Balls and Bins: Smaller Hash Families and Faster Evaluation

Article *in* Foundations of Computer Science, 1975., 16th Annual Symposium on · October 2011

DOI: 10.1109/FOCS.2011.49 · Source: DBLP

Udi Wieder
Microsoft
38 PUBLICATIONS **2,254** CITATIONS

Balls and Bins: Smaller Hash Families and Faster Evaluation

L. Elisa Celis* Omer Reingold† Gil Segev† Udi Wieder†

March 27, 2012

Abstract

A fundamental fact in the analysis of randomized algorithms is that when n balls are hashed into n bins independently and uniformly at random, with high probability each bin contains at most $O(\log n/\log\log n)$ balls. In various applications, however, the assumption that a truly random hash function is available is not always valid, and explicit functions are required.

In this paper we study the size of families (or, equivalently, the description length of their functions) that guarantee a maximal load of $O(\log n/\log\log n)$ with high probability, as well as the evaluation time of their functions. Whereas such functions must be described using $\Omega(\log n)$ bits, the best upper bound was formerly $O(\log^2 n/\log\log n)$ bits, which is attained by $O(\log n/\log\log n)$-wise independent functions. Traditional constructions of the latter offer an evaluation time of $O(\log n/\log\log n)$, which according to Siegel's lower bound [FOCS '89] can be reduced only at the cost of significantly increasing the description length.

We construct two families that guarantee a maximal load of $O(\log n/\log\log n)$ with high probability. Our constructions are based on two different approaches, and exhibit different trade-offs between the description length and the evaluation time. The first construction shows that $O(\log n/\log\log n)$-wise independence can in fact be replaced by "gradually increasing independence", resulting in functions that are described using $O(\log n\log\log n)$ bits and evaluated in time $O(\log n\log\log n)$. The second construction is based on derandomization techniques for space-bounded computations combined with a tailored construction of a pseudorandom generator, resulting in functions that are described using $O(\log^{3/2} n)$ bits and evaluated in time $O(\sqrt{\log n})$. The latter can be compared to Siegel's lower bound stating that $O(\log n/\log\log n)$-wise independent functions that are evaluated in time $O(\sqrt{\log n})$ must be described using $\Omega(2^{\sqrt{\log n}})$ bits.

A preliminary version of this work appeared in *Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 599-608, 2011.

*University of Washington, Seattle, WA 98195, USA. Email: `ecelis@cs.washington.edu`. Research supported by a UW Engineering Graduate Fellowship. Part of this work was completed while visiting Microsoft Research Silicon Valley.

†Microsoft Research Silicon Valley, Mountain View, CA 94043, USA. Email: `{omer.reingold,gil.segev, uwieder}@microsoft.com`.

1 Introduction

Traditional analysis of randomized algorithms and data structures often assumes the availability of a truly random function, whose description length and evaluation time are not taken into account as part of the overall performance. In various applications, however, such an assumption is not always valid and explicit constructions are required. This motivated a well-studied line of research aiming at designing explicit and rather small families of functions, dating back more than 30 years to the seminal work of Carter and Wegman [CW79].

In this paper we study explicit constructions of families for the classical setting of hashing n balls into n bins. A well-known and useful fact is that when n balls are hashed into n bins independently and uniformly at random, with high probability each bin contains at most $O(\log n/\log\log n)$ balls. Thus, a natural problem is to construct explicit and significantly smaller families of functions that offer the same maximal load guarantee. More specifically, we are interested in families $\mathcal{H}$ of functions that map a universe U into the set $\{1,\ldots,n\}$, such that for any set $S \subseteq U$ of size n a randomly chosen function $h \in \mathcal{H}$ guarantees a maximal load of $O(\log n/\log\log n)$ with high probability. The main measures of efficiency for such families are the description length and evaluation time of their functions.

It is well-known that any family of $O(\log n/\log\log n)$-wise independent functions guarantees a maximal load of $O(\log n/\log\log n)$ with high probability, and this already yields a significant improvement over a truly random function. Specifically, such functions can by represented by $O(\log^2 n/\log\log n)$ bits, instead of $O(|U|\log n)$ bits for a truly random function[1]. A natural approach for reducing the description length is to rely on k-wise independence for $k = o(\log n/\log\log n)$, but so far no progress has been made in this direction (even though to the best of our knowledge an explicit lower bound is only known for $k = 2$ [ADM$^+$99]). At the same time, a standard application of the probabilistic method shows that there exists such a family where each function is described by only $O(\log n)$ bits, which is in fact optimal. This naturally leads to the following open problem (whose variants were posed explicitly by Alon et al. [ADM$^+$99] and by Pagh et al. [PPR07]):

Problem 1: Construct an explicit family that guarantees a maximal load of $O(\log n/\log\log n)$ with high probability, in which each function is described by $o(\log^2 n/\log\log n)$ bits, or even $O(\log n)$ bits.

In terms of the evaluation time, an $O(\log n/\log\log n)$-wise independent function can be evaluated using traditional constructions in time $O(\log n/\log\log n)$. A lower bound proved by Siegel [Sie04] shows that the latter can be reduced only at the cost of significantly increasing the description length. For example, even for $k = O(\log n/\log\log n)$ a constant evaluation time requires polynomial space. In the same work Siegel showed a tight (but rather impractical) upper bound matching his lower bound. Subsequent constructions improved the constants involved considerably (see Section 1.2 for a more elaborated discussion), but all of these constructions suffer from descriptions of length at least n^ϵ bits for a small constant $\epsilon > 0$. This leads to the following open problem:

Problem 2: Construct an explicit family that guarantees a maximal load of $O(\log n/\log\log n)$ with high probability, in which each function is evaluated in time $o(\log n/\log\log n)$ and represented by $n^{o(1)}$ bits.

[1]For simplicity we assume that the universe size is polynomial in n, as otherwise one can reduce the size of the universe using a pair-wise independent function (that is described using $O(\log|U|)$ bits and evaluated in constant time).

1.1 Our Contributions

We present two constructions of hash families that guarantee a maximal load of $O(\log n/\log\log n)$ when hashing n elements into n bins with all but an arbitrary polynomially-small probability. These are the first explicit constructions in which each function is described using less than $O(\log^2 n/\log\log n)$ bits. Our constructions offer different trade-offs between the description length of the functions and their evaluation time. Table 1 summarizes the parameters of our constructions and of the previously known constructions.

Construction 1: gradually-increasing independence. In our first construction each function is described using $O(\log n \log\log n)$ bits and evaluated in time $O(\log n \log\log n)$. Whereas $O(\log n/\log\log n)$-wise independence suffices for a maximal load of $O(\log n/\log\log n)$, the main idea underlying our construction is that *the entire output* need not be $O(\log n/\log\log n)$-wise independent.

Our construction is based on concatenating the outputs of $O(\log\log n)$ functions which are *gradually* more independent: each function f in our construction is described using d functions $h_1, \ldots, h_d$, and for any $x \in [u]$ we define

$$f(x) = h_1(x) \circ \cdots \circ h_d(x) \ ,$$

where we view the output of each h_i as a binary string, and $\circ$ denotes the concatenation operator on binary strings. The first function h_1 is only $O(1)$-wise independent, and the level of independence gradually increases to $O(\log n/\log\log n)$-wise independence for the last function h_d. As we increase the level of independence, we decrease the output length of the functions from $\Omega(\log n)$ bits for h_1 to $O(\log\log n)$ bits for h_d. We instantiate these $O(\log\log n)$ functions using ϵ-biased distributions. The trade-off between the level of independence and the output length implies that each of these functions can be described using only $O(\log n)$ bits and evaluated in time $O(\log n)$.

Construction 2: derandomizing space-bounded computations. In our second construction each function is described using $O(\log^{3/2} n)$ bits and evaluated in time $O(\log^{1/2} n)$. Each function f in our construction is described using a function h that is $O(1)$-wise independent, and $\ell = O(2^{\log^{1/2} n})$ functions $g_1, \ldots, g_\ell$ that are $O(\log^{1/2} n)$-wise independent, and for any $x \in [u]$ we define

$$f(x) = g_{h(x)}(x) \ .$$

Naively, the description length of such a function f is $O(\ell \cdot \log^{3/2} n)$ bits, and the main idea underlying our approach is that instead of sampling the functions $g_1, \ldots, g_\ell$ independently and uniformly at random, they can be obtained as the output of an explicit pseudorandom generator for space-bounded computations using a seed of length $O(\log^{3/2} n)$ bits. Moreover, we present a new construction of a pseudorandom generator for space-bounded computations in which the description of each of these ℓ functions can be computed in time $O(\log^{1/2} n)$ without increasing the length of the seed.

Our generator is obtained as a composition of those constructed by Nisan [Nis92] and by Nisan and Zuckerman [NZ96] together with an appropriate construction of a randomness extractor for instantiating the Nisan-Zuckerman generator. The evaluation time of our second construction can be compared to Siegel's lower bound [Sie04] stating that $O(\log n/\log\log n)$-wise independent functions that are evaluated in time $O(\log^{1/2} n)$ must be described using $\Omega(2^{\log^{1/2} n})$ bits.

We note that a generator with an optimal seed length against space-bounded computations will directly yield a hash family with the optimal description length $O(\log n)$ bits. Unfortunately, the best known generator [Nis92] essentially does not give any improvement over using

$O(\log n/\log\log n)$-wise independence. Instead, our above-mentioned approach is based on techniques that were developed in the area of pseudorandomness for space-bounded computations which we show how to use for obtaining an improvement in our specific setting. Specifically, our construction is inspired by the pseudorandom generator constructed by Lu [Lu02] for the simpler class of combinatorial rectangles.

Extensions. It is possible to show that the hash families constructed in this paper can be successfully employed for storing elements using linear probing. In this setting our constructions guarantee an insertion time of $O(\log n)$ with high probability when storing $(1-\alpha)n$ elements in a table of size n, for any constant $0 < \alpha < 1$ (and have constant expected insertion time as follows from [PPR07]). Prior to our work, constructions that offered such a high probability bound had either description length of $\Omega(\log^2 n)$ bits with $\Omega(\log n)$ evaluation time (using $O(\log n)$-wise independence [SS90]) or description length of $\Omega(n^\epsilon)$ bits with constant evaluation time [Sie04, PT11].

