BRICS RS-00-36 R. Pagh: Dispersing Hash Functions

BRICS

Basic Research in Computer Science

Dispersing Hash Functions

Rasmus Pagh

BRICS Report Series **RS-00-36**

ISSN 0909-0878 **December 2000**

See back inner page for a list of recent BRICS Report Series publications. Copies may be obtained by contacting:

BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK–8000 Aarhus C
Denmark

Telephone:	**+45 8942 3360**
Telefax:	**+45 8942 3255**
Internet:	**BRICS@brics.dk**

BRICS publications are in general accessible through the World Wide Web and anonymous FTP through these URLs:

`http://www.brics.dk`
`ftp://ftp.brics.dk`
This document in subdirectory `RS/00/36/`

Dispersing Hash Functions*

Rasmus Pagh
BRICS†, Department of Computer Science, University of Aarhus,
8000 Aarhus C, Denmark
Email:pagh@brics.dk

Abstract

A new hashing primitive is introduced: dispersing hash functions. A family of hash functions F is dispersing if, for any set S of a certain size and random $h \in F$, the expected value of $|S|-|h[S]|$ is not much larger than the expectancy if h had been chosen at random from the set of all functions.

We give tight, up to a logarithmic factor, upper and lower bounds on the size of dispersing families. Such families previously studied, for example universal families, are significantly larger than the smallest dispersing families, making them less suitable for derandomization. We present several applications of dispersing families to derandomization (fast element distinctness, set inclusion, and static dictionary initialization). Also, a tight relationship between dispersing families and extractors, which may be of independent interest, is exhibited.

We also investigate the related issue of program size for hash functions which are nearly perfect. In particular, we exhibit a dramatic increase in program size for hash functions more dispersing than a random function.

1 Introduction

Universal families of hash functions [1], widely used in various areas of computer science (data structures, derandomization, cryptology), have the property, among other things, that any set S is *dispersed* by a random function from the family. More precisely, for a universal family F and any set S, if we pick a function uniformly at random from F, the expected value of $|S| - |h[S]|$ is not much larger than the expectancy if h had been chosen at random from the set of *all* functions. Another way of putting this property is that a dispersing family is good at *distinguishing* the elements of any set (the average function maps the elements to many different

*A preliminary version appeared at RANDOM 2000. Proceedings in Informatics 8, p. 53–67. © Carleton Scientific.
Section 4.3 and appendix A were added after publication, and some minor changes were made. Theorem numbers changed.

†Basic Research in Computer Science,
Centre of the Danish National Research Foundation.

values). For comparison, a universal family is good at distinguishing any *pair* of elements (few functions map them to the same value).

In section 3 we will see that hash function families much smaller than any universal family can be dispersing. In other words, dispersion is a property strictly weaker than universality. While our first upper bound is non-constructive, section 4 explores explicit construction 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 three derandomization examples given in section 5 use dispersing families in somewhat different ways than one would use universal families. We also give an example from the literature where replacing a universal family with a dispersing one immediately gives an improved result.

We will also consider a weaker form of dispersing families, where we only care about the *existence* of a single function h for which $|S|-|h[S]|$ is small. One special case of this has previously been intensely studied, namely perfect hash function families, where a function with $|h[S]| = |S|$ always exists. In section 6 we will see that the size of existentially dispersing families explodes once we require $|S|-|h[S]|$ to be smaller than that expected for a random h from the family of all functions. In other words, storing a "near-perfect" hash function is nearly as expensive as storing a perfect one.

1.1 Related work

The dispersion property of universal families was shown and first used in [6]. It has since found application in several papers [4, 9, 13].

Another formulation of the dispersion property of a family $\{h_i\}$ is that that $\mathbf{E}_i|h_i[S]|$ should be "large". The definition of a *disperser* is similar to this in that one requires $|\cup_i h_i[S]|$ 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 parameter ϵ (while we can only hope for some fraction of $|S|$). Nisan's survey [12] gives a good introduction to dispersers. It also covers the stronger notion of extractors, where the requirement is near-uniformity of the random variable $h_i(x)$, for uniformly and independently chosen h_i and $x \in S$.

Mehlhorn [10] has given tight bounds (up to a constant factor) on the number of bits needed to represent perfect and universal hash functions, i.e. determined the size of such families up to a polynomial (see also [5, 15]).

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 $(U \to R)$, and $\binom{U}{n}$ denotes the subsets of U of size n. The number of *collisions* for S under h is $C(S, h) = n - |h[S]|$. A uniform random choice is denoted by $\in_R$, and is always independent of other events. The base of "log" is 2, the base of "ln" is $e = 2.718\dots$

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 $h \in_R (U \to R)$. The probability that $i \notin h[S]$ is $(1 - \frac{1}{r})^n$ for $i \in \{0, \dots, r-1\}$. Thus the expected size of $R \backslash h[S]$ is $(1 - \frac{1}{r})^n\, r$, and the expected size of $h[S]$ is

$$\mu := r\ (1 - (1 - \tfrac{1}{r})^n) = r\ (\tbinom{n}{1}/r - \tbinom{n}{2}/r^2 + \dots) = n - \Theta(\tfrac{n^2}{r})\ . \tag{1}$$

