number-theoretic fact: The number of primes in the interval $(m/2; m]$, denoted $\pi(m) - \pi(m/2)$, is at least $m/(3 \ln m)$. This can be shown, for example, by using the upper and lower bounds on $\pi(m/2)$ and $\pi(m)$ of Dusart [4], for $m > 45$:

$$\pi(m) - \pi(m/2) \geq (m/\ln m) - \frac{4}{3} \frac{m}{2} / \ln(m/2) \geq m/(3 \ln m) .$$

Since $|x - y|$ has at most $\frac{\ln u}{\ln(m/2)}$ prime divisors larger than $m/2$ this implies that p divides $|x - y|$ with probability at most

$$\frac{\ln u}{\ln(m/2)} \frac{3 \ln m}{m} < 6 \log(u)/m \leq \frac{3}{4} \epsilon / r^2$$

using $m > 45$ for the first inequality.

Assume in the following that p does not divide $|x - y|$. Then it holds that the vector

$$(tx + s, ty + s) = \begin{pmatrix} x & 1 \\ y & 1 \end{pmatrix} \begin{pmatrix} t \\ s \end{pmatrix}$$

is uniform on \mathbb{Z}_p^2 . This is because the matrix is invertible, and hence induces a bijection. There are at most $\lceil p/r \rceil^2$ different values of $(tx + s, ty + s)$ that will result in $h(x) = a$ and $h(y) = b$. This means that:

$$\begin{aligned} \Pr[h(x) = a \wedge h(y) = b] &\leq \lceil p/r \rceil^2 / p^2 \\ &< 1/r^2 + 2/(pr) + 1/p^2 \\ &= (1 + 2r/p + (r/p)^2)/r^2 \\ &< (1 + \epsilon/4)/r^2 . \end{aligned}$$

The last inequality uses that $r/p \leq \frac{r}{m/2} \leq \epsilon/16$ since $r > 1$ and $u \geq 2r$. Summing up, the probability that $h(x) = a$ and $h(y) = b$ is at most $(1 + \epsilon)/r^2$, as desired. \square

References

- [1] J. Lawrence Carter and Mark N. Wegman. Universal classes of hash functions. *J. Comput. System Sci.*, 18(2):143–154, 1979.
- [2] Martin Dietzfelbinger, Joseph Gil, Yossi Matias, and Nicholas Pippenger. Polynomial hash functions are reliable (extended abstract). In *Proceedings of the 19th International Colloquium on Automata, Languages and Programming (ICALP '92)*, volume 623 of *Lecture Notes in Computer Science*, pages 235–246. Springer-Verlag, 1992.
- [3] Devdatt P. Dubhashi and Desh Ranjan. Balls and bins: A study in negative dependence. *Random Structures & Algorithms*, 13(2):99–124, 1998.
- [4] Pierre Dusart. The k^{th} prime is greater than $k(\ln k + \ln \ln k - 1)$ for $k \geq 2$. *Mathematics of Computation*, 68(225):411–415, January 1999.
- [5] Amos Fiat and Moni Naor. Implicit $O(1)$ probe search. *SIAM J. Comput.*, 22(1):1–10, 1993.
- [6] Amos Fiat, Moni Naor, Jeanette P. Schmidt, and Alan Siegel. Nonoblivious hashing. *Journal of the ACM*, 39(4):764–782, 1992.
- [7] Michael L. Fredman, János Komlós, and Endre Szemerédi. Storing a sparse table with $O(1)$ worst case access time. *J. Assoc. Comput. Mach.*, 31(3):538–544, 1984.

- [8] Torben Hagerup, Peter Bro Miltersen, and Rasmus Pagh. Deterministic dictionaries. *J. Algorithms*, 41(1):69–85, 2001.
- [9] Yijie Han. Deterministic sorting in $O(n \log \log n)$ time and linear space. *J. Algorithms*, 50(1):96–105, 2004.
- [10] Stasys Jukna. *Extremal Combinatorics with Applications in Computer Science*. Springer-Verlag, 2001.
- [11] Nathan Linial and Ori Sasson. Non-expansive hashing. *Combinatorica*, 18(1):121–132, 1998.
- [12] Kurt Mehlhorn. On the program size of perfect and universal hash functions. In *23th Annual Symposium on Foundations of Computer Science (FOCS 1982)*, pages 170–175, Chicago, Illinois, USA, 1982. IEEE.
- [13] Peter Bro Miltersen, 2000. Personal communication.
- [14] Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Cambridge University Press, 1995.
- [15] Noam Nisan. Extracting randomness: How and why: A survey. In *Proceedings of the 11th Annual IEEE Conference on Computational Complexity (CCC '96)*, pages 44–58. IEEE Comput. Soc. Press, 1996.
- [16] Rasmus Pagh. Low redundancy in static dictionaries with constant query time. *SIAM J. Comput.*, 31(2):353–363, 2001.
- [17] Ran Raz, Omer Reingold, and Salil Vadhan. Error reduction for extractors. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science (FOCS '99)*, pages 191–201. IEEE Comput. Soc. Press, 1999.
- [18] Ran Raz, Omer Reingold, and Salil P. Vadhan. Extracting all the randomness and reducing the error in Trevisan’s extractors. *J. Comput. Syst. Sci*, 65(1):97–128, 2002.
- [19] Milan Ružić. Constructing efficient dictionaries in close to sorting time. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP '08)*, Lecture Notes in Computer Science. Springer, 2008.
- [20] Ronen Shaltiel. Recent developments in explicit constructions of extractors. *Bulletin of the EATCS*, 77:67–95, 2002.
- [21] Amnon Ta-Shma, Christopher Umans, and David Zuckerman. Lossless condensers, unbalanced expanders, and extractors. *Combinatorica*, 27(2):213–240, 2007.
- [22] Andrew C.-C. Yao. Should tables be sorted? *J. Assoc. Comput. Mach.*, 28(3):615–628, 1981.