In addition, we note that our constructions can easily be augmented to offer $O(\log\log n)$-wise independence (for the first construction), and $O(\log^{1/2} n)$-wise independence (for the second construction) without affecting their description length and evaluation time. This may be useful, for example, in any application that involves tail bounds for limited independence.

Lower bounds. We accompany our constructions with formal proofs of two somewhat folklore lower bounds. First, we show that for a universe of size at least n^2, any family of functions has a maximal load of $\Omega(\log n/\log\log n)$ with high probability. Second, we show that the functions of any family that guarantees a maximal load of $O(\log n/\log\log n)$ with probability $1-\epsilon$ must be described by $\Omega(\log n + \log(1/\epsilon))$ bits.

	Description length (bits)	**Evaluation time**
Simulating full independence ([DW03, PP08])	$O(n\log n)$	$O(1)$
[Sie04],[DMadH90],[PT11]	n^ϵ (for constant $\epsilon < 1$)	$O(1)$
$O\left(\frac{\log n}{\log\log n}\right)$-wise independence (polynomials)	$O\left(\frac{\log^2 n}{\log\log n}\right)$	$O\left(\frac{\log n}{\log\log n}\right)$
This paper (Section 4)	$O\left(\log^{3/2} n\right)$	$O\left(\log^{1/2} n\right)$
This paper (Section 3)	$O(\log n\log\log n)$	$O(\log n\log\log n)$

Table 1: The description length and evaluation time of our constructions and of the previously known constructions that guarantee a maximal load of $O(\log n/\log\log n)$ with high probability (sorted in decreasing order of description lengths).

1.2 Related Work

As previously mentioned, a truly random function guarantees a maximal load of $O(\log n/\log\log n)$ with high probability, but must be described by $\Omega(u\log n)$ bits. Pagh and Pagh [PP08] and Dietzfelbinger and Woelfel [DW03], in a rather surprising and useful result, showed that it is possible to simulate full independence for any specific set of size n (with high probability) using only $O(n\log n)$ bits and constant evaluation time. A different and arguably simpler construction was later proposed by Dietzfelbinger and Rink [DR09].

In an influential work, Siegel [Sie04] showed that for a small enough constant $\epsilon > 0$ it is possible to construct a family of functions where there is a small probability of error, but if error is avoided then the family is n^{ϵ}-wise independent, and each function is described using $n^{\epsilon'}$ bits (where $\epsilon < \epsilon' < 1$). More importantly, a function is evaluated in constant time. While this construction has attractive asymptotic behavior it seems somewhat impractical, and was improved by Dietzfelbinger and Rink [DR09] who proposed a more practical construction (offering the same parameters). Siegel [Sie04] also proved a cell probe time-space tradeoff for computing almost k-wise independent functions. Assuming that in one time unit we can read a word of $\log n$ bits, denote by Z the number of words in the representation of the function and by T the number of probes to the representation. Siegel showed that when computing a k-wise δ-dependent function into $[n]$ then either $T \geq k$ or $Z \geq n^{\frac{1}{T}}(1-\delta)$. Observe that if k is a constant, setting $T \leq k-1$ already implies the space is polynomial in n. Also, computing a $O(\log n)$-wise independent function in time $O(\sqrt{\log n})$ requires the space to be roughly $O(2^{\sqrt{\log n}})$.

Few constructions diverged from the large k-wise independence approach. In [ADM$^+$99] it is shown that matrix multiplication over $\mathbb{Z}_2$ yields a maximal load of $O(\log n \log\log n)$ with high probability, where each function is described using $O(\log^2 n)$ bits and evaluated in time $O(\log n)$. Note that this family is only pairwise independent. The family of functions described in [DMadH90] (which is $O(1)$-wise independent) yields a maximal load of $O(\log n/\log\log n)$ with high probability, where each function is described using n^{ϵ} bits and evaluated in constant time (similar to [Sie04]). The main advantage of this family is its practicality: it is very simple and the constants involved are small.

Recently, Pătraşcu and Thorup [PT11] showed a another practical and simple construction that uses n^{ϵ} space and $O(1)$ time and can replace truly random functions in various applications, although it is only 3-wise independent. A different approach was suggested by Mitzenmacher and Vadhan [MV08], who showed that in many cases a pair-wise independent function suffices, provided the hashed elements themselves have a certain amount of entropy.

1.3 Paper Organization

The reminder of this paper is organized as follows. In Section 2 we present several basic notions, definitions, and tools that are used in our constructions. In Sections 3 and 4 we present our first and second constructions, respectively. In Section 5 we prove two lower bounds for hash families, and in Section 6 we discuss several extensions and open problems.

2 Preliminaries and Tools

In this section we present the relevant definitions and background as well as the existing tools used in our constructions.

2.1 Basic Definitions and the Computational Model

Throughout this paper, we consider log to be of base 2. For an integer $n \in \mathbb{N}$ we denote by $[n]$ the set $\{1,\dots,n\}$, and by U_n the uniform distribution over the set $\{0,1\}^n$. For a random variable X we denote by $x \leftarrow X$ the process of sampling a value x according to the distribution of X. Similarly, for a finite set S we denote by $x \leftarrow S$ the process of sampling a value x according to the uniform distribution over S. The *statistical distance* between two random variables X and Y over a finite domain Ω is $\mathrm{SD}(X,Y) = \frac{1}{2}\sum_{\omega\in\Omega} |\Pr[X=\omega] - \Pr[Y=\omega]|$. For two bit-strings x and y we denote by $x \circ y$ their concatenation.

We consider the *unit cost RAM model* in which the elements are taken from a universe of size u, and each element can be stored in a single word of length $w = O(\log u)$ bits. Any operation in the standard instruction set can be executed in constant time on w-bit operands. This includes addition, subtraction, bitwise Boolean operations, parity, left and right bit shifts by an arbitrary number of positions, and multiplication. The unit cost RAM model has been the subject of much research, and is considered the standard model for analyzing the efficiency of data structures and hashing schemes (see, for example, [DP08, Hag98, HMP01, Mil99, PP08] and the references therein).

2.2 Random Variables and Functions with Limited Independence

A family $\mathcal{F}$ of functions $f : [u] \to [v]$ is *k-wise δ-dependent* if for any distinct $x_1, \ldots, x_k \in [u]$ the statistical distance between the distribution $(f(x_1), \ldots, f(x_k))$ where $f \leftarrow \mathcal{F}$ and the uniform distribution over $[v]^k$ is at most δ. A simple example for k-wise independent functions (with $\delta = 0$) is the family of all polynomials of degree $k-1$ over a finite field. Each such polynomial can be represented using $O(k \max\{\log u, \log v\})$ bits and evaluated in time $O(k)$ in the unit RAM model assuming that a field element fits into a constant number of words.

For our constructions we require functions that have a more succinct representation, and still enjoy a fast evaluation. For this purpose we implement k-wise δ-dependent functions using ϵ-biased distributions [NN93]. A sequence of random variables $X_1, \ldots, X_n$ over $\{0,1\}$ is *ϵ-biased* if for any non-empty set $S \subseteq [n]$ it holds that $|\Pr[\oplus_{i \in S} X_i = 1] - \Pr[\oplus_{i \in S} X_i = 0]| \leq \epsilon$, where $\oplus$ is the exclusive-or operator on bits. Alon et al. [AGH$^+$92, Sec. 5] constructed an ϵ-biased distribution over $\{0,1\}^n$ where each point $x \in \{0,1\}^n$ in the sample space can be specified using $O(\log(n/\epsilon))$ bits, and each individual bit of x can be computed in time $O(\log(n/\epsilon))$. Moreover, in the unit cost RAM model with a word size of $w = \Omega(\log(n/\epsilon))$ bits, each block of $t \in [n]$ consecutive bits can be computed in time $O(\log(n/\epsilon) + t)$.[2]

Using the fact that for any k, an ϵ-biased distribution is also k-wise δ-dependent for $\delta = \epsilon 2^{k/2}$ (see, for example, [AGH$^+$92, Cor. 1]), we obtain the following corollary:

Corollary 2.1. *For any integers u and v such that v is a power of 2, there exists a family of k-wise δ-dependent functions $f : [u] \to [v]$ where each function can be described using $O(\log u + k \log v + \log(1/\delta))$ bits. Moreover, in the unit cost RAM model with a word size of $w = \Omega(\log u + k \log v + \log(1/\delta))$ bits each function can be evaluated in time $O(\log u + k \log v + \log(1/\delta))$.*

The construction is obtained from the ϵ-biased distribution of Alon et al. over $n = u \log v$ bits with $\epsilon = \delta 2^{-k \log v/2}$. One partitions the $u \log v$ bits into u consecutive blocks of $\log v$ bits, each of which represents a single output value in the set $[v]$.

A useful tail bound for limited independence. The following is a natural generalization of a well-known tail bound for $2k$-wise independent random variables [BR94, Lemma 2.2] (see also [DP09]) to random variables that are $2k$-wise δ-dependent.

Lemma 2.2. *Let $X_1, \ldots, X_n \in \{0,1\}$ be $2k$-wise δ-dependent random variables, for some $k \in \mathbb{N}$ and $0 \leq \delta < 1$, and let $X = \sum_{i=1}^n X_i$ and $\mu = \mathrm{E}[X]$. Then, for any $t > 0$ it holds that*

$$\Pr[|X - \mu| > t] \leq 2\left(\frac{2nk}{t^2}\right)^k + \delta\left(\frac{n}{t}\right)^{2k}.$$

[2] Specifically, the seed consists of two elements $x, y \in \mathrm{GF}[2^m]$, where $m = O(\log(n/\epsilon))$, and the ith output bit is the inner product modulo 2 of the binary representations of x^i and y. Here we make the assumption that multiplication over $\mathrm{GF}[2^m]$ and inner product modulo 2 of m-bit strings can be done in constant time.