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

$$\lambda := n - \mu = \sum_{i=1}^{n-1} \tbinom{n}{i+1}(-1/r)^i \in [\tfrac{n^2}{4r}; \tfrac{n^2}{2r}]\ . \tag{2}$$

We now turn to giving tail bounds. Let $S = \{s_1, \dots, s_n\}$ and let X_i be the 0-1 random variable which assumes the value 1 iff $h(s_i) \in \{h(s_1), \dots, h(s_{i-1})\}$. Clearly $C(S, h) = \sum_i X_i$. The random variables $X_1, \dots, X_n$ are not independent; however, they are *negatively related*:

Definition 1 *(Janson [7]) Indicator random variables $(I_i)_{i=0}^n$ are* negatively related *if for each j there exist random variables $(J_{ij})_{i=0}^n$ with distribution equal to the conditional distribution of $(I_i)_{i=0}^n$ given $I_j = 1$, and so that $J_{ij} \leq I_i$ for every i.*

The random variables Y_{ij} corresponding to the condition $X_j = 1$, $j > 1$, are defined in the same way as the X_i, except that we pick h from the set of functions satisfying $h(s_j) \in \{h(s_1), \dots, h(s_{j-1})\}$. The negative relation means that we can employ the Chernoff-style tail bounds of [7, Theorem 4]. In particular, tail bound (1.9) states that, for $c \leq 1$,

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

And tail bound (1.5) gives that, for $c \geq 1$,

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

Analogously, for a sequence $h_1, \ldots, h_b \in_R (U \to R)$, one can derive the following estimate, for $c \geq 1$:

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

3 Dispersing families

Definition 2 *A family $F \subseteq (U \to R)$, is (c,n,r,u)-*dispersing *if for any $S \subseteq U$, $|S| = n$, the expected value of $C(S,h)$ for $h \in_R F$ is at most $c\lambda$, where λ is given by (2). When parameters n, r and u follow from the context, we shall use the term c-*dispersing.

We first give a simple non-constructive 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 (the requirement on c ensures that the constant factor of the bound does not depend on c). Let $k(c) = \ln(c^c/e^{c-1}) = \Theta(c \log c)$. The family can be constructed by picking $h_1, \ldots, h_b \in_R (U \to R)$, where $b > \frac{n \ln(ue/n)}{k(c)\lambda} = O(\frac{r \log(u/n)}{n\, c \log c})$. With non-zero 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 any $S \in \binom{U}{n}$. By inequality (5),

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

Since there are less than $(ue/n)^n$ sets of size n (see e.g. [8, 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. Take any such family $F = \{h_1, \ldots, h_b\}$. We construct $U_1, \ldots, U_k$, with $U_k \subseteq U_{k-1} \subseteq \cdots \subseteq U_1 \subseteq U_0 = U$ such that $|U_i| \geq u\,(n/2r)^i$ and $|h_i[U_k]| \leq n/2$ for $i \leq k$. The set U_i is constructed from U_{i-1} using the pigeonhole principle to pick a subset with $|h_i[U_i]| \leq n/2$ of size at least $|U_{i-1}|/(2r/n)$. Setting $k = \lfloor \log(u/n)/\log(2r/n) \rfloor$ we have $|U_k| \geq n$ and can take $S \subseteq U_k$ of size n. Since F is c-dispersing we must have $\sum_i C(S,h_i) \leq b\,c\,\lambda$. On the other hand, by construction $\sum_i C(S,h_i) \geq k\,n/2$, so we must have:

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

We summarize the bounds as follows:

Theorem 3 *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:*

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

The gap between the bounds is $O(\frac{\log(2r/n)}{\log c})$. In a certain sense we have a tight bound in most cases: 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 3 convenient.

Corollary 4 *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:*

$$\frac{\log(u/n)}{2\delta \log(2r/n)} \leq |F| = O\left(\frac{\log(u/n)}{\delta \log(4\delta r/n)}\right) \quad .$$

3.1 An impossibility result

We have seen examples of c-dispersing families for $c \geq 1$. A natural question is whether such families exist for $c < 1$. This section supplies a generally negative answer for any constant $c < 1$. However, it is possible to disperse *slightly* more than a totally random function by using the family of all "evenly distributing" functions. This is analogous to universal hash functions, where it is also possible to improve marginally upon the performance of a truly random function [18].

Example 1 *Consider the case $n = r = 3$, $u = 6$, where $\lambda = 8/9$. If we pick a function at random from those mapping two elements of U to each element in the range, the expected number of collisions is $3/5$. That is, this family is $27/40$-dispersing.*

We need the following special case of a more general result shown in section 6.

Lemma 5 *Let $k(c) = \frac{(1-c)^2}{100 \log(4/(1-c))}$. Assume $r \leq k(c)\, n^2$ and $u \geq \frac{100\, r}{1-c}$, and let $h : U \to R$ be any function. For $S \in_R \binom{U}{n}$ we have $\Pr[C(S,h) \leq \frac{1+c}{2}\lambda] < 1 - \frac{2c}{1+c}$.*

Corollary 6 *For $0 < c < 1$, $r \leq k(c)\, n^2$ and $u \geq \frac{100\, r}{1-c}$, no (c, n, r, u)-dispersing family exists.*

Proof. Suppose F is a (c,n,r,u)-dispersing family with parameters satisfying the above. By an averaging argument, there must exist a function $h \in F$ such that for $S \in_R \binom{U}{n}$ the expected value of $C(S,h)$ is at most $c\lambda$. In particular, Markov's inequality gives that the probability of $C(S,h) \le \frac{1+c}{2}\lambda$ must be at least $1 - \frac{2c}{1+c}$, contradicting lemma 5. □

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 algorithm which, 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, not reached here, would be explicit families of sizes polynomially related to the bounds of theorem 3. Picking a random element from such a family uses a number of random bits which is within a constant factor of optimal (i.e., the *sample complexity* is optimal).

4.1 Universal families

Definition 7 *A family $F \subseteq (U \to R)$ is* c-universal *if for any $x, y \in U$, $x \neq y$ and $h \in_R F$, $\Pr[h(x) = h(y)] \le c/r$. It is* strongly c-universal *if for any $a, b \in R$, $\Pr[h(x) = a,\ h(y) = b] \le c/r^2$.*

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

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

where $m = 24\,r^2 \log(u)/\epsilon$. 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, and input x, we can compute $h(x)$ in polynomial time. As for universal families with parameter higher than 2, we note that taking *any* $1/c$ fraction of a 2-universal family yields a $2c$-universal family.

We now establish that universal families are dispersing, slightly generalizing an observation in [6]:

Proposition 8 *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 $C(S,h) \le K(S,h)$ 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 $\binom{n}{2}\, c/r < c\, n^2/2r \leq 2\, c\, \lambda$. $\Box$

Mehlhorn [10, Theorem D] 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 3. Hence, dispersion is a property strictly weaker than universality. The minimum sizes of c-universal and $O(c)$-dispersing families are polynomially related when $r/n \geq n^{\Omega(1)} + c^{1+\Omega(1)}$. Under the same condition, the family F_{su}, as well as a c-universal sub-family for $c > 2$, has size polynomially related to the minimum. In particular, we have explicit $O(1)$-dispersing families of optimal sample complexity for $r = n^{1+\Omega(1)}$.

4.2 Extractor based construction

This subsection addresses the construction of 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 a $O(1)$-dispersing family from an an *extractor* (see 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). We need only consider the case where r is a power of two, since such families are also $O(1)$-dispersing for ranges up to two times larger.

Definition 9 *A random variable X with values in a finite set T is* ϵ-close to uniform *if*

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

Definition 10 *A function $E : U \times \{0,1\}^s \to \{0,1\}^t$ is an* (n, ϵ)-extractor *if for any $S \in \binom{U}{n}$, the distribution of $E(x, y)$ for $x \in_R S$, $y \in_R \{0,1\}^s$ is ϵ-close to uniform over $\{0,1\}^t$.*

Non-constructive arguments show that for $t \leq \log n$ and $\epsilon > 0$ there exist (n, ϵ)-extractors with $s = O(\log(\log(u)/\epsilon))$. Much research effort is currently directed towards explicit construction of such functions (see e.g. the surveys [11, 12]).

Theorem 11 *Suppose r is a power of 2, $E : U \times \{0,1\}^s \to \{0,1\}^t$ is an $(\lfloor n/2 \rfloor, \epsilon)$-extractor, where $\epsilon = O(n/r)$, $F' \subseteq (U \to \{0,1\}^s)$ is strongly $(1+\epsilon)$-universal, and $F'' \subseteq (U \to \{0,1\}^{\log(r)-t})$ is 2-universal. Then the family*

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

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

Proof. Let $S \in \binom{U}{n}$. By the extractor property, the distribution of $E(x,z)$ for $x \in_R S$ and $z \in_R \{0,1\}^s$ 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 $i \in \{0,1\}^t \backslash B$ and $x \in_R S$ we have:

$$\Pr_{z \in_R \{0,1\}^s}[E(x,z) = i] \leq 2^{-t+1} \ . \tag{6}$$

Also note that the distribution of $E(x, f'(x))$ for $x \in_R S$ and $f' \in_R F'$ must be γ-close to uniform for $\gamma = O(\epsilon)$. We choose $f' \in_R F'$, $f'' \in_R F''$, set $h(x) = E(x, f'(x)) \circ f''(x)$, and bound the expected value of $C(S,h)$:

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

For $x \in_R S$ the first term is:

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

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

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

To bound $\Pr[E(x_1, f'(x_1)) = E(x_2, f'(x_2)) = i]$ for $i \in \{0,1\}^t \backslash B$ 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 $x_1 \in_R S_1$, $x_2 \in_R S \backslash S_1$. By symmetry, this procedure yields the same distribution. Since for any S_1, we choose x_1 and x_2 independently from disjoint sets of size at least $\lfloor n/2 \rfloor$, we have:

$$\begin{aligned}&\Pr_{f' \in_R F'}[E(x_1, f'(x_1)) = E(x_2, f'(x_2)) = i] \\ &\quad \leq (1+\epsilon) \Pr_{z_1, z_2 \in_R \{0,1\}^s}[E(x_1, z_1) = E(x_2, z_2) = i] \\ &\quad = (1+\epsilon) \Pr_{z_1 \in_R \{0,1\}^s}[E(x_1, z_1) = i] \Pr_{z_2 \in_R \{0,1\}^s}[E(x_2, z_2) = i] \\ &\quad \leq (1+\epsilon)\, (\tfrac{n}{\lfloor n/2 \rfloor}\, 2^{-t+1})^2 \ .\end{aligned} \tag{10}$$

The factor of $\frac{n}{\lceil n/2 \rceil}$ relative to (6) is due to the fact that x_1 and x_2 are sampled from a fraction $\geq \frac{\lfloor n/2 \rfloor}{n}$ of S. Plugging this into (9) we obtain:

$$\begin{aligned} &\tbinom{n}{2} \Pr[h(x_1) = h(x_2) \wedge E(x_1, f'(x_1)) \notin B \wedge E(x_2, f'(x_2)) \notin B] \\ &\quad \leq n^2\, 2^{-\log(r)+t}\, 2^t\, (1+\epsilon)\, (\tfrac{n}{\lceil n/2 \rceil}\, 2^{-t+1})^2 \\ &\quad \leq 36\, (1+\epsilon)\, n^2/r = O(n^2/r) \ . \end{aligned} \tag{11}$$

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

Corollary 12 *Given an explicit* $(\lfloor n/2 \rfloor, \epsilon)$*-extractor* $E : U \times \{0,1\}^s \to \{0,1\}^t$ *with* $\epsilon = O(n/r)$ *and* $t = \log n - O(1)$*, there is an explicit* $O(1)$*-dispersing family with sample complexity* $O(\log(r \log(u)/n) + s)$*.*

Proof. We use the construction of section 4.1 for the universal families of the above construction. The number of bits needed to sample $f' \in_R F'$ is $O(\log(2^s) + \log(\log(u)/\epsilon)) = O(s + \log(r \log(u)/n))$. The number of bits needed to sample $f'' \in F''$ is $O(\log(2^{\log(r)-t}) + \log\log u) = O(\log(r/n) + \log\log u)$. $\Box$
The best explicit extractor in current literature with the required parameters has seed length $s = O((\log\log(u\, r/n))^3)$ [17].

Of course, theorem 11 and corollary 12 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 12 to obtain an $O(1)$-dispersing family with range $\{1, \ldots, r'\}$, 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 $\Omega(n)$.

4.3 Equivalence of extractors and 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 13 *Let* $\{h_z\}_{z \in \{0,1\}^s} \subseteq (U \to \{0,1\}^t)$*,* $t = \log n - O(1)$*, be such that for any* $S \in \binom{U}{n}$ *the expected size of* $h_z[S]$ *for* $z \in_R \{0,1\}^s$ *is* $\Omega(n)$*. Then* $E(x, z) = h_z(x)$ *is an* $(n, 1 - \Omega(1))$*-extractor.*

Proof. Let $S \in \binom{U}{n}$. For $x \in_R S$ and $z \in_R \{0,1\}^s$, 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:

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

This implies that the distribution of $E(x, z)$ is $(1 - \Omega(1))$-close to uniform. $\square$

It should be noted that there is an efficient explicit way of converting an extractor with non-trivial constant error into an extractor with almost any smaller error [16]. Unfortunately, this conversion slightly weakens other parameters, so the problem of constructing optimal extractors 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 to has a special instruction which, given the parameters of a dispersing family, a "function number" i and an input x, returns $f_i(x)$, where f_i is the ith function from a dispersing family with the given parameters. The RAM also has an instruction to generate random numbers.

5.1 Element distinctness

We first consider the element distinctness problem for a list of n machine words. It is well known that in a comparison-based model this problem requires time $\Omega(n \log n)$, whereas using universal hashing (and indirect addressing) on a RAM, the problem can be solved in expected linear time, and 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 $O(\log w)$, using dispersing hash functions.

We pick a function h from a $(\frac{n/\log n}{\lambda}, n, n^2, 2^w)$-dispersing family (of size $w^{O(1)}$). By corollary 4, the expected size of $h[S]$ is $|S| - O(n/\log n)$, where S is the set of machine words considered. In time $O(n)$ we can sort the machine words according to their function values (using radix sort). Words involved in a collision are put into a balanced binary tree, each in time $O(\log n)$. Since only $O(n/\log n)$ elements (expected) can be inserted before a duplicate (if any) is found, this also takes expected time $O(n)$.

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. However, so far this has not been achieved. Possibly, a slightly stronger property than dispersion is needed.

5.2 Set inclusion

Now consider the problem of determining if the elements in a list of machine words is a subset of the elements in another list. We assume that both lists have no duplicate values, and denote the length of the longer list by n. The situation is very similar to that of element distinctness: Comparison-based algorithms require $\Omega(n \log n)$ time, and using universal hashing the time can be improved to $O(n)$, expected, using linear space. Again, the number of random bits required can be reduced to $O(\log w)$. The algorithm is a simple modification of that for element distinctness. Note that we immediately also have a test for set equality.