Proof. Using Markov's inequality we obtain that

$$\begin{aligned}
\Pr\left[|X-\mu|>t\right] &\leq \Pr\left[(X-\mu)^{2k}>t^{2k}\right] \\
&\leq \frac{\mathrm{E}\left[(X-\mu)^{2k}\right]}{t^{2k}} \\
&= \frac{\sum_{i_1,\ldots,i_{2k}\in[n]}\mathrm{E}\left[\prod_{j=1}^{2k}(X_{i_j}-\mu_{i_j})\right]}{t^{2k}} \\
&\leq \frac{\sum_{i_1,\ldots,i_{2k}\in[n]}\mathrm{E}\left[\prod_{j=1}^{2k}(\hat{X}_{i_j}-\mu_{i_j})\right]+\delta n^{2k}}{t^{2k}} \\
&= \frac{\mathrm{E}\left[(\hat{X}-\mu)^{2k}\right]}{t^{2k}}+\delta\left(\frac{n}{t}\right)^{2k} ,
\end{aligned}$$

where $\mu_i = \mathrm{E}\left[X_i\right]$ for every $i \in [n]$, $\hat{X}_1,\ldots,\hat{X}_n$ are independent random variables having the same marginal distributions as $X_1,\ldots,X_n$, and $\hat{X}=\sum_{i=1}^n \hat{X}_i$. In addition, as in [BR94, Lemma 2.2], it holds that

$$\frac{\mathrm{E}\left[(\hat{X}-\mu)^{2k}\right]}{t^{2k}} \leq 2\left(\frac{2nk}{t^2}\right)^k .$$

This implies that

$$\Pr\left[|X-\mu|>t\right] \leq 2\left(\frac{2nk}{t^2}\right)^k+\delta\left(\frac{n}{t}\right)^{2k} .$$

■

2.3 Randomness Extraction

The *min-entropy* of a random variable X is $\mathrm{H}_\infty(X) = -\log(\max_x \Pr\left[X=x\right])$. A *$k$-source* is a random variable X with $\mathrm{H}_\infty(X)\geq k$. A *$(T,k)$-block source* is a random variable $X=(X_1,\ldots,X_T)$ where for every $i\in[T]$ and $x_1,\ldots,x_{i-1}$ it holds that $\mathrm{H}_\infty(X_i|X_1=x_1,\ldots,X_{i-1}=x_{i-1})\geq k$. In our setting we find it convenient to rely on the following natural generalization of block sources:

Definition 2.3. *A random variable $X=(X_1,\ldots,X_T)$ is a (T,k,ϵ)-block source if for every $i\in[T]$ it holds that*

$$\Pr_{(x_1,\ldots,x_{i-1})\leftarrow(X_1,\ldots,X_{i-1})}\left[\mathrm{H}_\infty(X_i|X_1=x_1,\ldots,X_{i-1}=x_{i-1})\geq k\right]\geq 1-\epsilon .$$

The following lemma and corollary show that any source with high min-entropy can be viewed as a (T,k,ϵ)-block source.

Lemma 2.4 ([GW97])**.** *Let X_1 and X_2 be random variables over $\{0,1\}^{n_1}$ and $\{0,1\}^{n_2}$, respectively, such that $\mathrm{H}_\infty(X_1X_2)\geq n_1+n_2-\Delta$. Then, $\mathrm{H}_\infty(X_1)\geq n_1-\Delta$, and for any $\epsilon>0$ it holds that*

$$\Pr_{x_1\leftarrow X_1}\left[\mathrm{H}_\infty(X_2|X_1=x_1)<n_2-\Delta-\log(1/\epsilon)\right]<\epsilon .$$

Corollary 2.5. *Any random variable $X=(X_1,\ldots,X_T)$ over $(\{0,1\}^n)^T$ with $\mathrm{H}_\infty(X)\geq Tn-\Delta$ is a $(T,n-\Delta-\log(1/\epsilon),\epsilon)$-block source for any $\epsilon>0$.*

The following defines the notion of a strong randomness extractor.

Definition 2.6. *A function* $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *is a* strong (k, ϵ)-extractor *if for any k-source X over $\{0,1\}^n$ it holds that*

$$\mathrm{SD}\left((S, \mathsf{Ext}(X, S)), (S, Y)\right) \leq \epsilon \ ,$$

where S and Y are independently and uniformly distributed over $\{0,1\}^d$ and $\{0,1\}^m$, respectively.

For our application we rely on the generalization of the leftover hash lemma to block sources [CG88, ILL89, Zuc96], showing that a strong extractor enables to reuse the same seed for a block source. This generalization naturally extends to (T, k, ϵ)-block sources:

Lemma 2.7. *Let $X = (X_1, \ldots, X_T)$ be a (T, k, ϵ)-block source over $\{0,1\}^n$ and let $\mathcal{H}$ be a family of pairwise independent functions $h : \{0,1\}^n \to \{0,1\}^m$, where $m \leq k - 2\log(1/\epsilon)$. Then,*

$$\mathrm{SD}\left((h, h(X_1), \ldots, h(X_T)), (h, Y_1, \ldots, Y_T)\right) \leq 2T\epsilon \ ,$$

where $h \leftarrow \mathcal{H}$, and $(Y_1, \ldots, Y_T)$ is independently and uniformly distributed over $\left(\{0,1\}^m\right)^T$.

2.4 Pseudorandom Generators for Space-Bounded Computations

In this paper we model space-bounded computations as layered branching programs (LBP)[3]. An (s, v, ℓ)-LBP is a directed graph with $2^s(\ell+1)$ vertices that are partitioned into $\ell+1$ layers with 2^s vertices in each layer. For every $i \in \{0, \ldots, \ell-1\}$ each vertex in layer i has 2^v outgoing edges to vertices in layer $i+1$, one edge for every possible string $x_i \in \{0,1\}^v$. In addition, layer 0 contains a designated initial vertex, and each vertex in layer ℓ is labeled with 0 or 1. For an (s, v, ℓ)-LBP M and an input $x = (x_1, \ldots, x_\ell) \in \left(\{0,1\}^v\right)^\ell$, the computation $M(x)$ is defined by a walk on the graph corresponding to M, starting from the initial vertex in layer 0, and each time advancing to level i along the edge labeled by x_i. The value $M(x)$ is the label of the vertex that is reached in the last layer.

We now define the notion of a pseudorandom generator that ϵ-fools a branching program M. Informally, this means that M can distinguish between the uniform distribution and the output of the generator with probability at most ϵ.

Definition 2.8. *A generator $G : \{0,1\}^m \to \left(\{0,1\}^v\right)^\ell$ is said to ϵ-fool a layered branching program M if*

$$\left|\Pr\left[M(G(x)) = 1\right] - \Pr\left[M(y) = 1\right]\right| \leq \epsilon \ ,$$

where $x \leftarrow \{0,1\}^m$ and $y \leftarrow \left(\{0,1\}^v\right)^\ell$.

Theorem 2.9 ([Nis92, INW94]). *For any s, v, ℓ and ϵ there exists a generator $G : \{0,1\}^m \to \left(\{0,1\}^v\right)^\ell$ that ϵ-fools any (s, v, ℓ)-LBP, where $m = O(v + \log \ell (s + \log \ell + \log(1/\epsilon)))$, and for any seed $x \in \{0,1\}^m$ the value $G(x)$ can be computed in time* $\mathrm{poly}(s, v, \ell, \log(1/\epsilon))$.

3 A Construction Based on Gradually-Increasing Independence

In this section we present our first family of functions which, as discussed in Section 1.1, is based on replacing $O(\log n / \log\log n)$-wise independence with "gradually-increasing independence". The construction is obtained by concatenating the outputs of $O(\log\log n)$ functions which are *gradually*

[3]In our setting we are interested in layered branching programs that count the number of balls that are mapped to a specific bin (or, more generally, to a specific subset of the bins).

more independent. Each function $f \in \mathcal{F}$ in our construction consists of d functions $h_1, \ldots, h_d$ that are sampled independently, and for any $x \in [u]$ we define

$$f(x) = h_1(x) \circ \cdots \circ h_d(x) \ ,$$

where we view the output of each h_i as a binary string, and $\circ$ denotes the concatenation operator on binary strings[4]. Going from left to right each function in the above concatenation has a higher level of independence (from $O(1)$-wise almost independence for h_1, to $O(\log n / \log\log n)$-wise almost independence for h_d), and a shorter output length (from $\Omega(\log n)$ bits for h_1, to $O(\log\log n)$ bits for h_d). This trade-off enables us to represent each of these d functions using $O(\log n)$ bits and to evaluate it in time $O(\log n)$. This constructions allows us to prove the following theorem:

Theorem 3.1. *For any constant $c > 0$ and integers n and $u = \mathrm{poly}(n)$ there exists a family $\mathcal{F}$ of functions $f : [u] \to [n]$ satisfying the following properties:*

1. *Each $f \in \mathcal{F}$ is described using $O(\log n \log\log n)$ bits.*
2. *For any $f \in \mathcal{F}$ and $x \in [u]$ the value $f(x)$ can be computed in time $O(\log n \log\log n)$ in the unit cost RAM model.*
3. *There exists a constant $\gamma > 0$ such that for any set $S \subseteq [u]$ of size n it holds that*

$$\Pr_{f \leftarrow \mathcal{F}} \left[\max_{i \in [n]} \left| f^{-1}(i) \cap S \right| \leq \frac{\gamma \log n}{\log\log n} \right] > 1 - \frac{1}{n^c} \ .$$

In what follows we provide a more formal description of our construction (see Section 3.1), and then analyze it for proving Theorem 3.1 (see Section 3.2).

3.1 A Formal Description

We assume that n is a power of two, as otherwise we can choose the number of bins to be the largest power of two which is smaller than n, and this may affect the maximal load by at most a multiplicative factor of two. Let $d = O(\log\log n)$, and for every $i \in [d]$ let $\mathcal{H}_i$ be a family of k_i-wise δ-dependent functions $h_i : [u] \to \{0,1\}^{\ell_i}$, where:

- $n_0 = n$, and $n_i = n_{i-1}/2^{\ell_i}$ for every $i \in [d]$.
- $\ell_i = \lfloor (\log n_{i-1})/4 \rfloor$ for every $i \in [d-1]$, and $\ell_d = \log n - \sum_{i=1}^{d-1} \ell_i$.
- $k_i \ell_i = \Theta(\log n)$ for every $i \in [d-1]$, and $k_d = \Theta(\log n / \log\log n)$.
- $\delta = \mathrm{poly}(1/n)$.