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 $O(n \log n)$ [14]. 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 see how to achieve another trade-off, namely expected time $O(n \log^\epsilon n)$, for any constant $\epsilon > 0$, using $O(\log w)$ random bits.

Randomized universe reduction

Picking random functions from a $(\frac{n/\log n}{\lambda}, n, n^2, 2^w)$-dispersing family of size $w^{O(1)}$, we can first reduce our problem to two subproblems: one with $O(n/\log n)$ colliding elements and one within a universe of size n^2. The first subproblem is handled using the deterministic $O(n \log n)$ algorithm. The second subproblem can be dealt with by a deterministic algorithm described below. The expected number of random bits needed to find a suitable reduction function is $O(\log w)$.

A deterministic algorithm for "small" universes

By the above, we only need to deal with the case $w \leq 2 \log n$. However, we will make the weaker assumption that $w = \log^k n$ for some constant k. Let $\epsilon > 0$ be arbitrary. We start with a $(\frac{n/\log^\epsilon n}{\lambda}, n, 2^{\log^{k-\epsilon} n}, 2^w)$-dispersing family of size $O(\log^{O(\epsilon)} n)$. Running through the functions we find in time $O(n \log^{O(\epsilon)} n)$ a function h such that $|h[S]| \geq n - n/\log^\epsilon n$ (the size of $h[S]$ can be found by sorting in time $O(n(\log \log n)^2)$, see [19]). Now choose $S_1 \subseteq S$ maximally such that h is 1-1 on S. We have reduced our problem to two subproblems: A dictionary for $S \backslash S_1$ (which has size at most $n/\log^\epsilon n$) and a dictionary for $h[S_1]$ (which is contained in a universe of size $2^{\log^{k-\epsilon} n}$). For the S_1 dictionary, we need to store the original elements as associated information, since h is not 1-1 on U. The subproblems are split in a similar way, recursively. After a constant number of steps (taking $O(n \log^{O(\epsilon)} n)$ time), each subproblem either has $O(n/\log n)$ elements or a universe of size at most $2^{\log^{1+\epsilon} n}$. All the dictionaries for small sets can be constructed in

$$\begin{tikzcd}
\binom{2^{\log^k n}}{n} \arrow[r] \arrow[d] & \binom{2^{\log^{k-\epsilon} n}}{n} \arrow[r] \arrow[d] & \cdots \arrow[r] & \binom{2^{\log^{1+\epsilon} n}}{n} \arrow[r, "\text{split}"] & \binom{2^{\log^{\epsilon} n}}{\cdot} \\
\binom{2^{\log^k n}}{n/\log^{\epsilon} n} \arrow[r] \arrow[d] & \binom{2^{\log^{k-\epsilon} n}}{n/\log^{\epsilon} n} \arrow[r] \arrow[d] & \cdots & & \\
\vdots & {} & & & \\
\binom{2^{\log^k n}}{n/\log n} & & & &
\end{tikzcd}$$

Figure 1: Overview of subproblems for the deterministic dictionary construction algorithm.

$O(n)$ time using the $O(n \log n)$ algorithm. The small universes can be split into n parts of size $2^{\log^{\epsilon} n}$. Using the $O(n \log n)$ algorithm on each part takes total time $O(n \log^{\epsilon} n)$. The "translations" using dispersing functions are sketched in figure 1.

Theorem 14 *On a RAM with an instruction set supporting dispersing families, using $O(\log w)$ random bits, expected, and space $O(n)$ we can:*

- *Solve the element distinctness problem in expected time $O(n)$.*
- *Solve the set inclusion problem in expected time $O(n)$.*
- *Construct a static dictionary in time $O(n \log^{\epsilon} n)$, for any $\epsilon > 0$.*

5.4 An implicit dictionary

As a final application, we consider the implicit dictionary scheme of Fiat et al. [4]. 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 [20]. The question considered in [4] is how little extra memory is needed to enable constant worst case lookup time. The information stored outside the table in their construction 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 $(\frac{\Omega(n)}{\lambda}, n, O(n), 2^w)$-dispersing, so we can reduce the extra memory to $O(\log w)$ bits (this result was also shown in a follow-up paper [3], using an entirely new construction).

6 Existentially dispersing families

By the result of section 3.1, we cannot expect $C(S,h) \leq \lambda/2$ (or better) when picking h at random function from some family. But there are situations in which such functions are of interest, e.g. in perfect hashing. This motivates the following weaker notion of dispersion.

Definition 15 *A family $F \subseteq (U \to R)$ is* existentially (c,n,r,u)-dispersing *if for any $S \subseteq U$, $|S| = n$, there exists $h \in F$ with $C(S,h) \leq c\,\lambda$, where λ is given by (2). When parameters n, r and u follow from the context, we shall use the term* existentially c-dispersing.

Existentially 0-dispersing families are of course better known as *perfect* families. It is well known that for $r = O(n^2/\log n)$, perfect hash functions have program size $\Theta(n^2/r + \log\log u)$ [10] (i.e., the description of a perfect hash function requires this number of bits). In this section we investigate the "nearly perfect" families between existentially 0-dispersing and existentially 1-dispersing.

6.1 A dual tail bound

We will need a tail bound "dual" to the one in (3). Specifically, h is fixed to some function, and we consider $C(S,h)$ for $S \in_R \binom{U}{n}$. We want to upper bound the probability that $C(S,h) \leq c\lambda$, for $c < 1$. First a technical lemma:

Lemma 16 *Take $0 < c < 1$ and $k \in \mathbf{N}$ with $k < (1-c)\,n^2/4r$. For any $h \in (U \to R)$, when picking $S \in_R \binom{U}{n}$ we have*

$$\Pr[C(S,h) \leq c\lambda] \leq (\tfrac{5}{2}\,n^2/ku)^k + \exp(-(1-c-4\,k\,r/n^2)^2\,n^2/8r) \ .$$

Proof. In this proof we will use the inequalities of section 2 with various values substituted for r, n and c (λ is a function of these). We use primed variables to denote the substitution values.

We consider the random experiment in which $n+k$ elements $Y_1,\ldots,Y_{n+k}$ are picked randomly and independently from U. It suffices to bound the probability of the event "$|\{Y_1,\ldots,Y_{n+k}\}| < n$ or $|h(\{Y_1,\ldots,Y_{n+k}\})| \geq n - c\,\lambda$", which occurs with greater probability than $C(S,h) \leq c\,\lambda$ (Given that $|\{Y_1,\ldots,Y_{n+k}\}| \geq n$, the sets of $\binom{U}{n}$ are contained in $\{Y_1,\ldots,Y_{n+k}\}$ with the same positive probability). The probability that $|\{Y_1,\ldots,Y_{n+k}\}| < n$ is at most $e^{-n^2/4u}(\frac{e\,(n+k)^2}{2\,k\,u})^k$, by use of (4) with parameters $r' = u$, $n' = n+k$ and $c' = k/\lambda'$. Since $k < n/4$, we have the bound $(\frac{5}{2}\,n^2/ku)^k$.

To bound the probability that $|h(\{Y_1, \ldots, Y_{n+k}\})| \geq n - c\,\lambda$, first notice that without loss of generality we can assume the function values $h(Y_i)$ to be uniformly distributed over R — this can only increase the probability. Now (3) can be used with $r' = r$, $n' = n + k$ and $c' = (c\lambda + k)/\lambda' \leq c + 4\,k\,r/n^2$: The probability of $|h(\{Y_1, \ldots, Y_{n+k}\})| \geq n - c\,\lambda$ is at most $\exp(-(1-c')^2\lambda'/2) \leq \exp(-(1 - c - 4\,k\,r/n^2)^2\, n^2/8r)$. □

We can now show the dual tail bound, which also implies lemma 5.

Lemma 17 *Suppose* $0 < c < 1$*,* $r \leq \frac{(1-c)^2}{25}\, n^2$ *and* $u \geq \frac{100}{1-c}\, r$*. For any* $h \in (U \to R)$*, when picking* $S \in_R \binom{U}{n}$ *we have*

$$\Pr[C(S,h) \leq c\lambda] \leq 2^{-\frac{(1-c)\,n^2}{20\,r}} + e^{-\frac{(1-c)^2 n^2}{25\,r}} < 1 \ .$$

In particular,

$$\Pr[C(S,h) \leq c\lambda] = 2^{-\Omega((1-c)^2 n^2/r)} \ .$$

Proof. Note that we can choose a positive integer $k \in (\frac{(1-c)\,n^2}{20\,r}; \frac{(1-c)\,n^2}{10\,r}]$ and apply lemma 16. Since $u \geq \frac{100}{1-c}\, r$, we have $(\frac{5}{2}\, n^2/ku)^k \leq 2^{-k} < 2^{-\frac{(1-c)\,n^2}{20\,r}}$. By choice of k, $(1 - c - 4k\,r/n^2)^2/8 \geq \frac{(1-c)^2}{25}$. Finally, $\frac{(1-c)^2}{25}\, n^2/r \geq 1$, so the sum is at most $2^{-1} + e^{-1} < 1$. The second inequality follows directly. □

Proof of lemma 5. Applying lemma 17, we see that

$$\Pr[C(S,h) \leq \tfrac{1+c}{2}\lambda] < 2^{-\frac{(1-c)\,n^2}{40\,r}} + e^{-\frac{(1-c)^2\,n^2}{100\,r}} \leq 2^{1-\frac{(1-c)^2\,n^2}{100\,r}} \ .$$

For this to be less than $1 - \frac{2c}{1+c} > (1-c)/2$, it suffices that $\frac{(1-c)^2 n^2}{100\,r} \geq \log(\frac{4}{1-c})$. The lemma follows by isolating r.

6.2 The program size of existentially dispersing hash functions

Given lemma 17, a lower bound on the program size of existentially c-dispersing families, for $c < 1$, follows.