The exact constants for the construction depend on the error parameter c, and will be fixed by the analysis in Section 3.2. Note that Corollary 2.1 provides such families $\mathcal{H}_i$ where each function $h_i \in \mathcal{H}_i$ is represented using $O(\log u + k_i \ell_i + \log(1/\delta)) = O(\log n)$ bits and evaluated in time $O(\log u + k_i \ell_i + \log(1/\delta)) = O(\log n)$. Each function $f \in \mathcal{F}$ in our construction consists of d functions $h_1 \in \mathcal{H}_1, \ldots, h_d \in \mathcal{H}_d$ that are sampled independently and uniformly at random. For any $x \in [u]$ we define

$$f(x) = h_1(x) \circ \cdots \circ h_d(x) \ .$$

[4]We note that various approaches based on multilevel hashing are fundamental and widely used in the design of data structures (see, for example, [FKS84]).

3.2 Analyzing the Construction

We naturally view the construction as a tree consisting of $d+1$ levels that are numbered $0, \ldots, d$, where levels $0, \ldots, d-1$ consist of "intermediate" bins, and level d consists of the actual n bins to which the elements are hashed. For a given set $S \subseteq [u]$ of size n, level 0 consists of a single bin containing the n elements of S. Level 1 consists of 2^{ℓ_1} bins, to which the elements of S are hashed using the function h_1. More generally, each level $i \in \{1, \ldots, d-1\}$ consists of $2^{\sum_{j=1}^{i} \ell_j}$ bins, and the elements of each such bin are hashed into $2^{\ell_{i+1}}$ bins in level $i+1$ using the function h_{i+1}.

Recall that we defined $n_0 = n$, and $n_i = n_{i-1}/2^{\ell_i}$ for every $i \in [d]$, so n_i is roughly equal to $n_{i-1}^{3/4}$. The number n_i is the expected number of elements in each bin in level i, and we show that with high probability no bin in levels $0, \ldots, d-1$ contains more than $(1+\alpha)^i n_i$ elements, where $\alpha = \Omega(1/\log\log n)$. This will be guaranteed by the following lemma.

Lemma 3.2. *For any $i \in \{0, \ldots, d-2\}$, $\alpha = \Omega(1/\log\log n)$, $0 < \alpha_i < 1$, and set $S_i \subseteq [u]$ of size at most $(1+\alpha_i)n_i$ it holds that*

$$\Pr_{h_{i+1} \leftarrow \mathcal{H}_{i+1}} \left[\max_{y \in \{0,1\}^{\ell_{i+1}}} \left| h_{i+1}^{-1}(y) \cap S_i \right| \leq (1+\alpha)(1+\alpha_i)n_{i+1} \right] > 1 - \frac{1}{n^{c+1}} \ .$$

Proof. Fix $y \in \{0,1\}^{\ell_{i+1}}$, let $X = \left| h_{i+1}^{-1}(y) \cap S_i \right|$, and assume without loss of generality that $|S_i| \geq \lfloor (1+\alpha_i)n_i \rfloor$ (as otherwise dummy elements can be added). Then X is the sum of $|S_i|$ indicator random variables that are k_{i+1}-wise δ-dependent, and has expectation $\mu = |S_i|/2^{\ell_{i+1}}$. Lemma 2.2 states that

$$\begin{aligned} \Pr\left[X > (1+\alpha)\mu\right] &\leq 2\left(\frac{|S_i| k_{i+1}}{(\alpha\mu)^2}\right)^{k_{i+1}/2} + \delta\left(\frac{|S_i|}{\alpha\mu}\right)^{k_{i+1}} \\ &= 2\left(\frac{2^{2\ell_{i+1}} k_{i+1}}{\alpha^2 |S_i|}\right)^{k_{i+1}/2} + \delta\left(\frac{2^{\ell_{i+1}}}{\alpha}\right)^{k_{i+1}} \ . \end{aligned}$$

We now upper bound each of above two summands separately. For the first one, recall that $\ell_{i+1} \leq (\log n_i)/4$, and combined with the facts that $|S_i| \geq (1+\alpha_i)n_i - 1 \geq n_i$ and $\alpha = \Omega(1/\log\log n)$, this yields

$$2\left(\frac{2^{2\ell_{i+1}} k_{i+1}}{\alpha^2 |S_i|}\right)^{k_{i+1}/2} \leq 2\left(\frac{k_{i+1}}{\alpha^2 2^{2\ell_{i+1}}}\right)^{k_{i+1}/2} \leq \frac{1}{2n^{c+2}} \ , \tag{3.1}$$

where the last inequality follows from the choice of k_{i+1} and ℓ_{i+1} such that $k_{i+1}\ell_{i+1} = \Omega(\log n)$. This also enables us to upper bound the second summand, noting that for an appropriate choice of $\delta = \mathrm{poly}(1/n)$ it holds that

$$\delta\left(\frac{2^{\ell_{i+1}}}{\alpha}\right)^{k_{i+1}} \leq \frac{1}{2n^{c+2}} \ . \tag{3.2}$$

Therefore, by combining Equations (3.1) and (3.2), and recalling that $n_{i+1} = n_i/2^{\ell_{i+1}}$ we obtain

$$\begin{aligned} \Pr\left[X > (1+\alpha)(1+\alpha_i)n_{i+1}\right] &= \Pr\left[X > (1+\alpha)(1+\alpha_i)\frac{n_i}{2^{\ell_{i+1}}}\right] \\ &\leq \Pr\left[X > (1+\alpha)\mu\right] \\ &\leq \frac{1}{n^{c+2}} \ . \end{aligned}$$

The lemma now follows by a union bound over all $y \in \{0,1\}^{\ell_{i+1}}$ (there are at most n such values). ■

We are now ready to prove Theorem 3.1. The description length and evaluation time of our construction were already argued in Section 3.1, and therefore we focus here on the maximal load.

Proof of Theorem 3.1. Fix a set $S \subseteq [u]$ of size n. We begin the proof by inductively arguing that for every level $i \in \{0, \dots, d-1\}$, with probability at least $1 - i/n^{c+1}$ the maximal load in level i is at most $(1+\alpha)^i n_i$ elements per bin, where $\alpha = \Omega(1/\log\log n)$. For $i = 0$ this follows by our definition of level 0: it contains a single bin with the $n_0 = n$ elements of S. Assume now that the claim holds for level i, and we now directly apply Lemma 3.2 for each bin in level i with $(1+\alpha_i) = (1+\alpha)^i$. A union bound over all bins in level i (at most n such bins), implies that with probability at least $1 - (i/n^{c+1} + 1/n^{c+1})$ the maximal load in level $i+1$ is at most $(1+\alpha)^{i+1} n_{i+1}$, and the inductive claim follows. In particular, this guarantees that with probability at least $1 - (d-1)/n^{c+1}$, the maximal load in level $d-1$ is $(1+\alpha)^{d-1} n_{d-1} \leq 2n_{d-1}$, for an appropriate choice of $d = O(\log\log n)$.

Now we would like to upper bound the number n_{d-1}. Note that for every $i \in [d-1]$ it holds that $\ell_i \geq (\log n_{i-1})/4 - 1$, and therefore $n_i = n_{i-1}/2^{\ell_i} \leq 2n_{i-1}^{3/4}$. By simple induction this implies $n_i \leq 2^{\sum_{j=0}^{i-1}(3/4)^j} n^{(3/4)^i} \leq 16 n^{(3/4)^i}$. Thus, for an appropriate choice of $d = O(\log\log n)$ it holds that $n_{d-1} \leq \log n$. In addition, the definition of the n_i's implies that $n_{d-1} = n/2^{\sum_{j=1}^{d-1}\ell_j}$, and therefore $\ell_d = \log n - \sum_{i=j}^{d-1} \ell_j = \log n_{d-1}$.

That is, in level $d-1$ of the tree, with probability at least $1 - (d-1)/n^{c+1}$, each bin contains at most $2n_{d-1} \leq 2\log n$ elements, and these elements are hashed into n_{d-1} bins using the function h_d. The latter function is k_d-wise δ-dependent, where $k_d = \Omega(\log n/\log\log n)$ and therefore the probability that any $t = \gamma \log n/\log\log n \leq k_d$ elements from level $d-1$ are hashed into any specific bin in level d is at most

$$\begin{aligned}
\binom{2n_{d-1}}{t}\left(\left(\frac{1}{n_{d-1}}\right)^t + \delta\right) &\leq \left(\frac{2en_{d-1}}{t}\right)^t \left(\left(\frac{1}{n_{d-1}}\right)^t + \delta\right) \\
&\leq \left(\frac{2e}{t}\right)^t + \delta\left(\frac{2en_{d-1}}{t}\right)^t \\
&\leq \left(\frac{2e\log\log n}{\gamma \log n}\right)^{\frac{\gamma\log n}{\log\log n}} + \delta\left(\frac{2e\log\log n}{\gamma}\right)^{\frac{\gamma\log n}{\log\log n}} \\
&\leq \frac{1}{2n^{c+3}} + \frac{1}{2n^{c+3}} \\
&= \frac{1}{n^{c+3}} \ ,
\end{aligned}$$

for an appropriate choice of $t = \gamma\log n/\log\log n$ and $\delta = \mathrm{poly}(1/n)$. This holds for any pair of bins in levels $d-1$ and d, and therefore a union bound over all such bins implies that the probability that there exists a bin in level d with more than t elements is at most $1/n^{c+1}$. This implies that with probability at least $1 - d/n^{c+1} > 1 - 1/n^c$ a randomly chosen function f has a maximal load of $\gamma\log n/\log\log n$. ■

4 A Construction Based on Generators for Space-Bounded Computations

The starting point of our second construction is the observation that any pseudorandom generator which fools small-width branching programs directly defines a family of functions with the desired maximal load. Specifically, for a universe $[u]$, a generator that produces u blocks of length $\log n$ bits each can be interpreted as a function $f : [u] \to [n]$, where for any input $x \in [u]$ the value

$h(x)$ is defined to be the xth output block of the generator. Fixing a subset $S \subseteq [u]$, we observe that the event in which the load of any particular bin is larger than $t = O(\log n / \log\log n)$ can be recognized by a branching program of width $t+1 < n$ (all the program needs to do is count up to t). Assuming the existence of such an explicit generator with seed of length $O(\log n)$ bits implies a family of functions with description length of $O(\log n)$ bits (which is optimal up to a constant factor).

Unfortunately, the best known generator [Nis92] has seed of length $\Omega(\log^2 n)$ bit, which essentially does not give any improvement over using $O(\log n/\log\log n)$-wise independence. Still, our construction uses a pseudorandom generator in an inherent way, but instead of generating $O(u \log n)$ bits directly it will only produce $O\left(2^{\log^{1/2} n}\right)$ descriptions of $O\left(\log^{1/2} n\right)$-wise independent functions. Our construction will combine these functions into a single function $f : [u] \rightarrow [n]$. The construction is inspired by the pseudorandom generator of Lu [Lu02] for the class of combinatorial rectangles (which by itself seems too weak in our context).

We now describe our family $\mathcal{F}$. Let $\mathcal{H}$ be a family of k_1-wise independent functions $h : [u] \rightarrow [\ell]$ for $k_1 = O(1)$ and $\ell = O\left(2^{\log^{1/2} n}\right)$, and let $\mathcal{G}$ be a family of k_2-wise independent functions $g : [u] \rightarrow [n]$ for $k_2 = O\left(\log^{1/2} n\right)$. Each function $f \in \mathcal{F}$ consists of a function $h \in \mathcal{H}$ that is sampled uniformly at random, and of ℓ functions $g_1, \ldots, g_\ell \in \mathcal{G}$ that are obtained as the output of a pseudorandom generator. The description of each g_j is given by the jth output block of the generator. For any $x \in [u]$ we define

$$f(x) = g_{h(x)}(x) \ .$$

Using the generator provided by Theorem 2.9, the description length of each such f is only $O\left(\log^{3/2} n\right)$ bits. Moreover, we present a new construction of a pseudorandom generator in which the description of each g_j can be computed in time $O\left(\log^{1/2} n\right)$, without increasing the length of the seed. Thus, for any $x \in [u]$ the time required for computing $f(x) = g_{h(x)}(x)$ is $O\left(\log^{1/2} n\right)$: the value $h(x)$ can be computed in time $O(k_1) = O(1)$, the description of $g_{h(x)}$ can be computed in time $O\left(\log^{1/2} n\right)$, and then the value $g_{h(x)}(x)$ can be computed in time $O(k_2) = O\left(\log^{1/2} n\right)$. This allows us to prove the following theorem:

Theorem 4.1. *For any constant $c > 0$ and integers n and $u = \mathrm{poly}(n)$ there exists a family $\mathcal{F}$ of functions $f : [u] \rightarrow [n]$ satisfying the following properties:*

1. *Each $f \in \mathcal{F}$ is described using $O\left(\log^{3/2} n\right)$ bits.*
2. *For any $f \in \mathcal{F}$ and $x \in [u]$ the value $f(x)$ can be computed in time $O\left(\log^{1/2} n\right)$ in the unit cost RAM model.*
3. *There exists a constant $\gamma > 0$ such that for any set $S \subseteq [u]$ of size n it holds that*

$$\Pr_{f \leftarrow \mathcal{F}}\left[\max_{i \in [n]} \left|f^{-1}(i) \cap S\right| \le \frac{\gamma \log n}{\log\log n}\right] > 1 - \frac{1}{n^c} \ .$$

The proof of Theorem 4.1 proceeds in three stages. First, in Section 4.1 we analyze the basic family $\widehat{\mathcal{F}}$ that is obtained by sampling the functions $g_1, \ldots, g_\ell$ independently and uniformly at random. Then, in Section 4.2 we show that one can replace the descriptions of $g_1, \ldots, g_\ell$ with the output of a pseudorandom generator that uses a seed of length $O\left(\log^{3/2} n\right)$ bits. Finally in Section 4.3 we present a new generator that makes it possible to compute the description of each function g_j in time $O\left(\log^{1/2} n\right)$ without increasing the length of the seed.

4.1 Analyzing the Basic Construction

We begin by analyzing the basic family $\widehat{\mathcal{F}}$ in which each function $\hat{f}$ is obtained by sampling the functions $h \in \mathcal{H}$ and $g_1, \ldots, g_\ell \in \mathcal{G}$ independently and uniformly at random, and defining $\hat{f}(x) = g_{h(x)}(x)$ for any $x \in [u]$. For analyzing $\widehat{\mathcal{F}}$ we naturally interpret it as a two-level process: The function h maps the elements into $\ell = O\left(2^{\log^{1/2} n}\right)$ first-level bins, and then the elements of each such bin are mapped into n second-level bins using the function g_j for the jth first-level bin.

When hashing a set $S \subseteq [u]$ of size n, we expect each first-level bin to contain roughly n/ℓ elements, and in Claim 4.2 we observe that this in fact holds with high probability (this is a rather standard argument). Then, given that each of the first-level bins contains at most, say, $2n/\ell$ elements, in Claim 4.3 we show that if the g_j's are sampled independently and uniformly at random from a family of $O\left(\log^{1/2} n\right)$-wise independent functions, then the maximal load in the second-level bins is $O(\log n/\log\log n)$ with high probability.

For a set $S \subset [u]$ denote by S_j the subset of S mapped to first level bin j; i.e. $S_j = S \cap h^{-1}(j)$.

Claim 4.2. *For any set $S \subseteq [u]$ of size n it holds that*

$$\Pr_{h \leftarrow \mathcal{H}}\left[\max_{j \in [\ell]} |S_j| \le 2 \cdot \frac{n}{\ell}\right] > 1 - \frac{1}{n^{c+5}} \quad .$$

Proof. For any $j \in [\ell]$ the random variable $|S_j|$ is the sum of n binary random variables that are k_1-wise independent, and has expectation n/ℓ. Letting $\ell = \beta 2^{\log^{1/2} n}$ for some constant $\beta > 0$, Lemma 2.2 (when setting $\delta = 0$ and $t = n/\ell$) guarantees that for an appropriate choice of the constant k_1 it holds that

$$\begin{aligned}\Pr_{h \leftarrow \mathcal{H}}\left[|S_j| > 2 \cdot \frac{n}{\ell}\right] &\le 2\left(\frac{nk_1}{(n/\ell)^2}\right)^{k_1/2} \\ &= 2\left(\frac{\beta k_1}{2^{\log n - 2\sqrt{\log n}}}\right)^{k_1/2} \\ &\le \frac{1}{n^{c+6}} \quad .\end{aligned}$$

This holds for any $j \in [\ell]$, and therefore a union bound over all $\ell \le n$ values yields

$$\begin{aligned}\Pr_{h \leftarrow \mathcal{H}}\left[\max_{j \in [\ell]} |S_j| > 2 \cdot \frac{n}{\ell}\right] &\le \ell \cdot \frac{1}{n^{c+6}} \\ &\le \frac{1}{n^{c+5}} \quad .\end{aligned}$$

■

Claim 4.3. *There exists a constant $\gamma > 0$ such that for any set $S \subseteq [u]$ of size n and for any $i \in [n]$ it holds that*

$$\Pr_{\hat{f} \leftarrow \widehat{\mathcal{F}}}\left[\left|\hat{f}^{-1}(i) \cap S\right| \le \frac{\gamma \log n}{\log\log n}\right] > 1 - \frac{1}{n^{c+4}} \quad .$$

Proof. For any set $S \subseteq [u]$ of size n, Claim 4.2 guarantees that with probability at least $1 - 1/n^{c+5}$ it holds that

$$\max_{j \in [\ell]} |S_j| \leq 2 \cdot \frac{n}{\ell} \enspace .$$

From this point on we condition on the latter event and fix the function h. Let $k_{i,j} = |S_j \cap g_j^{-1}(i)|$ be the number of elements mapped to second level bin i via first level bin j. The event in which one of the n second-level bins contains more than $t = O(\log n / \log \log n)$ elements is the union of the following two events:

- Event 1: There exists a second-level bin $i \in [n]$ and first level bin j such that $k_{i,j} \geq \alpha \log^{1/2} n$ for some constant α. If we set the functions g_j to be $\alpha \log^{1/2} n$-wise independent, the probability of this event is at most

$$\begin{aligned} \binom{|S_j|}{\alpha \log^{1/2} n} \cdot \left(\frac{1}{n}\right)^{\alpha \log^{1/2} n} &\leq |S_j|^{\alpha \log^{1/2} n} \cdot \left(\frac{1}{n}\right)^{\alpha \log^{1/2} n} \\ &\leq \left(\frac{2}{\ell}\right)^{\alpha \log^{1/2} n} \\ &\leq \frac{1}{n^{c+7}} \enspace , \end{aligned}$$

  for an appropriate choice of $\ell = O\left(2^{\log^{1/2} n}\right)$ and α. There are $n\ell < n^2$ such pairs of bins, and therefore a union bound implies that this event occurs with probability at most $1/n^{c+5}$.

- Event 2: Some second-level bin $i \in [n]$ has $t = O(\log n / \log \log n)$ elements with $k_{i,j} \leq \alpha \log^{1/2} n$ for all j. Since $g_1, \ldots, g_\ell$ are sampled independently from a family that is $\alpha \log^{1/2} n$-wise independent, the probability of this event is at most

$$\begin{aligned} \binom{n}{t} \left(\frac{1}{n}\right)^t &\leq \left(\frac{ne}{t}\right)^t \left(\frac{1}{n}\right)^t \\ &= \left(\frac{e}{t}\right)^t \\ &\leq \frac{1}{n^{c+6}} \enspace , \end{aligned}$$

  for an appropriate choice of $t = O(\log n / \log \log n)$. This holds for any second-level bin $i \in [n]$, and therefore a union bound over all n such bins implies that this event occurs with probability at most $1/n^{c+5}$.