Theorem 18 *For any constant* c*,* $0 < c < 1$*, there exist further constants* $k_1, k_2, k_3 > 0$ *such that for* $u \geq k_1\, r$ *and* $r \leq k_2\, n^2$*, elements of an existentially* (c, n, r, u)*-dispersing family* F *cannot be stored using less than*

$$\max(k_3\, n^2/r,\ \log\lfloor \log(u/n)/\log(2\,r/n)\rfloor) \ \textit{bits.}$$

Proof. Take $k_1 = 100/(1-c)$, $k_2 = (1-c)^2/25$. Then by lemma 17 there exists $k_3 > 0$ such that any function $h \in (U \to R)$, less than a fraction $2^{-k_3\, n^2/r}$ of $S \in \binom{U}{n}$ has $C(S,h) \leq n\, c\, \lambda$. Since for each $S \in \binom{U}{n}$ there must be a function $h \in F$ with $C(S,h) \leq c\lambda$, we must have $|F| \geq 2^{k_3\, n^2/r}$.

The lower bound of $\log\lfloor\log(u/n)/\log(2\,r/n)\rfloor$ stems from the observation that S in the lower bound proof of section 3 is constructed to have an image of size at most $n/2$ for $\lfloor\log(u/n)/\log(2\,r/n)\rfloor$ arbitrary functions. □

The bound of the theorem is within a constant factor of the upper bound on the size of perfect families for a wide range of parameters (specifically, when $r = O(n^2/\log n)$ and either $\log\log u = O(n^2/r)$ or $\log\log u = (1+\Omega(1))\log\log r$). At least in these cases, being "nearly perfect" is almost as hard as being perfect.

7 Open problems

The most obvious open problem is to find explicit dispersing families of close to minimal size. 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. The issue of constructing explicit c-dispersing families with parameter c close to r/n is interesting in view of the applications given in section 5.

There also are some things left to completely settle regarding the size of dispersing and existentially dispersing families. First of all, there still is a small gap between the bounds of theorem 3. Secondly, only quite weak lower bounds are known on the size of existentially c-dispersing families for $c > 1$. The best upper bounds are the same as for c-dispersing families, but can existentially dispersing families be much smaller?

Acknowledgments: The author would like to thank Martin Dietzfelbinger and Johan Kjeldgaard-Pedersen for helpful discussions, and Peter Bro Miltersen for valuable suggestions, including the possible use of extractors.

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 [6].

Theorem 19 *For $0 < \epsilon \leq 1$ and $m \geq 24\, r^2 \log(u)/\epsilon$ the family*

$$F_{\mathrm{su}} = \{h \mid h(x) = ((t\,x + s) \text{ mod } p) \text{ mod } r,\ m/2 \leq p \leq m,\ p \text{ prime},\ 0 \leq s, t < p\} \subseteq (U \to R)$$

is strongly $(1 + \epsilon)$-universal.

Proof. (Sketch) For any $a, b \in R$ and $x, y \in U$, where $x \neq y$, and $h \in_R F_{\mathrm{su}}$ we must show that we have $\Pr[h(x) = a \wedge h(y) = b] \leq (1 + \epsilon)/r^2$. So pick a random prime p in the range $m/2$ to m and $s, t \in_R \{0, \ldots, p - 1\}$. According to the prime number theorem, p divides $x\,y\,|x - y|$ with probability at most

$$\ln(u^3)/\ln(m/2)/(m/\ln(m)/3) < 12 \log(u)/m \leq 1/2r^2$$

using $m > 4$. With probability at most $2/m \leq 1/4r^2$, parameter t is zero. Hence, with probability at least $1 - 3\epsilon/4r^2$ we have that p does not divide $x\,y\,|x - y|$ and $t \neq 0$.

Conditioned on this, x and y are distinct nonzero numbers in $\mathbb{Z}_p$, and hence the values $t\,x + s$ and $t\,y + s$ are independent and uniformly distributed in $\mathbb{Z}_p$. This means that the probability that $h(x) = a$ and $h(y) = b$ is at most $\lceil p/r \rceil^2/p^2 \leq 1/r^2 + 2/pr + 1/p^2 \leq (1 + r/p + (r/p)^2)/r^2 < (1 + \epsilon/4)/r^2$. Summing up, the chances of $h(x) = a$ and $h(y) = b$ are at most $(1 + \epsilon)/r^2$, as desired. □

References

[1] Carter, J. L., and Wegman, M. N. Universal classes of hash functions. *J. Comput. System Sci. 18*, 2 (1979), 143–154.

[2] Dietzfelbinger, M., Gil, J., Matias, Y., and Pippenger, N. Polynomial hash functions are reliable (extended abstract). In *Proceedings of the 19th International Colloquium on Automata, Languages and Programming (ICALP '92)*, vol. 623 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1992, pp. 235–246.

[3] Fiat, A., and Naor, M. Implicit $O(1)$ probe search. *SIAM J. Comput. 22*, 1 (1993), 1–10.

[4] Fiat, A., Naor, M., Schmidt, J. P., and Siegel, A. Nonoblivious hashing. *Journal of the ACM 39*, 4 (1992), 764–782.