Combining all of the above, we obtain that there exists a constant γ such that for all sufficiently large n it holds that

$$\Pr_{\hat{f} \leftarrow \widehat{\mathcal{F}}} \left[\max_{i \in [n]} \left| \hat{f}^{-1}(i) \cap S \right| \leq \frac{\gamma \log n}{\log \log n} \right] > 1 - \frac{3}{n^{c+5}} > 1 - \frac{1}{n^{c+4}} \enspace .$$

■

4.2 Derandomizing the Basic Construction

As discussed above, the key observation that allows the derandomization $g_1, \ldots, g_\ell \in \mathcal{G}$ is the fact that the event in which the load of any particular bin is larger than $t = O(\log n / \log \log n)$ can be recognized in $O(\log n)$ space (and, more accurately, in $O(\log t)$ space). Specifically, fix a set $S \subseteq [u]$ of size n, a second-level bin $i \in [n]$, and a function $h \in \mathcal{H}$. Consider the layered branching program $M_{S,h,i}$ that has $\ell+1$ layers each of which contains roughly n vertices, where every layer $j \in [\ell]$ takes as input the description of the function g_j and keeps count of the number of elements from S that are mapped to bin i using the functions $h, g_1, \ldots, g_j$. In other words, the jth layer adds to the count the number of elements $x \in S$ such that $h(x) = j$ and $g_j(x) = i$. Each vertex in the final layer is labeled with 0 and 1 depending on whether the count has exceeded t (note that there is no reason to keep on counting beyond t, and therefore it suffices to have only t vertices in each layer).

Let $G : \{0,1\}^m \to (\{0,1\}^v)^\ell$ be a generator that ϵ-fools any (s,v,ℓ)-LBP, where $\epsilon = 1/n^{c+4}$, $s = O(\log n)$, $v = O(k_2 \log n) = O\left(\log^{3/2} n\right)$, and $\ell = O\left(2^{\log^{1/2} n}\right)$. Theorem 2.9 provides an explicit construction of such a generator with a seed of length $m = O(v + \log \ell \cdot (s + \log \ell + \log(1/\epsilon))) = O\left(\log^{3/2} n\right)$. For any seed $x \in \{0,1\}^m$ we use $G(x) = (g_1, \ldots, g_\ell) \in (\{0,1\}^v)^\ell$ for providing the descriptions of the function $g_1, \ldots, g_\ell$.

By combining Claim 4.3 with the fact that G is a generator that ϵ-fools any (s,v,ℓ)-LBP we obtain the following claim:

Claim 4.4. *There exists a constant $\gamma > 0$ such that for any set $S \subseteq [u]$ of size n it holds that*

$$\Pr_{f \leftarrow \mathcal{F}}\left[\max_{i \in [n]} \left|f^{-1}(i) \cap S\right| \le \frac{\gamma \log n}{\log \log n}\right] > 1 - \frac{1}{n^c} \quad .$$

Proof. Let $\gamma > 0$ be the constant specified by Claim 4.3, and again denote by $\widehat{\mathcal{F}}$ the basic family obtained by sampling $h \in \mathcal{H}$ and $g_1, \ldots, g_\ell \in \mathcal{G}$ independently and uniformly at random. Then for any set $S \subseteq [u]$ of size n and $i \in [n]$ it holds that

$$\begin{aligned}
\Pr_{f \leftarrow \mathcal{F}}\left[\left|f^{-1}(i) \cap S\right| > \frac{\gamma \log n}{\log \log n}\right] &= \Pr_{h \leftarrow \mathcal{H}}\left[\Pr_{x \leftarrow \{0,1\}^m}\left[M_{S,h,i}(G(x)) = 1\right]\right] \\
&\le \Pr_{h \leftarrow \mathcal{H}}\left[\Pr_{g_1,\ldots,g_\ell \leftarrow \mathcal{G}}\left[M_{S,h,i}(g_1,\ldots,g_\ell) = 1\right]\right] + \frac{1}{n^{c+4}} \qquad (4.1) \\
&= \Pr_{\hat{f} \leftarrow \widehat{\mathcal{F}}}\left[\left|\hat{f}^{-1}(i) \cap S\right| > \frac{c \log n}{\log \log n}\right] + \frac{1}{n^{c+4}} \\
&< \frac{2}{n^{c+4}} \quad , \qquad (4.2)
\end{aligned}$$

where Equation (4.1) follows from the fact that G is a generator that $1/n^{c+4}$-fools the branching program $M_{S,h,i}$, and Equation (4.2) follows from Claim 4.3. A union bound over all bins $i \in [n]$ yields

$$\begin{aligned}
\Pr_{f \leftarrow \mathcal{F}}\left[\max_{i \in [n]} \left|f^{-1}(i) \cap S\right| \ge \frac{c \log n}{\log \log n}\right] &\le n \cdot \frac{2}{n^{c+4}} \\
&\le \frac{1}{n^c} \quad .
\end{aligned}$$

■

4.3 A More Efficient Generator

As shown in Section 4.2, we can instantiate our construction with any generator $G : \{0,1\}^m \to (\{0,1\}^v)^\ell$ that ϵ-fools any (s,v,ℓ)-LBP, where $s = O(\log n)$, $v = O\left(\log^{3/2} n\right)$, $\ell = O\left(2^{\log^{1/2} n}\right)$, and

$\epsilon = 1/n^{c+4}$. The generator constructed by Impagliazzo, Nisan, and Wigderson [INW94] (following [Nis92]), whose parameters we stated in Theorem 2.9, provides one such instantiation, but for this generator the time to compute each v-bit output block seems at least logarithmic. Here we construct a more efficient generator for our parameters, where each v-bit output block can be computed in time $O\left(\log^{1/2} n\right)$ without increasing the length of the seed.

The generator we propose uses as a building blocks the generators constructed by Nisan [Nis92] and by Nisan and Zuckerman [NZ96]. In what follows we first provide a high-level description of these two generators, and then turn to describe our own generator.

4.3.1 Nisan's Generator

Let $\mathcal{H}$ be a family of pairwise independent functions $h : \{0,1\}^{v_2} \to \{0,1\}^{v_2}$. For every integer $k \geq 0$ Nisan [Nis92] constructed a generator $G_{\mathsf{N}}^{(k)} : \{0,1\}^{v_2} \times \mathcal{H}^k \to \left(\{0,1\}^{v_2}\right)^{2^k}$ that is defined recursively by $G_{\mathsf{N}}^{(0)}(x) = x$, and

$$G_{\mathsf{N}}^{(k)}(x, h_1, \ldots, h_k) = G_{\mathsf{N}}^{(k-1)}(x, h_1, \ldots, h_{k-1}) \circ G_{\mathsf{N}}^{(k-1)}(h_k(x), h_1, \ldots, h_{k-1}) \ ,$$

where $\circ$ denotes the concatenation operator. For any integers v_2 and k, Nisan proved that $G_{\mathsf{N}}^{(k)}$ is a generator that 2^{-v_2}-fools any $(v_2, v_2, 2^k)$-LBP.

When viewing the output of the generator as the concatenation of 2^k blocks of length v_2 bits each, we observe that each such block can be computed by evaluating k pairwise independent functions. In our setting we are interested in $v_2 = O(\log n)$ and $2^k = O\left(2^{\log^{1/2} n}\right)$, and in this case each output block can be computed in time $O\left(\log^{1/2} n\right)$. Formally, from Nisan's generator we obtain the following corollary:

Corollary 4.5. *For any $s_2 = O(\log n)$, $v_2 = O(\log n)$, $\ell_2 = O\left(2^{\log^{1/2} n}\right)$, and $\epsilon = \mathrm{poly}(1/n)$, there exists a generator $G_{\mathsf{N}} : \{0,1\}^{m_2} \to \left(\{0,1\}^{v_2}\right)^{\ell_2}$ that ϵ-fools any (s_2, v_2, ℓ_2)-LBP, where $m_2 = O\left(\log^{3/2} n\right)$. In the unit cost RAM model with a word size of $w = \Omega(\log n)$ bits each v_2-bit output block can be computed in time $O\left(\log^{1/2} n\right)$.*

4.3.2 The Nisan-Zuckerman Generator and an Efficient Instantiation

Given a (k, ϵ)-extractor $\mathsf{Ext} : \{0,1\}^{t_1} \times \{0,1\}^{d_1} \to \{0,1\}^{v_1}$ Nisan and Zuckerman [NZ96] constructed a generator $G_{\mathsf{NZ}}^{\mathsf{Ext}} : \{0,1\}^{t_1} \times \left(\{0,1\}^{d_1}\right)^{\ell_1} \to \left(\{0,1\}^{v_1}\right)^{\ell_1}$ that is defined as

$$G_{\mathsf{NZ}}^{\mathsf{Ext}}(x, y_1, \ldots, y_{\ell_1}) = \mathsf{Ext}(x, y_1) \circ \cdots \circ \mathsf{Ext}(x, y_{\ell_1}) \ ,$$

where $\circ$ denotes the concatenation operator. When viewing the output of the generator as the concatenation of ℓ_1 blocks of length v_1 bits each, we observe that the time to compute each such block is the time to compute the extractor Ext. In our setting we are interested in $t_1 = O\left(\log^{3/2} n\right)$, $v_1 = O\left(\log^{3/2} n\right)$, $k = t_1 - O(\log n)$, and $\epsilon = \mathrm{poly}(1/n)$, and in Lemma 4.7 we construct an extractor that has a seed of length $d_1 = O(\log n)$ bits and can be computed in time $O\left(\log^{1/2} n\right)$. As a corollary, from the Nisan-Zuckerman generator when instantiated with our extractor we obtain:

Corollary 4.6. *For any $s_1 = O(\log n)$, $v_1 = O\left(\log^{3/2} n\right)$, $\ell_1 = O\left(2^{\log^{1/2} n}\right)$, and $\epsilon = \mathrm{poly}(1/n)$, there exists a generator $G_{\mathsf{NZ}}^{\mathsf{Ext}} : \{0,1\}^{m_1} \to (\{0,1\}^{v_1})^{\ell_1}$ that ϵ-fools any (s_1, v_1, ℓ_1)-LBP, where $m_1 = O\left(\log^{3/2} n + 2^{\log^{1/2} n} \cdot \log n\right)$. Moreover, there exists an extractor Ext such that in the unit cost RAM model with a word size of $w = \Omega(\log n)$ bits each v_1-bit output block of the generator $G_{\mathsf{NZ}}^{\mathsf{Ext}}$ can be computed in time $O\left(\log^{1/2} n\right)$.*

The following lemma presents the extractor that we use for instantiating the Nisan-Zuckerman generator.

Lemma 4.7. *Let $t_1 = \Theta\left(\log^{3/2} n\right)$, $\Delta = O(\log n)$ and $\epsilon = \mathrm{poly}(1/n)$. There exists a $(t_1 - \Delta, \epsilon)$-extractor $\mathsf{Ext} : \{0,1\}^{t_1} \times \{0,1\}^{d_1} \to \{0,1\}^{v_1}$, where $d_1 = O(\log n)$ and $v_1 = \Omega\left(\log^{3/2} n\right)$, that can be computed in time $O\left(\log^{1/2} n\right)$ in the unit cost RAM model with a word size of $w = \Omega(\log n)$ bits.*

Proof. Given a random variable X over $\{0,1\}^{t_1}$ we partition it into $T = t_1/z$ consecutive blocks $X = X_1 \circ \cdots \circ X_T$ each of length z bits, where $z = \lceil 2(\Delta + 3\log(2T/\epsilon)) \rceil = O(\log n)$. Without loss of generality we assume that z divides t_1, and otherwise we ignore the last block. Corollary 2.5 guarantees that X is a $(T, z - \Delta - \log(2T/\epsilon), \epsilon/2T)$-block source. Let $\mathcal{H}$ be a family of pair-wise independent functions $h : \{0,1\}^z \to \{0,1\}^{z'}$, where $z' = \lfloor z - \Delta - 3\log(2T/\epsilon) \rfloor = \Omega(\log n)$ and each $h \in \mathcal{H}$ is described by $d_1 = O(\log n)$ bits. We define an extractor $\mathsf{Ext} : \{0,1\}^{t_1} \times \mathcal{H} \to \{0,1\}^{Tz'}$ by applying a randomly chosen $h \in \mathcal{H}$ to each of the T blocks of the source. That is,

$$\mathsf{Ext}(x_1 \circ \cdots \circ x_T, h) = h(x_1) \circ \cdots \circ h(x_T) \ .$$

Lemma 2.7 implies that the distribution $(h, h(x_1), \ldots, h(x_T))$ is ϵ-close to the distribution $(h, y_1, \ldots, y_T)$, where $h \leftarrow \mathcal{H}$, $(x_1, \ldots, x_T) \leftarrow X$, and $(y_1, \ldots, y_T) \leftarrow \left(\{0,1\}^{z'}\right)^T$. In addition, in the unit cost RAM model with a word size of $w = \Omega(\log n)$ bits each application of h can be done in constant time, and therefore the extractor can be computed in time $T = O\left(\log^{1/2} n\right)$. Finally, note that the number of outputs bits is $Tz' = t_1 z'/z = \Omega\left(\log^{3/2} n\right)$. ■

4.3.3 Our Generator

Recall that we are interested in a generator $G : \{0,1\}^m \to (\{0,1\}^v)^\ell$ that ϵ-fools any (s, v, ℓ)-LBP, where $s = O(\log n)$, $v = O\left(\log^{3/2} n\right)$, $\ell = O\left(2^{\log^{1/2} n}\right)$, and $\epsilon = 1/n^{c+4}$. Let $G_{\mathsf{NZ}} : \{0,1\}^{m_1} \to (\{0,1\}^v)^\ell$ be the Nisan-Zuckerman generator that is given by Corollary 4.6 that $\epsilon/2$-fools any (s, v, ℓ)-LBP, where $m_1 = O\left(\log^{3/2} n + 2^{\log^{1/2} n} \cdot \log n\right)$. In addition, let $G_{\mathsf{N}} : \{0,1\}^{m_2} \to (\{0,1\}^{v_2})^\ell$ be Nisan's generator that is given by Corollary 4.5 that $\epsilon/2$-fools any (s, v_2, ℓ)-LBP, where $v_2 = O(\log n)$ and $m_2 = O\left(\log^{3/2} n\right)$. We define a generator G as follows:

$$G(x_1, x_2) = G_{\mathsf{NZ}}(x_1, G_{\mathsf{N}}(x_2)) \ .$$

That is, given a seed (x_1, x_2) it first computes the output $(y_1, \ldots, y_\ell)$ of Nisan's generator using the seed x_2, and then it computes the output $\mathsf{Ext}(x_1, y_1) \circ \cdots \circ \mathsf{Ext}(x_1, y_{\ell_1})$ of the Nisan-Zuckerman generator. Observe that the time to compute the ith v-bit output block is the time to compute the ith output block for both generators, which is $O\left(\log^{1/2} n\right)$. In addition, note that the length of seed is $O\left(\log^{3/2} n\right)$ bits since each of x_1 and x_2 is of length $O\left(\log^{3/2} n\right)$ bits. Thus, it only remains to prove that G indeed ϵ-fools any (s, v, ℓ)-LBP. This is proved in the following lemma.

Lemma 4.8. *For the parameters s, v, ℓ, m, and ϵ specified above, G is a generator that ϵ-fools any (s, v, ℓ)-LBP.*

Proof. Let M be an (s, v, ℓ)-LBP, and let $m'_1 = O\left(\log^{3/2} n\right)$. Then,

$$\begin{aligned}
&\left|\Pr_{x\leftarrow\{0,1\}^m}[M(G(x))=1] - \Pr_{u\leftarrow\{0,1\}^{v\ell}}[M(u)=1]\right| \\
&= \left|\Pr_{\substack{x_1\leftarrow\{0,1\}^{m'_1}\\ x_2\leftarrow\{0,1\}^{m_2}}}[M(G_{\mathsf{NZ}}(x_1,G_{\mathsf{N}}(x_2)))=1] - \Pr_{u\leftarrow\{0,1\}^{v\ell}}[M(u)=1]\right| \\
&\leq \left|\Pr_{\substack{x_1\leftarrow\{0,1\}^{m'_1}\\ x_2\leftarrow\{0,1\}^{m_2}}}[M(G_{\mathsf{NZ}}(x_1,G_{\mathsf{N}}(x_2)))=1] - \Pr_{\substack{x_1\leftarrow\{0,1\}^{m'_1}\\ y_2\leftarrow\{0,1\}^{v_2\ell}}}[M(G_{\mathsf{NZ}}(x_1,y_2)=1]\right| \qquad (4.3)\\
&\quad + \left|\Pr_{\substack{x_1\leftarrow\{0,1\}^{m'_1}\\ y_2\leftarrow\{0,1\}^{v_2\ell}}}[M(G_{\mathsf{NZ}}(x_1,y_2)=1] - \Pr_{u\leftarrow\{0,1\}^{v\ell}}[M(u)=1]\right| . \qquad (4.4)
\end{aligned}$$

We now show that the expressions in Equations (4.3) and (4.4) are upper bounded by $\epsilon/2$ each, and thus the lemma follows. The expression in Equation (4.4) is the advantage of M in distinguishing between the output of G_{NZ} (with a uniformly chosen seed) and the uniform distribution. Since G_{NZ} was chosen to $\epsilon/2$-fool any (s, v, ℓ)-LBP, then this advantage is upper bounded by $\epsilon/2$.

For bounding the expression in Equation (4.3) it suffices to bound the expression resulting by fixing x_1^* to be the value of x_1 that maximizes it. Then, we define $M^*(\cdot) = M(G_{\mathsf{NZ}}(x_1^*, \cdot))$, which is an (s, v_2, ℓ)-LBP. Since G_{N} was chosen to $\epsilon/2$-fool any (s, v_2, ℓ)-LBP, we obtain

$$\begin{aligned}
&\left|\Pr_{\substack{x_1\leftarrow\{0,1\}^{m'_1}\\ x_2\leftarrow\{0,1\}^{m_2}}}[M(G_{\mathsf{NZ}}(x_1,G_{\mathsf{N}}(x_2)))=1] - \Pr_{\substack{x_1\leftarrow\{0,1\}^{m'_1}\\ y_2\leftarrow\{0,1\}^{v_2\ell}}}[M(G_{\mathsf{NZ}}(x_1,y_2)=1]\right| \\
&\leq \left|\Pr_{x_2\leftarrow\{0,1\}^{m_2}}[M^*(G_{\mathsf{N}}(x_2))=1] - \Pr_{y_2\leftarrow\{0,1\}^{v_2\ell}}[M^*(y_2)=1]\right| \\
&\leq \frac{\epsilon}{2} .
\end{aligned}$$

■

5 Lower Bounds for Balls-and-Bins Hash Functions

In this section we provide formal proofs of two somewhat folklore lower bounds. First, in Theorem 5.1 we show that for $u \geq n^2$ any family $\mathcal{H}$ of functions $h : [u] \to [n]$ has a maximal load of $\Omega(\log n/ \log\log n)$ with high probability when hashing n balls into n bins. Then, in Theorem 5.4 we show that any such family that guarantees a maximal load of $O(\log n/ \log\log n)$ (and even of $O(\log n))$ with probability $1-\epsilon$ must be of size $\Omega(n/\epsilon \log n)$.

Theorem 5.1. *For all sufficiently large n and $u \geq n^2$, and for any family $\mathcal{H}$ of functions $h : [u] \to [n]$, there exists a set $S \subseteq [u]$ of size n such that*

$$\Pr_{h \leftarrow \mathcal{H}} \left[\max_{i \in [n]} \left| h^{-1}(i) \cap S \right| < \frac{\log n}{2 \log \log n} \right] < \frac{1}{n} \enspace .$$

By the minimax principle it is enough to show the following:

Lemma 5.2. *For all sufficiently large n and $u \geq n^2$, and for any fixed function $h : [u] \to [n]$ it holds that*

$$\Pr_{S \subseteq [u], |S| = n} \left[\max_{i \in [n]} \left| h^{-1}(i) \cap S \right| < \frac{\log n}{2 \log \log n} \right] < \frac{1}{n} \enspace .$$

Proof. Let S' denote a random multiset obtained by sampling n balls independently and uniformly at random from $[u]$, with replacement. We claim it is enough to show that the bound holds for S', in other words the collisions in the sampling of S' do not contribute much to the overall load. To see this observe that since $u \geq n^2$, the probability there is some ball that has been sampled at least 3 times is smaller than $1/n^3$. Now, let $p_i = \Pr_{x \leftarrow [u]}[h(u) = i]$ denote the probability a randomly sampled ball is placed in bin i. Thus, our goal is to bound the maximal load of a process where n balls are sequentially placed in bins where the allocation of each ball is determined by $\vec{p} = (p_1, \ldots, p_n)$. Let $\alpha(k, \vec{p})$ denote the probability that after n uniformly sampled balls are placed in n bins using h, the maximal load is at least k.