[5] Fredman, M. L., and Komlós, J. On the size of separating systems and families of perfect hash functions. *SIAM Journal on Algebraic and Discrete Methods 5*, 1 (1984), 61–68.

[6] Fredman, M. L., Komlós, J., and Szemerédi, E. Storing a sparse table with $O(1)$ worst case access time. *J. Assoc. Comput. Mach. 31*, 3 (1984), 538–544.

[7] Janson, S. Large deviation inequalities for sums of indicator variables. Tech. Rep. 34, Department of Mathematics, Uppsala University, 1993.

[8] Jukna, S. *Extremal Combinatorics with Applications in Computer Science*. Springer-Verlag, Berlin, 2000.

[9] Linial, N., and Sasson, O. Non-expansive hashing. *Combinatorica 18*, 1 (1998), 121–132.

[10] Mehlhorn, K. *Data structures and algorithms. 1, Sorting and searching.* Springer-Verlag, Berlin, 1984.

[11] Miltersen, P. B. Derandomizing complexity classes. Manuscript, to appear in Handbook of Randomized Computation, 2000.

[12] Nisan, N. Extracting randomness: How and why: A survey. In *Proceedings of the 11th Annual IEEE Conference on Computational Complexity (CCC '96).* IEEE Comput. Soc. Press, Los Alamitos, CA, 1996, pp. 44–58.

[13] Pagh, R. Low Redundancy in Static Dictionaries with $O(1)$ Lookup Time. In *Proceedings of the 26th International Colloquium on Automata, Languages and Programming (ICALP '99)*, vol. 1644 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1999, pp. 595–604.

[14] Pagh, R. Faster Deterministic Dictionaries. In *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '00).* ACM Press, New York, 2000, pp. 487–493.

[15] Radhakrishnan, J. Improved bounds for covering complete uniform hypergraphs. *Inform. Process. Lett. 41*, 4 (1992), 203–207.

[16] Raz, R., Reingold, O., and Vadhan, S. Error reduction for extractors. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science (FOCS '99)* (Los Alamitos, CA, 1999), IEEE Comput. Soc. Press, pp. 191–201.

[17] Raz, R., Reingold, O., and Vadhan, S. Extracting all the randomness and reducing the error in trevisan's extractors. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing (STOC '99).* ACM Press, 1999, pp. 149–158.

[18] Sarwate, D. V. A note on: "Universal classes of hash functions" [J. Comput. System Sci. **18** (1979), no. 2, 143–154; MR 80f:68110a] by J. L. Carter and M. N. Wegman. *Inform. Process. Lett. 10*, 1 (1980), 41–45.

[19] THORUP, M. Faster deterministic sorting and priority queues in linear space. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '98)*. ACM Press, New York, 1998, pp. 550–555.

[20] YAO, A. C.-C. Should tables be sorted? *J. Assoc. Comput. Mach. 28*, 3 (1981), 615–628.

Recent BRICS Report Series Publications

RS-00-36 Rasmus Pagh. *Dispersing Hash Functions*. December 2000. 18 pp. Preliminary version appeared in *4th. International Workshop on Randomization and Approximation Techniques in Computer Science*, RANDOM '00, Proceedings in Informatics 8, 2000, pages 53–67.

RS-00-35 Olivier Danvy and Lasse R. Nielsen. *CPS Transformation of Beta-Redexes*. December 2000. 12 pp.

RS-00-34 Olivier Danvy and Morten Rhiger. *A Simple Take on Typed Abstract Syntax in Haskell-like Languages*. December 2000. 25 pp. To appear in *Fifth International Symposium on Functional and Logic Programming*, FLOPS '01 Proceedings, LNCS, 2001.

RS-00-33 Olivier Danvy and Lasse R. Nielsen. *A Higher-Order Colon Translation*. December 2000. 17 pp. To appear in *Fifth International Symposium on Functional and Logic Programming*, FLOPS '01 Proceedings, LNCS, 2001.

RS-00-32 John C. Reynolds. *The Meaning of Types — From Intrinsic to Extrinsic Semantics*. December 2000. 35 pp.

RS-00-31 Bernd Grobauer and Julia L. Lawall. *Partial Evaluation of Pattern Matching in Strings, revisited*. November 2000. 48 pp.

RS-00-30 Ivan B. Damgård and Maciej Koprowski. *Practical Threshold RSA Signatures Without a Trusted Dealer*. November 2000. 14 pp.

RS-00-29 Luigi Santocanale. *The Alternation Hierarchy for the Theory of μ-lattices*. November 2000. 44 pp. Extended abstract appears in *Abstracts from the International Summer Conference in Category Theory*, CT2000, Como, Italy, July 16–22, 2000.

RS-00-28 Luigi Santocanale. *Free μ-lattices*. November 2000. 51 pp. Short abstract appeared in *Proceedings of Category Theory 99*, Coimbra, Portugal, July 19–24, 1999. Full version to appear in a special conference issue of the *Journal of Pure and Applied Algebra*.

RS-00-27 Zoltán Ésik and Werner Kuich. *Inductive -Semirings*. October 2000. 34 pp.