Claim 5.3. *For every k it holds that $\alpha(k, \vec{p}) \geq \alpha(k, \vec{u})$ where $\vec{u}$ is the uniform distribution over the n bins.*

Proof. The proof follows a symmetrization argument. Assume for contradiction that there exist two bins $i, j \in [n]$ such that both p_i and p_j are positive and different from $1/n$. Define $\vec{p}_{i,j}$ to be the vector obtained from $\vec{p}$ by replacing both p_i and p_j with $(p_i + p_j)/2$. It is enough to show that $\alpha(k, \vec{p}) \geq \alpha(k, \vec{p}_{i,j})$. Let A_ℓ be the event that bins i and j receive a total of exactly $\ell > 0$ balls. The probability of A_ℓ is the same both under $\vec{p}$ and $\vec{p}_{i,j}$. Further, conditioned on A_ℓ, the load of bins i and j is independent of the rest of the allocation, and the load of the remaining bins is distributed the same in both processes. Thus we can concentrate on bins i and j and the probability that one of them receives at least k balls conditioned on A_ℓ.

Observe that if $\ell \geq 2k - 1$ then with probability 1 one of the bins received at least k balls and the lemma holds trivially. We focus on the regime $\ell \leq 2k - 2$ in which at most one of the bins gets k balls. We define $p = p_i/(p_i + p_j)$ and

$$\Phi(p) = \sum_{m=k}^{\ell} \binom{\ell}{m} p^m (1-p)^{\ell - m}$$

Now, the probability that one of these bins receives at least k balls is $\Phi(p) + \Phi(1-p)$. When taking the derivative by p we get $\Phi'(p) - \Phi'(1-p)$ which obviously vanishes at $p = 1/2$. Also, the second derivative is positive in $(0, 1)$, which completes the proof of the claim. ■

The proof of Lemma 5.2 is completed by observing that when n balls are placed independently and uniformly at random in n bins, with probability $1 - 1/n$ there would be a bin with more than $\log n / \log \log n$ bins, c.f. [MU05, Lemma 5.12]. ■

Theorem 5.4. *For all n and $u \geq 2n \log n$, and for any family $\mathcal{H}$ of functions $h : [u] \to [n]$, if for any set $S \subseteq [u]$ of size n it holds that*

$$\Pr_{h \leftarrow \mathcal{H}} \left[\max_{i \in [n]} \left| h^{-1}(i) \cap S \right| < \log n \right] < \epsilon \ ,$$

then $|\mathcal{H}| \geq n/(\epsilon \log n)$.

Proof. Given a family $\mathcal{H}$ we show how to construct a bad set for it. Initially the bad set S^* is empty. Consider any function $h \in \mathcal{H}$. Since $u \geq 2n \log n$, there is a set $S_h \subseteq U$ such that $|S_h| \geq \log n$ and all elements of S_h are mapped by h to the same bin i. Thus, we can take $\log n$ elements of S_h and add them to S^*. This means $\left| h^{-1}(i) \cap S^* \right| \geq \log n$. Now, observe that $u - |S^*| \geq 2n \log n - \log n$, so we can repeat the argument for the next function h' by finding a set $S_{h'}$ which maps at least $2 \log n - 1$ elements into the same bin. Again, we can take $\log n$ of the elements in $S_{h'}$ and add them to S^*. We continue in this fashion, noting that at step i we have $u - |S^*| \geq 2n \log n - i \log n$. Hence, we can repeat this process $n/\log n$ times, until S^* contains n elements. These bad functions should be at most an ϵ fraction of the set of all functions, giving the desired bound. ■

6 Extensions and Open Problems

Applications. As discussed in Section 1.1, our two constructions can be successfully employed for storing elements using linear probing, guaranteeing an insertion time of $O(\log n)$ with high probability. An interesting open problem is whether the techniques developed in this paper, or similar techniques, can be used to construct small hash families that are suitable for other applications. For instance, $O(\log n)$-wise independence is known to suffice for two-choice hashing [ABK+99, Vöc99], cuckoo hashing [PR04], and more. Existing constructions have so far focused on simplicity and fast computation [DW03, Woe06, PP08, PT11], albeit with a significant increase to the description length.

Augmenting our constructions with *k*-wise independence. Our constructions can easily be augmented to offer $O(\log \log n)$-wise independence (for the first construction), and $O(\log^{1/2} n)$-wise independence (for the second construction) without affecting their description length and evaluation time. This may be useful, for example, in any application that involves tail bounds for limited independence.

Specifically, any function f resulting from either one of our constructions can be modified to $f(x) + h(x) \bmod n$, where h is sampled from a family of $O(\log \log n)$-wise independent functions for the first construction, and from a family of $O(\log^{1/2} n)$-wise independent functions for the second construction. Our analysis easily extends to this case, and the resulting functions clearly enjoy the level of independence offered by h. By implementing h using a polynomial of the appropriate degree, the description length and evaluation time of our constructions are not affected.

A time-space lower bound. Our constructions offer two different trade-offs between the description length and the evaluation time. It would be interesting to prove a lower bound on the time-space trade-off of any family that guarantees a maximal load of $O(\log n/\log \log n)$ with high probability when hashing n balls into n bins. For example, can we rule out constructions that are optimal in both aspects (i.e., description length $O(\log n)$ bits and evaluation time $O(1)$)?

We note that any such lower bound must be computational in nature and cannot be proven in the cell probe model, since in the cell probe model by definition any computation on $O(\log n)$ bits could be done in $O(1)$ time.

Acknowledgment

We thank Moni Naor for many inspiring discussions.

References

[ABK+99] Y. Azar, A. Z. Broder, A. R. Karlin, and E. Upfal. Balanced allocations. *SIAM Journal on Computing*, 29(1):180–200, 1999.

[ADM+99] N. Alon, M. Dietzfelbinger, P. B. Miltersen, E. Petrank, and G. Tardos. Linear hash functions. *Journal of the ACM*, 46(5):667–683, 1999.

[AGH+92] N. Alon, O. Goldreich, J. Håstad, and R. Peralta. Simple construction of almost k-wise independent random variables. *Random Structures and Algorithms*, 3(3):289–304, 1992.

[BR94] M. Bellare and J. Rompel. Randomness-efficient oblivious sampling. In *Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science*, pages 276–287, 1994.

[CG88] B. Chor and O. Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM Journal on Computing*, 17(2):230–261, 1988.

[CW79] L. Carter and M. N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18(2):143–154, 1979.

[DMadH90] M. Dietzfelbinger and F. Meyer auf der Heide. A new universal class of hash functions and dynamic hashing in real time. In *Proceedings of the 17th International Colloquium on Automata, Languages and Programming*, pages 6–19, 1990.

[DP08] M. Dietzfelbinger and R. Pagh. Succinct data structures for retrieval and approximate membership. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming*, pages 385–396, 2008.

[DP09] D. P. Dubhashi and A. Panconesi. Concentration of Measure for the Analysis of Randomized Algorithms. Cambridge University Press, 2009.

[DR09] M. Dietzfelbinger and M. Rink. Applications of a splitting trick. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming*, pages 354–365, 2009.

[DW03] M. Dietzfelbinger and P. Woelfel. Almost random graphs with simple hash functions. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, pages 629–638, 2003.

[FKS84] M. L. Fredman, J. Komlós, and E. Szemerédi. Storing a sparse table with $O(1)$ worst case access time. *Journal of the ACM*, 31(3):538–544, 1984.

[GW97] O. Goldreich and A. Wigderson. Tiny families of functions with random properties: A quality-size trade-off for hashing. *Random Structures and Algorithms*, 11(4):315–343, 1997.

[Hag98] T. Hagerup. Sorting and searching on the word RAM. In *Proceedings of the 15th Annual Symposium on Theoretical Aspects of Computer Science*, pages 366–398, 1998.

[HMP01] T. Hagerup, P. B. Miltersen, and R. Pagh. Deterministic dictionaries. *Journal of Algorithms*, 41(1):69–85, 2001.

[ILL89] R. Impagliazzo, L. A. Levin, and M. Luby. Pseudo-random generation from one-way functions. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 12–24, 1989.

[INW94] R. Impagliazzo, N. Nisan, and A. Wigderson. Pseudorandomness for network algorithms. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, pages 356–364, 1994.

[Lu02] C.-J. Lu. Improved pseudorandom generators for combinatorial rectangles. *Combinatorica*, pages 417–434, 2002.

[Mil99] P. B. Miltersen. Cell probe complexity - a survey. In *Proceedings of the 19th Conference on the Foundations of Software Technology and Theoretical Computer Science, Advances in Data Structures Workshop*, 1999.

[MU05] M. Mitzenmacher and E. Upfal. Probability and Computing: Randomized Algorithms and Probabilistic Analysis. Cambridge University Press, 2005.

[MV08] M. Mitzenmacher and S. Vadhan. Why simple hash functions work: Exploiting the entropy in a data stream. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 746–755, 2008.

[Nis92] N. Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992.

[NN93] J. Naor and M. Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM Journal on Computing*, 22(4):838–856, 1993.

[NZ96] N. Nisan and D. Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–52, 1996.

[PP08] A. Pagh and R. Pagh. Uniform hashing in constant time and optimal space. *SIAM Journal on Computing*, 38(1):85–96, 2008.

[PPR07] A. Pagh, R. Pagh, and M. Ružić. Linear probing with constant independence. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing*, pages 318–327, 2007.

[PR04] R. Pagh and F. F. Rodler. Cuckoo hashing. *Journal of Algorithms*, 51(2):122–144, 2004.

[PT11] M. Pătraşcu and M. Thorup. The power of simple tabulation hashing. To appear in *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing*, 2011.

[Sie04] A. Siegel. On universal classes of extremely random constant-time hash functions. *SIAM Journal on Computing*, 33(3):505–543, 2004. A preliminary version appeared in *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science*, pages 20–25, 1989.

[SS90] J. P. Schmidt and A. Siegel. The analysis of closed hashing under limited randomness. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 224–234, 1990.

[Vöc99] B. Vöcking. How asymmetry helps load balancing. *Journal of the ACM*, 50(4):131–140, 1999.

[Woe06] P. Woelfel. Asymmetric balanced allocation with simple hash functions. In *Symposium on Discrete Algorithms*, pages 424–433, 2006.

[Zuc96] D. Zuckerman. Simulating BPP using a general weak random source. *Algorithmica*, 16(4/5):367–391, 1996.